Høgskulen på Vestlandet

BACHELOROPPGAVE

Design og utvikling av en testplattform for utviklerkandidater

Design and development of a test platform for developer candidates

Anders-Marius Westum Gjerdalen
Anine Bjørnetun Hammersborg
Birk Johannessen

Dataingeniør
Fakultet for ingeniør- og naturvitskap (FIN)
Institutt for datateknologi, elektroteknologi og realfag
Veileder: Violet Ka I Pun

Innleveringsdato: 22.05.2023

Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel:	*Dato:*
Design og utvikling av en testplattform for utviklerkandidater	*22.05.2022*
Forfatter(e):	*Antall sider u/vedlegg: 44*
Anders-Marius Gjerdalen, Anine Hammersborg og Birk Johannessen	*Antall sider vedlegg: 12*
Studieretning:	*Antall disketter/CD-er:*
Dataingeniør	*0*
Kontaktperson ved studieretning:	*Gradering:*
Violet Ka I Pun	*A-F*
Merknader:	

Oppdragsgiver:	*Oppdragsgivers referanse:*
Wide Assessment AS	
Oppdragsgivers kontaktperson:	*Telefon:*
Andreas Hammerbeck	+47 473 72 074

Sammendrag:

Design og utvikling av en testplattform for utviklerkandidater er et bachelorprosjekt som tar for seg hvordan å utforme en webapplikasjon, der kandidater kan sende inn kodeutfordringer til rekrutteringshensikter. Problemstillingen for prosjektet ble «Hvordan kan prosjektgruppen utvikle en testplattform som effektiviserer en ansettelsesprosess innenfor IT-bransjen?». Det er brukt ingeniørfaglige metoder som Lean startup for å oppnå en funksjonell testplattform som kan benyttes til mange hensikter. Skalering til brukermasser og datasikkerhet er utfordringer som prosjektet har håndtert for å oppnå et tilfredsstillende resultat. Videre er det gjennomført evaluering mot arbeidet som er utført, med brukerevaluering etter SUS-standarden og tilhørende evaluering mot prosjekteier. Prosjektet konkluderer med at arbeidet kan effektivisere en jobbsøknadsprosess.

Stikkord:

API	Webutvikling	Design

Høgskulen på Vestlandet, Fakultet for ingeniør- og natuvitskap
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

Det har vært en inspirerende, utfordrende og samtidig lærerik reise å skrive denne bacheloroppgaven. Vi ønsker å uttrykke vår takknemlighet til vår veileder, Violet Ka I Pun, for hennes verdifulle råd og samarbeid gjennom hele prosjektperioden.

En spesiell takk går til Andreas Hammerbeck, teknisk direktør i WA.works, for hans tekniske innspill, samt Michelle Sæther, teknisk direktør i Link Utvikling, for hennes veiledning innen grafisk design. Vi vil også uttrykke vår takknemlighet til alle andre som har bidratt med verdifulle synspunkter og tilbakemeldinger.

Vi håper at denne oppgaven vil være til nytte for alle som er interessert i kodeutfordringer. Prosjektet gir et unikt innsyn i hvordan disse utfordringene fungerer og hvordan de kan bli utformet.

INNHOLDSFORTEGNELSE

Ordliste

Begrep	Forklaring
Grensesnitt	Et grensesnitt refererer til den delen av et system eller programvare som lar brukeren kommunisere med systemet eller programvaren.
UI/UX	User Interface – elementer brukere kan interagere med. User Experience – brukeropplevelse.
CSS	Cascading style sheets. Et stilformateringsspråk. Gir dokumenter et stilfullt innhold. Blir brukt på nettsider.
GitHub	GitHub er en webbasert plattform som gir utviklere og organisasjoner et sted å lagre, dele og samarbeide om kodeprosjekter og programvareutvikling.
Først-inn-først-ut (FIFO)	Det er en metode for håndtering av data eller elementer i en sekvensiell rekkefølge basert på prinsippet om at det første elementet som blir lagt til eller opprettet, også blir det første elementet som blir behandlet eller fjernet
Vektbalanserer (Load Balancer)	En enhet som distribuerer nettverkstrafikk jevnt mellom flere servere for å optimalisere ytelse og sikre påliteligheten i en nettverksinfrastruktur.
JavaScript (JS)	JavaScript, et programmeringsspråk som blir brukt på nettsider.
Server	En server er en datamaskin eller et program som gir tjenester eller ressurser til andre datamaskiner eller programmer som kalles klienter.
Kodeinjeksjon	Code injection, også kjent som injeksjonsangrep, er en type angrep der en angriper kan injisere skadelig kode i et program eller en applikasjon. Dette kan gjøres ved å utnytte sårbarheter i programvaren, eller ved å manipulere data som sendes til applikasjonen.
HTTP-Request manipulating,	Er en teknikk som brukes av angripere for å manipulere HTTP-forespørsler som sendes til en webserver.
REST API (Representational State Transfer Application Programming Interface)	Det er et grensesnitt som tillater kommunikasjon og utveksling av data mellom forskjellige systemer.
Kompilator	Det er et programvareverktøy som brukes til å konvertere kildekode skrevet i et høynivåspråk til maskinkode som kan utføres av datamaskinen.
React	Et JavaScript-rammeverk der man kan lage og gjenbruke grafiske modeller.
Klientside	Klientsiden refererer til den delen av et datasystem som brukerne direkte samhandler med, og som viser brukergrensesnittet. Frontend består typisk av teknologier som HTML, CSS og JavaScript som

	brukes til å designe og implementere brukergrensesnittet og funksjonalitet.
Serverside	Serversiden refererer til den delen av et datasystem som ligger bak brukergrensesnittet (klientsiden) og som ikke er synlig for brukerne. Serversiden omfatter de teknologiene, serverne, databasehåndteringssystemene og applikasjonslogikken som muliggjør funksjonaliteten og datahåndteringen i et datasystem.
Rammeverk	Et rammeverk er et strukturert sett med retningslinjer, konsepter, standarder og praksiser som brukes til å håndtere og løse komplekse problemer eller utfordringer innenfor en bestemt kontekst. Et rammeverk kan også kalles en modell eller en referansearkitektur. Eksempel React, Vue og Angular.
Super-sett	En samling av datatyper eller klasser som inkluderer alle elementene fra en annen samling, i tillegg til flere elementer utenfor.
Spring, .NET og Node.	Spring, .NET og Node er rammeverk for å bygge programvareapplikasjoner, som tilbyr robusthet, allsidighet og effektive utviklingsfunksjoner.
Relasjonsdatabase	MariaDB, PostgreSQL; Microsoft SQL Server; Det er en type database som brukes innenfor IT for å lagre og organisere data på en strukturert måte. Den bruker tabeller og modeller for å representere og relatere til data
Minimal Viable Product (MVP)	MVP refererer til en tidlig versjon av et produkt eller en tjeneste som er laget med minimale funksjoner, men som fortsatt er funksjonell nok til å gi en verdi til brukerne.
Smidig utvikling (Agile development)	Smidig utvikling er en tilnærming innen programvareutvikling som fokuserer på kontinuerlig tilpasning og samarbeidende arbeidsmetoder. Den legger vekt på iterativ utvikling, hyppig levering av funksjonalitet, fleksibilitet til å håndtere endringer og en kultur som fremmer samarbeid mellom teammedlemmer og kunder.
Figma	Figma er en design- og prototypingplattform som brukes i IT-bransjen.
Foreign key (Fremmednøkkel)	Det refererer til en kolonne eller en gruppe av kolonner i en databse som etablerer en relasjon eller kobling mellom to tabeller.
Single Page Application (SPA)	SPA er en type nettapplikasjon der all funksjonaliteten til applikasjonen lastes inn i én enkelt side, vanligvis ved hjelp av JavaScript og AJAX-teknologi.
Websocket	Det er en kommunikasjonsprotokoll som brukes i IT for sanntidsinteraksjon mellom en nettleser og en webserver.

TCP-kommunikasjonskanal	Det er en viktig protokoll i IT som brukes for å etablere pålitelig kommunikasjonskanaler mellom enheter på et datanettverk.
Migrasjonsobjekt	Det er en enhet eller en ressurs som skal flyttes fra en eksisterende IT-infrastruktur eller system til en annen.
Klassemodeller	Det er en metode for å representere og definere strukturen og oppførselen til objekter i et system ved hjelp av klasser.
Frakoble (decouple)	Decouple er et begrep som brukes når man skiller eller frakobler to eller flere ting som vanligvis er sammenkoblet eller avhengige av hverandre. Det innebærer å bryte eller redusere koblingen eller forbindelsen mellom dem.
Hangfire	Hangfire er en åpen kildekode bibliotek for .NET-utviklere som gir en enkel og pålitelig måte å utføre bakgrunns oppgaver i en applikasjon.
Kontroller-objekter (Controllers)	En metode i et program som er inngangspunktet fra internett. Gjerne returnerer den svaret som programmet skal utregne.
Skalere horisontalt	Det er en metode for å øke kapasiteten eller ytelsen til et system ved å legge til flere enheter i en horisontal retning, for eksempel ved å legge til flere servere i et nettverk.
Hardkode	Å skrive inn data eller konstanter direkte i koden på en fast måte, fremfor å hente dem fra en variabel.
Sandkassemiljø (sandbox environment)	Er et isolert og kontrollert IT-miljø der utviklere eller testere kan eksperimentere, teste programvare og utføre ulike operasjoner uten å påvirke det virkelige produksjonsmiljøet.
Microsoft Visual Studio Code (VScode)	Det er et kraftig og allsidig koderedigeringsprogram som tilbyr en rekke funksjoner som er nyttige for utviklere.
Åpen kildekode	Det er kode som er tilgjengelig for offentligheten og kan bli undersøkt, modifisert og distribuert til alle som er interessert.
CLIwrap	CLIwrap er et programvarebibliotek eller verktøy som gir et grensesnitt for å pakke inn kommandolinjeprogrammer med en høyere nivå-API. Det forenkler prosessen med å utføre kommandolinjekommandoer fra et program.
Intellisense	Det er en intelligensbasert funksjon som gir utviklere kontekstuell hjelp og forslag mens de skriver kode.
Syntaksutheving	Det er en funksjonalitet som brukes i tekstredigeringsprogrammer og utviklingsmiljøer for å gjøre koden mer lesbar ved å markere og fargelegge deler av koden i henhold til syntaktiske regler.

AI	AI (artificial intelligence) står for kunstig intelligens, som beskriver datamaskiners eller maskinsystemers evne til å utføre oppgaver som vanligvis krever menneskelig intelligens.
Monaco-grensesnitt	Det er et koderedigeringsverktøy som er utviklet av Microsoft.
Navigasjonsbar	En navigasjonsbar, eller navigasjonsmeny, er en grafisk brukergrensesnittkomponent som gir brukeren en måte å navigere gjennom en nettside eller applikasjon på.
Enhetstest	En programmert test som verifiserer om en modul, oppfører seg som forventet.

Figurliste

Benytter formatet (x-y) fremfor (x.y) på grunn av tekniske utfordringer med Microsoft Word. Hvor x representerer kapittel og y figur nummer løpende i kapittelet.

Tabelliste

1 INNLEDNING

1.1 Kontekst

De siste 30 årene har samfunnet opplevd en betydelig digitaliseringsrevolusjon, som har ført til økt etterspørsel etter personer med ferdigheter innen design, utvikling og vedlikehold av teknologiske løsninger (Røtnes, et al., 2021). I dagens teknologiske samfunn er det avgjørende for en bedrift å ansette de riktige kandidatene med de nødvendige ferdighetene for å kunne drive en vellykket virksomhet. Det kan imidlertid være utfordrende å identifisere og velge den beste kandidaten blant et stort antall søkere.

For å tiltrekke seg dyktige IT-kandidater bruker bedrifter en rekke rekrutteringsmetoder som inkluderer stillingsannonser, sosiale nettverk, karrieremesser og bedriftspresentasjoner. Eksterne rekrutteringsbyråer som finner kandidater på vegne av bedrifter, er også vanlige. Disse metodene kan bidra til å gi bedriftene en bedre oversikt over kandidatene og deres kompetanse. Tradisjonelt har bedrifter lagt stor vekt på kandidaters utdanningsbakgrunn og erfaring, spesielt når det gjelder tekniske stillinger. Derimot er det blitt stadig vanligere å ansette utviklere uten relevant utdanning. Bedrifter ser etter alternative måter for å vurdere kandidatens ferdigheter (Stokke, 2021). Prosjektgruppen fikk derfor i oppgave å lage et tilskudd til dagens rekrutteringsmetoder i form av en testplattform.

1.2 Motivasjon og mål

Bachelorprosjektet ble igangsatt på et tidspunkt som er svært relevant for et samfunn i konstant endring og et økende behov for optimaliserte rekrutteringsprosesser. Selv om det finnes velprøvde metoder for rekruttering, er det viktig å være oppmerksom på de kontinuerlige endringene og utfordringene som preger dagens arbeidsmarked innenfor IT-bransjen. For å holde følge med den raske utviklingen og sikre en konkurransedyktig rekrutteringsstrategi, er det nødvendig å identifisere og implementere nye verktøy. Dette kan forbedre rekrutteringsprosessen og sikre en effektiv og vellykket tilnærming til ansettelse.

Det foreligger flere faktorer til at en testplattform for IT-kandidater kan ha et forretningspotensiale. En av faktorene kan være at en testplattform bidrar til å identifisere de mest kvalifiserte kandidatene tidlig i prosessen. Dette kan bidra til kostnadsbesparelser for bedriftene ved å redusere antall intervjurunder. En ytterligere faktor vil være at kandidatene får muligheten til å demonstrere tekniske ferdigheter gjennom praktiske oppgaver, som kan gi bedrifter en realistisk vurdering av deres evner. Det kan gi en indikasjon på hvordan kandidatene faktisk vil prestere i arbeidssituasjoner, i motsetning til å kun basere seg på bakgrunn eller erfaring.

Ved å ha et sett med oppgaver på en felles testplattform som alle kandidater må gjennomgå, blir det mulig å sammenligne og vurdere kandidatene på en objektiv måte. Dette kan bidra til å minske risikoen for skjev vurdering eller ubevisst påvirkning i en ansettelsesprosess. For å unngå slike utfordringer kan tydelige kriterier og evalueringsmetoder bistå bedrifter med å ta en presis beslutning i løpet av en ansettelsesprosess.

En motivasjon med prosjektet er derfor å identifisere kvalifiserte kandidater uavhengig av utdanning og erfaring. For å kunne vurdere kandidaten, vil prosjektet presentere en testplattform hvor kandidater kan fremvise sine tekniske løsninger. Testplattformen vil gi kandidatene en tilbakemelding basert på hvor godt gjennomført løsningene er, og dermed gi bedriftene en mer nøyaktig oversikt over kandidatens evner og ferdigheter.

Målet for prosjektet ble å utvikle en testplattform for kodetester hvor bedrifter kan finne de rette kandidatene for stillingene sine. Prosjektet ble delt opp i delmål, hvor det første ble å utvikle et funksjonelt og brukervennlig grensesnitt. Det andre delmålet ble å sikre en likeverdig vurdering av tester for hver enkelt kandidat.

I kombinasjon med konteksten i dagens arbeidsmarked og viktigheten av å finne den kvalifiserte kandidat, resulterte det i at prosjektgruppen ville undersøke denne problemstillingen: «**Hvordan kan prosjektgruppen utvikle en testplattform som effektiviserer en ansettelsesprosess innenfor IT-bransjen?»**

Bachelorprosjektet kan knyttes opp mot flere relevante forskningsspørsmål:

1. «Hvordan utvikle en testplattform som konsekvent kan sammenligne forskjellige løsninger?» Prosjektgruppen må derfor utforske hva som spesifikt gjør en testplattform upartisk slik at resultater kan sammenlignes og formidles objektivt.

2. «Er testplattformen egnet til å skalere for en større brukerbase?» Prosjektgruppen ønsker å levere et produkt som kan skaleres, fordi tidligere prosjekter har møtt problemer når produktet får større brukerbase(Finley, 2014).

3. «Hvordan utvikle en testplattform som er utvidbar for flere programmeringsspråk?» Prosjektgruppen ønsker å levere et produkt som kan på et senere tidspunkt støtte flere programmeringsspråk. Grunnen til det er at plattformen skal kunne dekke et større mangfold for både kandidater og bedrifter.

4. «Er testplattformen forståelig og brukervennlig?» Moderne nettsider har generelt bedre brukeropplevelser enn det som ble utviklet før. Brukere har derfor høyere forventinger til UI- og UX-design (Big Fish, 2021)Dermed er det ønskelig med en testplattform som har et godt utviklet brukergrensesnitt.

1.3 Prosjekteier

Prosjekteier er Wide Assessment (WA), et digitalt rekrutteringsfirma som spesialiserer seg innenfor IT-bransjen. WA ble stiftet i 2016 med mål å omforme det som var et marked dominert av rekrutteringsbyråer (Andreassen, 2023). Det førte til utviklingen av Wide Assessment Works (WA.works) en digital rekrutteringsplattform som er skreddersydd for IT-bransjen og den høye etterspørselen etter teknologer (WA.works, 2023).

WA.works snur den tradisjonelle søknadsprosessen ved å fokusere mer på kandidatjakt enn jobbjakt. Det er dermed blitt enkelt for bedrifter å finne talentfulle og erfarne utviklere, designere, prosjektledere og andre IT-relevante kandidater. Det er med andre ord en plattform som har tilpasset seg situasjonen i IT-arbeidsmarkedet (WA.works, 2023).

På denne digitale plattformen har kandidater mulighet til å lage en personlig profil og utfører en selvevaluering, som resulterer i en digital CV. Selvevalueringen er en sentral del av plattformen der kandidater får fremlagt sine IT-relevante ferdigheter og evaluert sitt ferdighetsnivå.

1.4 Oppbygging av rapporten

Kapittel 1 har til hensikt å gi leseren av rapporten en forståelse av prosjektets kontekst og motivasjon, samtidig som prosjekteier, mål og problemstilling defineres. Videre i Kapittel 2, vil bakgrunn av prosjektet fremstilles, sammen med tilhørende litteratur. I kapittel 3 vil det bli diskutert forskjellige løsningsforslag, hvor den endelige løsningen begrunnes. I tillegg vil ingeniørfaglig metode bli beskrevet, og tekniske verktøy bli valgt.

I kapittel 4 vil det bli gitt en beskrivelse av produktets design, med formål om å gi leseren en enhetlig forståelse av hva som ble utviklet av prosjektgruppen. Videre i kapittel 5 vil resultatene bli presentert og evaluert. Diskusjon rundt fremgangsmåte, konsekvenser og resultat vil forekomme i kapittel 6. Rapporten avsluttes i kapittel 7 med en konklusjon av resultatet, og fremlegger forslag til framtidig arbeid.

2 PROSJEKTBESKRIVELSE

I dette kapittelet presenteres prosjektets bakgrunn sammen med prosjekteiers opprinnelige ideer, krav, ressurser og avgrensinger. Avslutningsvis legges det frem relevant litteratur om prosjektet.

2.1 Praktisk bakgrunn

2.1.1 Tidligere arbeid

Eksternt arbeid

Ideen om en testplattform er ikke ny, nettsider som Codewars og HackerRank er godt etablerte plattformer som brukere kan benytte for å teste kodeferdigheter (Codewars u.å.; HackerRank, u.å). Begge plattformene tilbyr praktiske oppgaver for å hjelpe brukere med individuell forbedring av ferdigheter innenfor IT-faget. Til tross for likheter mellom plattformene, har de ulike funksjoner og formål. Codewars er et nettsamfunn hvor kodetestene er laget av og for brukerne. I HackerRanks tilnærming til rekruttering, er plattformen designet slik at bedriftene selv utformer kodetestene, og dermed legger til rette for direkte rekruttering gjennom plattformen. Prosjektgruppen har tatt inspirasjon av disse plattformene med tanke på design og utforming av en løsning.

Internt arbeid

Prosjekteier har allerede en fungerende rekrutteringsplattform med 1043 selskaper og 16108 kandidater (WA.works, 2023). Det vil dermed ikke være behov for å utvikle et brukersystem til testplattformen. Prosjektgruppen har derfor fokus på å utvikle funksjonalitet knyttet til testplattformen. I tillegg har prosjekteier hatt et liknende prosjekt i form av en stilformaterings test (CSS-test) som en tidligere bachelorgruppe har utviklet (Surdal & Aadland, 2022). Det prosjektet blir derfor et relevant arbeid for prosjektgruppen å analysere. CSS er det standardiserte språket for å omforme nettsider fra ren tekst til stilfullt innhold. Formålet med CSS-testen var å evaluere brukernes evne til å anvende CSS-kode for å gjenskape et gitt bilde. Dette prosjekter er ikke en del av WA.works sin plattform i 2023. Senere i denne rapporten vil det begrunnes om prosjektgruppen skal videreutvikle dette arbeidet.

2.1.2 Initielle krav

I forbindelse med bachelorprosjektet presenterte prosjekteier krav for utviklingen av testplattformen, som er beskrevet i vedlegg 1. Siden de initielle kravene er et viktig utgangspunkt for utviklingsprosessen måtte kravene tydelig defineres. Det legger grunnlaget for en strukturert og målrettet tilnærming til testplattformen. Oversikten over kravene sikrer at prosjektgruppen kan fokusere på de riktige områdene. For å oppnå målene, må de nødvendige funksjonene og egenskapene implementeres. Dette er oversikten over kravene til testplattformen fra vedlegget:

- «Brukeren skal kunne velge en test, ut av en liste med tester»
- «Når en test starter, skal brukere få opp et tekstinput felt hvor man skriver 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»

Basert på de initielle kravene ble det nødvendig å identifisere prosjektets avgrensninger.

2.2 Avgrensninger

En naturlig avgrensning er prosjektperiodens varighet. Det krever betydelig tid og innsats å beherske teknologier som var relevante for prosjektet. Essensen av testplattformen måtte identifiseres og vektlegges for å oppnå ønsket funksjonalitet. Prosjektgruppen avtalte derfor møter med prosjekteier for å avgrense og tilpasse de initielle kravene. Det ble enighet om å kun implementere en testplattform som fungerer med et programmeringsspråk, men er tilrettelagt for utvidelse av flere. Videre ble det konkretisert hvordan poeng skal utdeles til brukeren av testplattformen, med å først gi brukeren tilbakemelding på hvor riktig brukerens test var, og deretter på kjøretid.

2.3 Ressurser

I løpet av prosjektperioden ble prosjektgruppen gitt tilgang til lokaler hos prosjekteier. I tillegg hadde prosjekteier også et GitHub-prosjekt med kodeeksempler og materiale fra fjorårets CSS-oppgave. For å tilegne seg grunnleggende kunnskap, er det tiltenkt å benytte kurs som er relevante til prosjektet på CodeAcademy (CodeAcademy, u.å.). Prosjekteier var alltid tilgjengelig for spørsmål og rådgivning.

For å evaluere prosjektet vil det bli gjennomført brukertester, som består av IT-studenter, utviklere fra arbeidslivet og prosjekteier. Dette utgjør en viktig faktor for en god evalueringsprosess.

2.4 Relevant litteratur

Dette delkapittelet skal gi leseren innsikt i den litteraturen som er benyttet for å sette prosjektet i et perspektiv for kostnadsbesparelser. Videre introduseres litteratur som grunnlegger mulige tekniske utfordringer til prosjektet.

2.4.1 Rekruttering

Når en IT-kandidat skal søke på en stilling, involverer det at kandidaten i forkant har sendt inn en CV og søknadsbrev. Deretter er det opp til bedriftene å evaluere og selektere ut hvem som får komme på teknisk intervju. I etterkant blir kandidater vurdert på nytt, om de får komme til neste runde med intervjuer. Dette er en gjentagende prosess til en kandidat er valgt. Dette er tidkrevende med hensyn til antall søkere på en stilling og den arbeidskraften det krever å gjennomføre en ansettelsesprosess.

Hensikten med en testplattform er å avlaste bedrifter når det kommer til en ansettelsesprosess. Ettersom dette omhandler prosjektgruppens problemstilling, ble det naturlig å finne relevant litteratur knyttet til dette. I artikkelen «Hva koster det å gjøre en rekruttering selv?», legges det frem argumenter rundt kostnaden av antall intervjurunder og reisekostnader knyttet til en rekrutteringsprosess. Disse argumentene blir demonstrert i antall timer: «Grundig gjennomgang av 50-80 søknader med vedlegg = 25 timer, 10 førstegangsintervjuer på to timer (inkl. grundige forberedelser) = 30 timer» (Sperre, 2015). Tallene viser at det er kostbart å ansette en ny kollega, og desto mer hvis vedkommende ikke var riktig person for stillingen.

2.4.2 Kodeutføreslse, vektbalansering og kø teori

Et av forskningsspørsmålene omhandlet utviklingen av et skalerbart produkt i samsvar med en økende brukermasse. I artikkelen «AccTEE: A WebAssembly-based Two-way Sandbox for Trusted Resource Accounting», beskrives det hvordan utførelse av kode fra ukjente klienter er en operasjon som ofte gjøres med mistillit, der utførende part ikke vet hva slags ressurser den kan kreve (Goltzsche et al., 2019). En mulig løsning på dette blir beskrevet videre i artikkelen «Performance, Scalability, and Semantics of Concurrent FIFO Queues» om vektbalansering (Load balancing) av flere ressurser og hvordan kø-teori som først-inn-først-ut (FIFO) er implementert for å oppnå skalerbare løsninger (Kirsch et al., 2012).

Leseren av rapporten skal etter dette kapittelet forstå bakgrunnen til prosjektet, prosjekteiers krav og være oppdatert på relevant litteratur til prosjektet. Neste kapittel skal design av prosjektet presenteres.

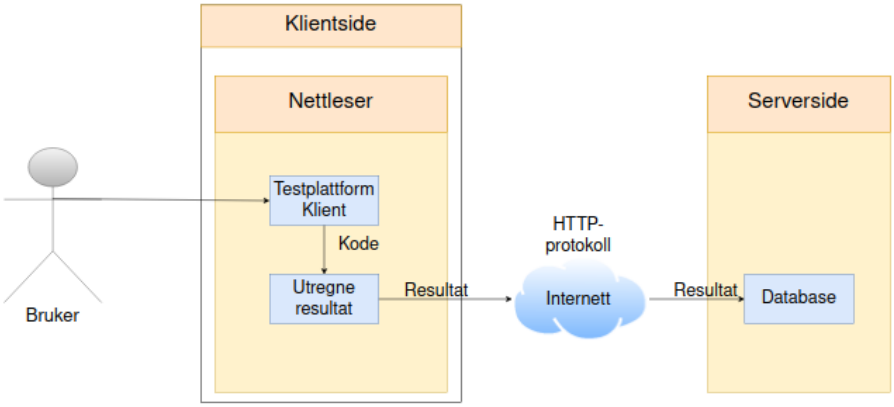
3 DESIGN AV PROSJEKTET

I dette kapittelet fremlegges mulige løsninger for testplattformen, etterfulgt av diskusjon og en begrunnelse for valgt løsning. Videre skal passende verktøy diskuteres og bestemmes. Avslutningsvis i kapittelet presenteres ingeniørfaglige prosjektmetodikker som benyttes sammen med en evalueringsplan.

3.1 Forslag til løsning

3.1.1 Integrert testplattform utført i nettleseren

Første forslag til løsning er en nettsideapplikasjon som håndterer logikk på klientsiden. Forslaget vektlegger å utregne resultatet av en kodetest i nettleseren til brukeren, noe som speiler løsningen i den nevnte CSS-testen fra punkt 2.1.1.



Figur 3-1: Flytvisualisering av første forslag

Figur 3-1 viser flyten av kodeutregning i forslaget. Resultatet blir utregnet i nettleseren og formidlet til en database på serversiden. Med dette forslaget vil prosjektgruppen kunne bygge videre på arbeidet som allerede er gjort med CSS-testen, ettersom den har samme arkitektur som Surdal og Aadland (2021). Forslaget antas å ha den korteste veien til en funksjonell testplattform, ettersom det er et klientsideforslag som har begrenset mengde serverfunksjonalitet. Prosjektgruppen kan med dette forslaget i stor grad fokusere på funksjoner på klientsiden.

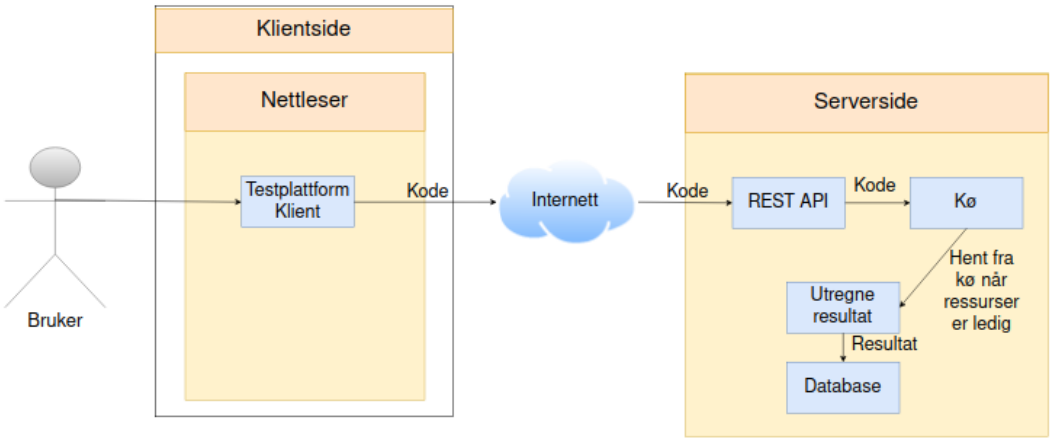
Selv om prosjektgruppen kan bruke det tidligere arbeidet med CSS-testen, er det viktig å definere forskjellen på en CSS-test og en kodetest. CSS er et stilspråk og gir derfor grafiske resultater, i motsetning til kodetest som gir et tekstresultat basert på logiske beregninger.

Forslaget har noen utfordringer, den første utfordringen omhandler utvidelse av testplattformen for å tilby flere programmeringsspråk. Dette er fordi nettlesere er designet for å utføre kode på Javascript (JS) (Rossen, 1997). En annen utfordring med forslaget er under utregningen av resultatet. Kjøretiden til programmet kan bli påvirket av hastigheten til klientens datamaskin, den er vist som klientside i figur 3-1. Dermed kan en kandidat tilegne seg en urettferdig fordel ved å ha en kraftig datamaskin under levering av løsningen. For eksempel kan en kandidat få lavere kjøretid enn en

annen med samme løsning. Den siste utfordringen med løsningen er å hindre at en klient kan manipulere resultatet av en test. Når resultatet blir kalkulert på klientsiden kan en kandidat manipulere en HTTP-forespørsel (HTTP-Request manipulering) for å forfalske resultatet av egen test.

3.1.2 Integrert testplattform utført på serverside

Det neste forslaget har noen likhetstrekk med det første, men skiller seg imidlertid ut ved utførelse av brukerkode på serversiden. Forslaget har en enkel klientside og kompleks serverside, der klientsiden fungerer som en formidler mellom bruker og serversiden. Arbeidet til klientsiden vil derfor primært være å sende kandidatens testkode til serveren. Serversiden vil ha et REST API (Representational State Transfer Application Programming Interface) som klienten kan kommunisere med for å motta tester og behandle brukerdata.



Figur 3-2: Flytvisualisering av andre forslag

Figur 3-2 viser en flyt for å utregne et resultat for forslaget. Som nevnt i relevant litteratur i punkt 2.4.2 kan ikke serversiden vite hva slags ressurser som kreves for brukerkoden. Derfor kan et REST API bli overlastet når en brukermengde krever mer ressurser enn hva som er tilgjengelig på serversiden. Konsekvensen av dette er nedetider til testplattformen. Litteraturen nevner videre en løsning med vektbalansering av ressurser ved å bruke en køstruktur. Videre i figur 3-2 viser serversiden hvordan serverressurser henter kode fra en kø struktur, utregner resultatet og lagrer i databasen. Ved å ta i bruk dette forslaget, vil det tilrettelegges for en situasjon der serveren kontrollerer mengden ressurser brukerkoden får under kjøretid. Det tillater en objektiv sammenligning av utførte tester.

Den største utfordringen til forslaget er hvordan det må forholde seg til kodeinjeksjon (Code injection). Kodeinjeksjon er en angrepsvinkel til en ond aktør som vil hacke en tjeneste. Dette er et av de farligste sikkerhetshullene som en kan finne i en applikasjon (OWASP, 2021). For å sikre serveren mot angrep bør sikkerhetstiltak identifiseres for å minske angrepsoverflaten.

3.1.3 Selvstendig programvareplattform

Det siste forslaget til løsning er forskjellig fra de to forrige ved at den ikke integreres mot WA.works. Forslaget er en programvare som brukeren må laste ned på egen datamaskin. Dette forslaget kan både benytte seg av klient- og serverside -utførelse og møter dermed de samme utfordringene som i de tidligere forslagene.

I motsetning til første forslag – som møtte utfordringer i utvidelsen av programmeringsspråk - er det nå flere muligheter for en programvareløsning. I dette forslaget kan kompilatorene til de forskjellige programmeringsspråkene bli lastet ned sammen med programvaren. En utfordring dette forslaget møter er å forholde seg til løsninger i flere forskjellige operativsystemer. Det vil være behov for flere forskjellige utgaver av programvaren på Windows, MacOS og Linux.

Etter å ha gått gjennom de ulike alternative løsningene, skal fordeler og ulemper ved hver enkelt løsning diskuteres.

3.1.4 Diskusjon av alternativene

Forslagene som er presentert har både fordeler og ulemper. Prosjektgruppen skal hovedsakelig bestemme om det skal bygge videre på arbeidet som ble gjort mot CSS-testen, eller utvikle et av forslagene fra bunnen av.

I vurderingen om det første forslaget - Integrert testplattform utført i nettleseren – ble det identifisert hvordan prosjektgruppen kunne raskt utarbeide en funksjonell testplattform. I dette forslaget kunne det potensielt bygges videre på CSS-testen, men arbeidet ble vurdert som lite gjenbrukbart i forhold til prosjektets hensikter mot en testplattform. Det ble også avklart i kodebasen hvordan CSS-testen manglet serverfunksjonalitet, som betyr at dette arbeidet må utvikles fra bunn. Arkitekturen til forslaget møter problematikk med manipulasjon av HTTP-forespørsel. Ved å utnytte denne arkitekturen kan brukere av testplattformen forfalske resultater, som kan føre til mistillit. Videre er det utfordrende å utvide tilgjengelige programmeringsspråk i denne arkitekturen, da den er begrenset til verktøyene i nettleseren. Dette kan påvirke noen brukeres evne til å konkurrere, da de ikke nødvendigvis har JS erfaring.

Det neste forslaget – integrert testplattform utført på serverside – har arkitektur som eliminerer problematikken rundt utvidelse av programmeringsspråk og HTTP-manipulasjon, men introduserer et nytt problem rundt kodeinjeksjon. Det er identifisert hvordan lignende løsninger som Codewars har samme oppførsel i arkitekturen som forslaget. Denne informasjonen ble funnet etter en analyse av HTTP-trafikken på nettsiden til Codewars. For å ferdigstille prosjektet med dette forslaget må prosjektgruppen beherske flere teknologier, noe som er utfordrende å tillære seg.

Siste forslag til løsning - selvstendig programvareplattform - introduserer et alternativ til en integrert webløsning, som de andre forslagene vektlegger. Prosjektgruppen vurderer dette forslaget som bedre egnet til klientutførelse av kode enn det første forslaget, da det eliminerer problematikken rundt utvidelse av programmeringsspråk. Forslaget beholder likevel problematikken rundt HTTP-forespørsel manipulasjon, med mindre arkitekturen fra forslag to – integrert testplattform utført på serverside - benyttes. Det er imidlertid identifisert som utfordrende for prosjektgruppen å utvikle en

løsning for flere operativsystemer, da medlemmene ikke har erfaring med programvareutvikling.

3.2 Valgt løsning

Etter diskusjon av forslagene og samtaler med prosjekteier, konkluderte prosjektgruppen med forslaget fra punkt 3.1.2 - integrert testplattform utført på serverside - som det beste alternativet. Faktorene for valget er basert på diskusjonen som er fremlagt, der hovedargumentene var en testplattform som støtter flere programmeringsspråk, har potensialet for å gi en objektiv poengsum og som har tillit fra brukeren. Det ble identifisert hvordan kodeinjeksjon kunne være en trussel i valgt løsning, noe prosjektgruppen må finne tiltak for under utviklingen av testplattformen.

3.3 Valg av verktøy

I diskusjonen ble det identifisert hvordan den valgte løsningen krever anvendelse av flere teknologier, som skal defineres i dette delkapittelet. Prosjektgruppen hadde fritt utgangspunkt til å velge passende teknologier for både klientsiden og serversiden.

Først skal prosjektgruppen identifisere passende rammeverk for utviklingen av klientsiden. Verktøy som React, Vue og Angular har blitt diskutert. Medlemmer av prosjektgruppen har noe erfaring med Vue rammeverket fra et fag på studieplanen, og er derfor et potensielt valg som kan sikre fremgang under utvikling. React er også et egnet verktøy som er populært i arbeidsmarkedet og er dermed nyttig for medlemmene av prosjektgruppen å tillære seg.

På serversiden har prosjektgruppen et bredt teknologiutvalg å velge mellom, det skal gjøres valg av rammeverk for kommunikasjon og relasjonsdatabase. Spring, Node og .NET er blant de vurderte alternativene for kommunikasjonsrammeverk, hver med sine egne fordeler og karakteristikker som har blitt diskutert. Prosjektgruppen har kjennskap med de underliggende programmeringsspråkene til Spring og Node gjennom studieplanen, men ikke med selve rammeverkene. Under vurdering av passende relasjonsdatabase var MariaDB, PostgreSQL og Oracle blant de som ble diskutert. Prosjektgruppen har særlig hatt erfaring med PostgreSQL fra studiet, noe som gjorde det til et aktuelt alternativ.

Prosjekteier oppfordret teknologi som de selv bruker; React i kombinasjon med TypeScript (TS) på klientsiden sammen med .NET og PostgreSQL på serversiden. TS er et super-sett (superset) av JS, og er et verktøy som gjør det lettere å forholde seg til prinsippene i objektorientert programmering (Zolotarev, 2022; Odongo, 2021).

Teknologivalgene har blitt oppveiet av prosjektgruppen, med formål å sikre utviklingsfremgang og vurdere om valgene er i tråd med prosjektets krav. Det endelige valget falt på React med JS, .NET og PostgreSQL. Valgene ligner på oppfordringene fra oppdragsgiver, men har en justering. JS ble valgt fremfor TS med hensikt å redusere antall nye teknologier prosjektgruppen må tillære, noe som sikrer utviklingsfremgang i startfasen. Prosjekteier sine oppfordringer ble vektlagt for å sikre potensiell videreutvikling av testplattformen. I tillegg ble ekspertisen og veiledning som prosjekteier

kunne tilby rundt disse teknologiene vurdert som svært verdifulle for prosjektet. Prosjektgruppen identifiserte også teknologiene som relevante for stillinger i arbeidslivet, som er en ekstra motivasjon for prosjektgjennomføringen.

Etter valg av løsning og verktøy, skal en passende prosjektmetodikk velges.

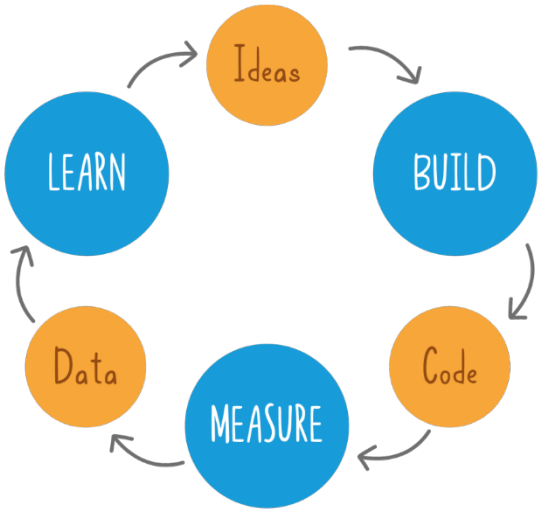
3.4 Prosjektmetodikk

Det vil i dette delkapittelet bli diskutert og valgt hvilken prosjektmetode prosjektgruppen skal følge. Videre vil det defineres en prosjektplan og en risikoanalyse vil bli utarbeidet.

3.4.1 Utviklingsmetodikk

Lean startup

Lean startup er en forretningsmetode der formålet er å redusere utviklingstiden og dermed ressursforbruket i utviklingsperioden (Raaheim, 2012). Det er en metodikk i kategorien smidig utvikling (agile development) og har tre steg for en iterativ «bygg, evaluer og lær» (Build, measure and learn) prosess. Dette er en utviklingsmetodikk med brukeren i sentrum. Hensikten er å starte med en Minimal Viable Product (MVP) basert på en hypotese, teste den og deretter videreutvikle en prototype.



Figur 3-3: Lean startup (Jimeno J.L., 2021)

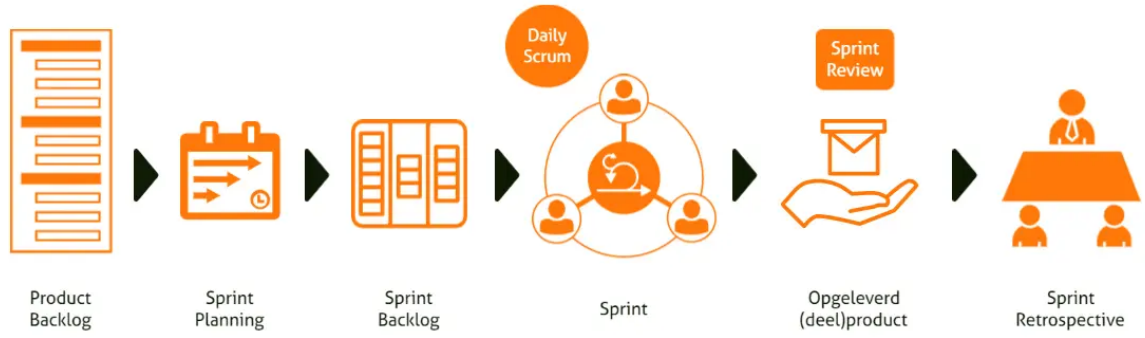
Som vist i figur 3-3 er det en syklus hvor stegene går ut på å utvikle, teste og lære av tilbakemeldingene fra brukere. Ved å benytte en slik utviklingsmetodikk kan prosjektgruppen skape en tilbakemeldingssyklus. I mange tilfeller er ikke brukere klar over hva de faktisk ønsker eller trenger, men gjennom interaksjon med produktet kan dette bli tydeligere (Sarah K. 2015).

I forkant av en brukertest blir det utarbeidet en hypotese som baserer seg på det første – build - steget. Deretter blir det viktig å lage en MVP av produktet som kan gi tilstrekkelig informasjon for å validere hypotesen. Dataen som samles inn i dette steget brukes til å analysere om hypotesen er

gyldig eller ikke. Hvis hypotesen er gyldig, skal produktet skaleres opp for å møte behovene til en større brukergruppe. Derimot om hypotesen ikke er gyldig, bør produktet endres til en annen løsning. Det avsluttende – learn - steget omhandler en kontinuerlig iterasjon av produktet med tanke på forbedringspotensialer fra brukere.

Scrum

Scrum er et rammeverk som vektlegger iterativ og inkrementell utvikling, med formål å effektivisere samarbeid og fleksibilitet i et prosjektteam. Rammeverket har definert tre hovedroller som er avgjørende for effektiv implementering av Scrum-metodikken. Den første rollen er produkteier, som er ansvarlig for å definere og prioritere kravene i produktkøen. Den andre rollen er Scrum master, der vedkommende har hovedansvaret for å veilede Scrum-teamet i sprintene. Den siste rollen er utviklingsteamet, hvor teamet selv har ansvaret for å organisere utviklingsarbeid som må gjøres for å implementere kravene (Scrum, 2023). En sprint startes av Scrum-teamet og interessenter som samler inn krav til en produktkø (backlog). Som vist i figur 3-4 kan Scrum tidslinjen deles inn i fasene: *sprint planleggingsmøte*, *daglig Scrum*, *sprint evaluering* og *sprint refleksjon*.



Figur 3-4: The Scrum Framework (Agilewerken. nl, 2023)

Sprint planleggingsmøtene markerer begynnelsen av en sprint. Her definerer utviklingsteamet og produkteier målene for sprinten. Teamet planlegger oppgaver og estimerer arbeidet som kan fullføres innenfor den fastsatte sprintperioden.

Daglig Scrum er en kort daglig statusoppdatering der teamet møtes for å diskutere fremdriften. Hver deltaker deler sitt arbeid som er utført siden forrige møte, hva de skal gjøre videre, og eventuelle hindringer de står overfor. Målet er å sikre god kommunikasjon og samarbeid innad i teamet for å raskt identifisere eventuelle problemer.

Sprint evaluering er en gjennomgang av det inkrementelle produktet som er blitt utviklet i sprinten. Teamet presenterer resultatene for interessenter, der tilbakemeldinger og kommentarer mottas. Dette møtet gir en mulighet til å vurdere produktets fremgang, validere krav og justere prioriteringer basert på tilbakemeldingene.

Sprint refleksjon er en evaluering av teamets ytelse og en refleksjon over forbedringsområder. Teamet diskuterer hva som fungerte bra i sprinten, hva som kunne blitt gjort annerledes og identifisere områder som kan forbedres. Refleksjonen er en viktig del av kontinuerlig forbedring i Scrum-prosessen, og bidrar til å styrke teamets samarbeid og effektivitet.

Videre skal prosjektgruppen velge og begrunne metode som passer til prosjektet.

Metodikkvalg

Prosjektgruppen har sett på begge metodikkene, Lean startup og Scrum som passende for prosjektet. Begge metodikkene er populære innenfor programvareutvikling og smidig prosjektledelse. Fordelene med å velge Scrum er at den gir en tydelig struktur med definerte roller og styring av prosjektet. Lean startup baserer seg på en kundesentret tilnærming, der brukerens behov og preferanser vektlegges. Det er også en metodikk som muliggjør rask testing og validasjon av ideer og hypoteser.

Hovedforskjellen mellom metodikkene er at Scrum baserer seg på å utvikle et komplett produkt, uten at det nødvendigvis samsvarer nøyaktig med det kunden ønsker. Derimot har Lean startup kunden i sentrum, og har som mål å utvikle det kunden ønsker ved hjelp av regelmessig testing.

Det ble identifisert hvordan Scrum baserer seg på daglige møter, evaluering og refleksjon. Scrum inneholder mange steg og prosesser, som kan være unødvendig for en mindre gruppe å forholde seg til. Lean startup derimot er en mindre strukturell metodikk som tillater fleksibilitet, siden den er basert på kontinuerlig testing hvor prosjektgruppen har frihet til å bestemme testintervallet.

Sett i perspektiv mot problemstillingen - om en testplattform som kan effektivisere en ansettelsesprosess - vil det være fordelaktig å benytte seg av Lean startup med tanke på kundens ønske i sentrum og tilbakemeldingssyklusen. For å effektivisere ansettelsesprosessen ser prosjektgruppen det som viktig at testplattformen er et produkt som kunden vil ta i bruk fra en eventuell lanseringsdato. Dermed er det viktig å sette brukers perspektiv i sentrum. Dette vil resultere i et produkt som har en høy sannsynlighet for at brukeren vil benytte testplattformen. I motsetning til Scrum hvor hovedfokuset er på å utvikle et funksjonelt produkt raskt for å rekke en frist selv om ikke alle brukers ønsker nødvendigvis er tatt hensyn til. Konklusjonen ble dermed at Lean startup var den best egnede utviklingsmetodikken for prosjektgruppen.

3.4.2 Prosjektplan

I planleggingsfasen ble det utarbeidet et Gantt diagram ved å anvende en mal fra Microsoft som vist i vedlegg 2. Hensikten med diagrammet var å illustrere perioden til hver enkelt arbeidsoppgave og visualisere prosjektgruppens progresjon (Gantt, 2023). Planlegging av en prosjektperiode med et Gantt-diagram krever grundig analyse av oppgavenes avhengigheter. De refererer til sammenhengen mellom oppgavene, der en oppgave må fullføres før en annen kan begynne. Disse avhengighetene danner en logisk sekvens som definerer prosjektets fremdrift. Det blir derfor viktig at prosjektgruppen identifiserer og forstår avhengigheten mellom arbeidsoppgavene.

3.4.3 Risikovurdering

Risikoanalysen i vedlegg 3 av prosjektet ble utarbeidet ved hjelp av malen fra prosjekthåndboken (HVL, 2023). Malen ble brukt som en veiledning for å evaluere forskjellige risikoer som kunne oppstå i forbindelse med prosjektets utvikling. I denne analysen ble risikoene vurdert, og deres alvorlighetsgrad beregnet ved å kombinere sannsynlighet og konsekvens. For å visualisere analysen ble risikomatrisen i tabell 3-1 anvendt. Konsekvensnivået ble multiplisert med sannsynligheten for å lage en risikofaktor som reflekterte den mulige påvirkningen.

Risikoer med en betydelig påvirkning og en høy sannsynlighet for å inntreffe ble prioritert først, slik at det ikke ble en hindring for prosjektet. Etter å ha identifisert og vurdert risikoene, ble det utviklet tiltak for å håndtere de ulike faktorene. Tiltakene i vedlegg 3 inneholdt konkrete avgjørelser for å redusere sannsynligheten for at risikoene skulle oppstå, samtidig ble strategier utarbeidet for å begrense konsekvensene som kunne forekomme.

Sannsynlighet	**Svært Høy (5)**	**5**	**10**	**15**	**20**	**25**
	Høy (4)	**4**	**8**	**12**	**16**	**20**
	Middels (3)	**3**	**6**	**9**	**12**	**15**
	Lav (2)	**2**	**4**	**6**	**8**	**10**
	Svært Lav (1)	**1**	**2**	**3**	**4**	**5**
		Svært Lav (1)	**Lav (2)**	**Middels (3)**	**Høy (4)**	**Svært Høy (5)**
	Konsekvens					

Tabell 3-1: Risikomatrise basert på prosjekthåndboken utarbeidet ved NTNU

Gjennomføringen av risikoanalysen bidro til å øke bevisstheten og forståelsen rundt prosjektets risikobilde. Det ga prosjektgruppen muligheten til å være proaktiv i sin tilnærming og ta beslutninger basert på en solid forståelse av de mulige utfordringene.

Etter å ha gjennomført risikoanalysen og økt bevisstheten rundt prosjektets risikobilde, er det på tide å definere en evalueringsplan.

3.5 Evalueringsplan

Evalueringen spilte en avgjørende rolle når det kom til å fastslå i hvilken grad testplattformen oppfylte kravene som ble satt ved prosjektstart. Som en del av denne evalueringsprosessen vil det bli gjennomført brukerevalueringer for å vurdere testplattformens brukervennlighet og funksjonaliteter. Disse involverer både IT-studenter og utviklere som prosjekteier stilte til rådighet. Brukerevaluering forekommer når prosjektet har nådd en funksjonell MVP, og vil hovedsakelig bestå av systemtesting. Det er tiltenkt å opprettholde en kontinuerlig dialog både med bruker og prosjekteier, der tilbakemeldinger og observasjoner vil bli registrert og analysert fortløpende.

Det vil være relevant å evaluere utviklere ettersom de er potensielle brukere av testplattformen. Brukerne vil bli involvert angående testplattformens brukervennlighet og deres interesse i arbeidsmarkedet. Dette kan føre til innsikt i brukerens synspunkter om brukervennligheten og deres potensielle motivasjon for å benytte testplattformen. Deres innsikt og erfaringer kan dermed gi verdifulle perspektiver, og bidra til å identifisere eventuelle problemer og forbedringspotensialer. En annen aktuell brukergruppe er IT-studenter på grunn av deres daglige interaksjon med lignende teknologier og kan være framtidens brukere av testplattformen. Studenter har mindre erfaring enn utviklere, men er likevel aktuelle brukere for å evaluere hvor intuitiv og brukervennlig den er.

I evalueringen er det planlagt å gi brukeren et evalueringsskjema etter interaksjon med testplattformen. Evalueringsskjemaet skal på formatet til System Usability Scale (SUS). SUS er et utbredt format som er laget av John Brooke og blir ofte brukt for å teste brukervennligheten til applikasjoner. Skjemaet består av ti påstander med fem svaralternativer knyttet til hvor enig brukeren er til påstanden (Brooke, 1995).

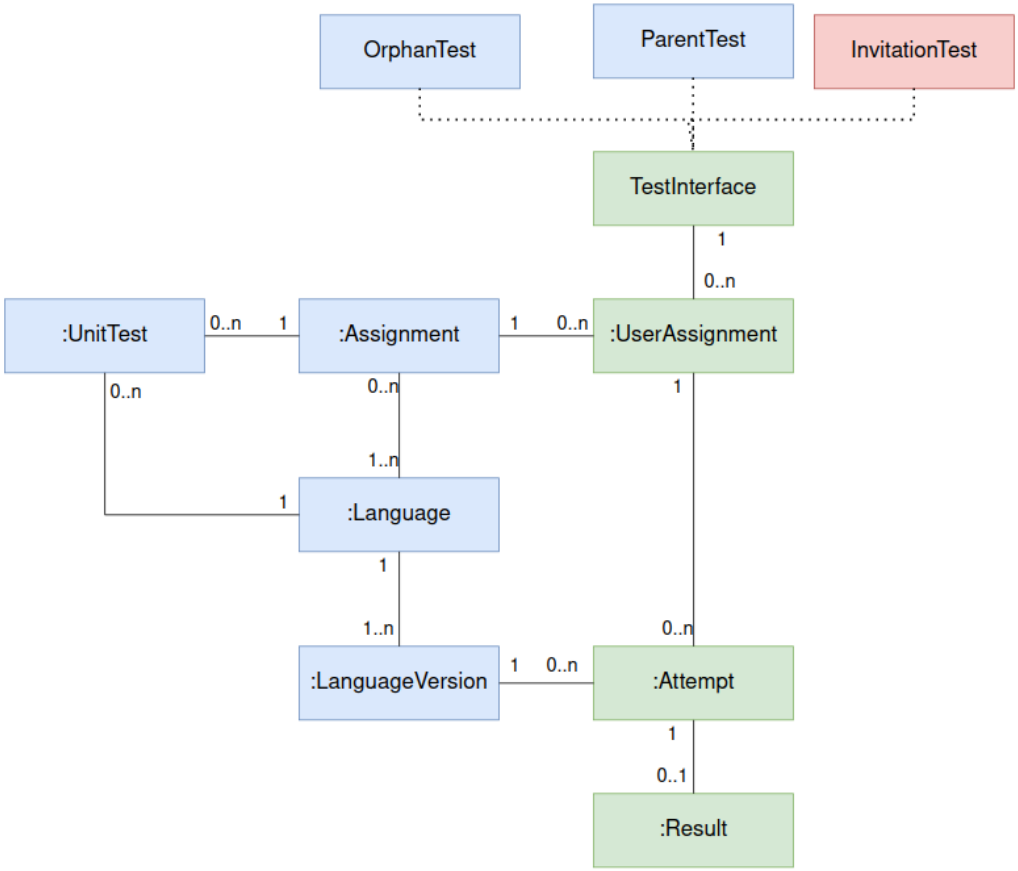
4 Detaljert løsning

I forrige kapittel ble valg av løsning, verktøy og metodikk presentert. Videre vil dette kapittelet belyse detaljer om løsninger og teknologier som er brukt til å utvikle testplattformen.

4.1 Design

Systemutvikling krever skissering av design og klasser for å oppnå en felles forståelse for hva prosjektgruppen skal utvikle. Det var essensielt å investere tid i planlegging for å oppnå utviklingsretning.

Første skissering viser et klassediagram basert på kravene til oppdragsgiver.



Figur 4-1: Klassediagram med de initielle kravene

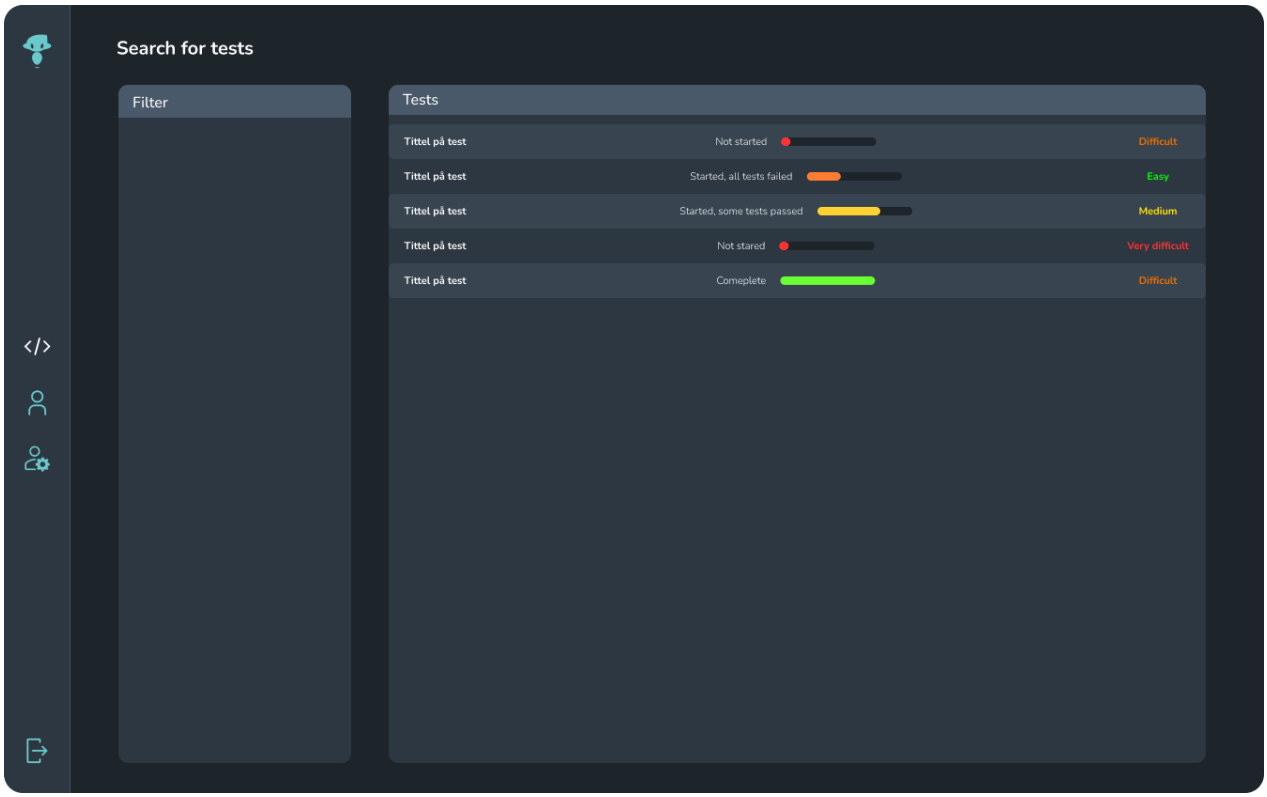
Figur 4-1 viser systemet prosjektgruppen skal utvikle. Elementene i figuren er fargekodet – hvor grønn representerer brukergenerert data og blå representerer serverdefinert data. En test (TestInterface) har et en-til-mange forhold til brukeroppgaver (UserAssignment), som videre har samme forhold til forsøk (Attempt). Hvert forsøk har et resultat som skal utregnes som et en-til-kanskje-et forhold. Årsaken til det siste forholdet omhandler hvordan forsøk asynkront beregner et resultat, og vil forekomme først i databasen. Hver oppgave (Assignment) er tilgjengelig på flere programmeringsspråk (Language) og har en tilhørende enhetstest (UnitTest). Enhetstest er en test som blir brukt til å avgjøre resultatet av et forsøk. Til slutt har hvert språk en versjon

(LanguageVersion), da brukere kan velge hvilken versjon av programmeringsspråket koden blir utført på. I figuren er det en relasjon mellom forsøk og versjon, som tillater en bruker å endre programmeringsspråk under gjennomføring av en brukeroppgave.

Skisseringen av klassediagrammet er lagt opp slik at en bruker enkelt kan velge forskjellige tester til ulike formål. Testene bruker et grensesnitt som muliggjør nye testimplementasjoner, da en ny test integreres mot grensesnittet (TestInferface). En foreldreløs test (Orphan Test) er en samling av oppgaver som brukeren eier. Data i foreldreløse tester kontrollerer brukeren selv og bestemmer hvem som eventuelt skal få innsyn. Forelder Test (Parent Test) er en samling av oppgaver som har én eier, der bedrifter på plattformen kan publisere tester til kandidater. Barnløs test eller invitasjon test (Invitation Test) er en samling av oppgaver som har én eier, men ikke en registrert bruker. En slik test tillater en bedrift å gi en vilkårlig kandidat lenker til å utføre tester uavhengig om kandidaten har registrert en bruker hos WA.works. Den er markert rød ettersom den ikke er godt nok definert i systemet. Systemet prosjektgruppen har skissert er veldig stor med mange brukstilfeller, noe som førte til en beslutning på å kun utvikle et system der foreldreløse tester kan utføres.

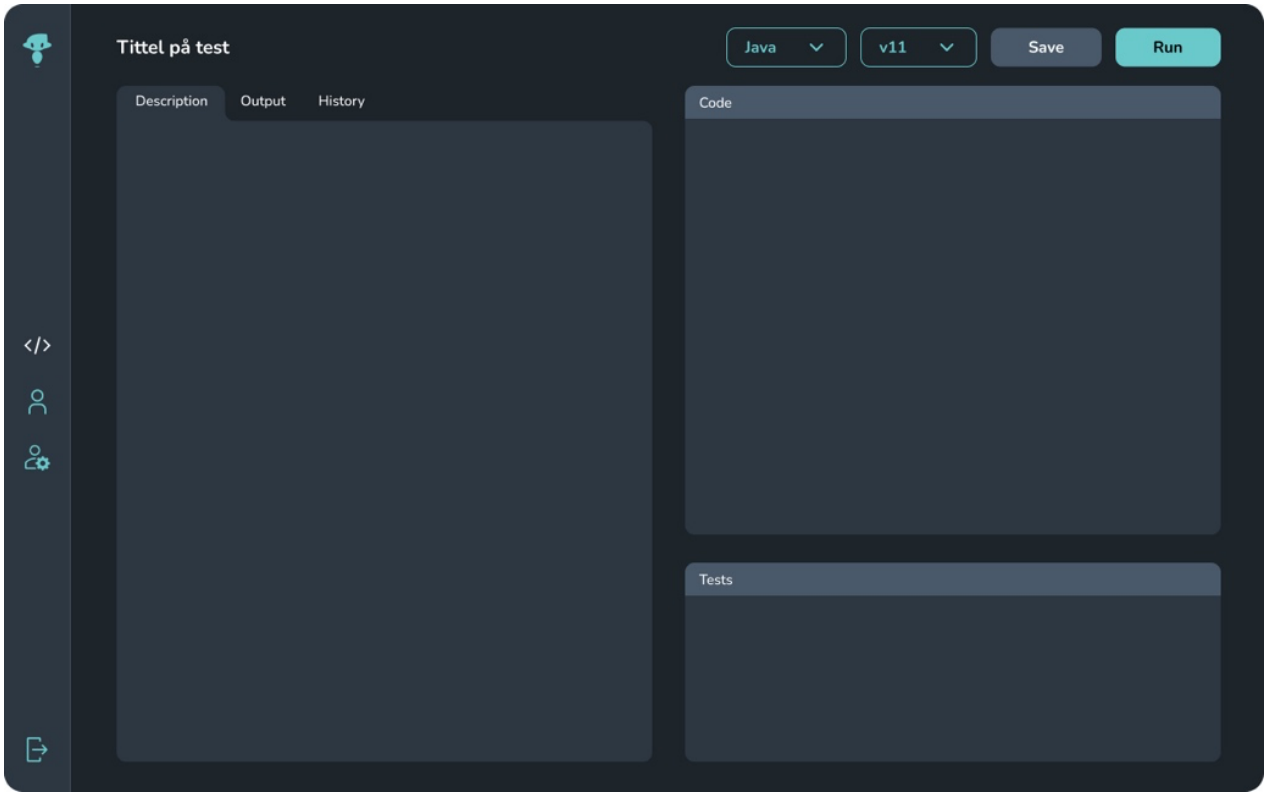
Systemet som er modellert i figur 4-1 er midlertidig uavhengig av prosjekteier sine systemer. Bruker- og bedriftsrelasjoner som nevnt i avsnittet over er tiltenkte fremmednøkler (Foreign key) som vil knytte testplattformen til prosjekteier.

Når fremstillingen av klassediagrammet var på plass, ble det naturlig å starte med skisseringen av det første utkastet av brukergrensesnittet for den tiltenkte testplattformen.



Figur 4-2: Figma-design av testoversikt

Figur 4-2 er en skisse av testoversikten på klientsiden. På denne siden velger brukeren mellom ulike oppgaver i en test, som også er vist i klassediagrammet. Brukeren vil ha oversikt over navnet til oppgaven, progresjonen og tilhørende vanskelighetsgrad. Brukeren kan bruke filteret (Filter) for å begrense fremstilte oppgaver. Når brukeren trykker på en av oppgavene, vil brukeren bli navigert til neste modul, oppgavesiden.



Figur 4-3: Figma-design av oppgavesiden

Figur 4-3 er en skisse av oppgavesiden og viser hvordan brukeren velger programmeringsspråk og versjon i nedtrykksmenyen over. Brukeren kan videre skrive kode i kodeboksen (Code) basert på oppgavebeskrivelsen (Description). Videre informasjon om oppgavens enhetstester er presentert i testboksen (Tests). Når brukeren er klar for å teste koden trykker brukeren på kjør (Run) knappen. Brukeren kan til slutt velge å lagre (Save) resultatet.

Etter skissering og modellering er det dannet et utgangspunkt for utvikling av testplattformen.

4.2 Utvikling

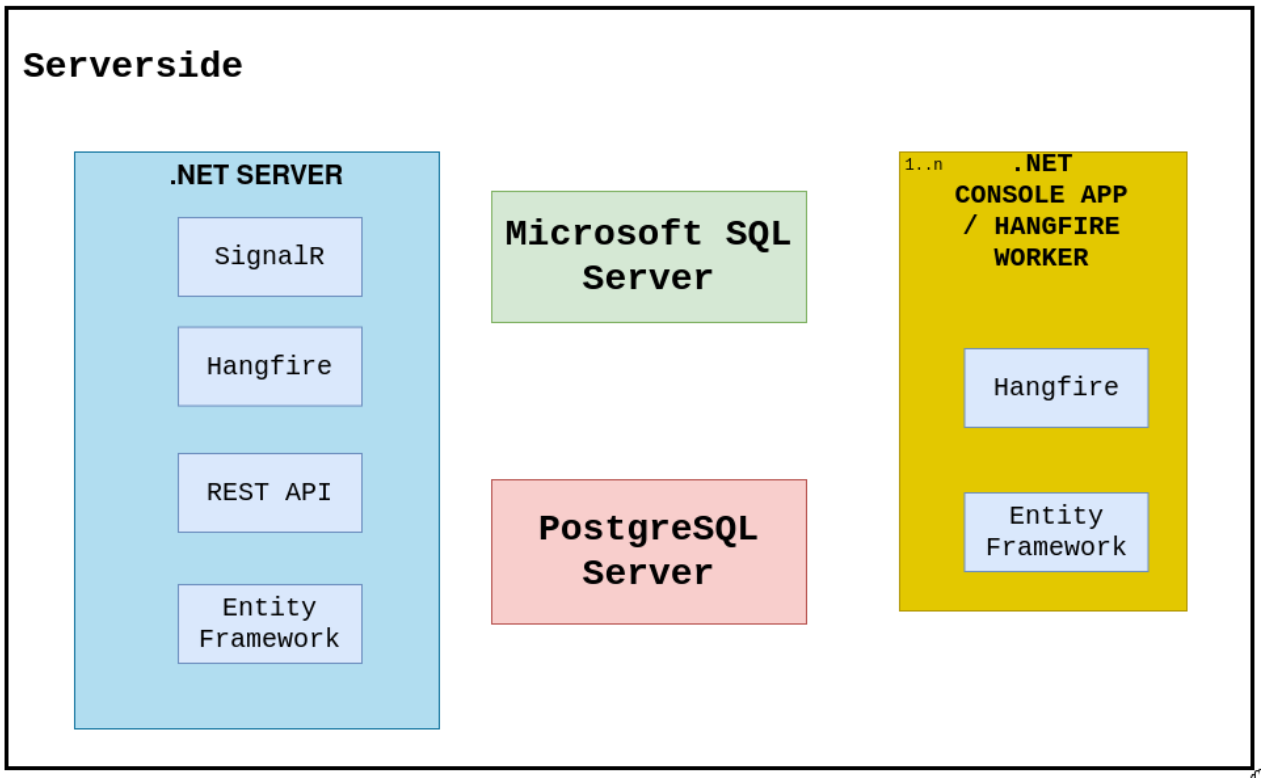
Etter modellering av systemet og design av nettsiden, legges det frem i detaljer hvilke teknologier prosjektgruppen benyttet i utviklingsfasen.

React og .NET er hovedrammeverkene som prosjektgruppen brukte for å utvikle testplattformen. Kombinasjonen av disse teknologiene utgjør en Single Page Application (SPA) arkitektur. SPA arkitekturen gir klienten nødvendige modeller av hele applikasjonen og bruker en serversides REST API til å hente, oppdatere og begrense informasjon. Fordelen med SPA er en brukeropplevelse med

reaktive elementer på brukergrensesnittet, og navigering som ikke krever flere innlastinger av nettsiden (BasuMallick, 2022).

4.2.1 Serverside

Serversiden av testplattformen er basert på én .NET-server, to relasjonsdatabaser og flere .NET konsoll applikasjoner. Serversiden sin funksjon i SPA arkitekturen er å fylle klienten opp med brukerrelevant innhold. For å oppnå dette benyttes det et REST API som klientsiden kan hente nødvendig informasjon fra. Serversiden bruker også en TCP-kommunikasjonskanal til klienter gjennom en Websocket.

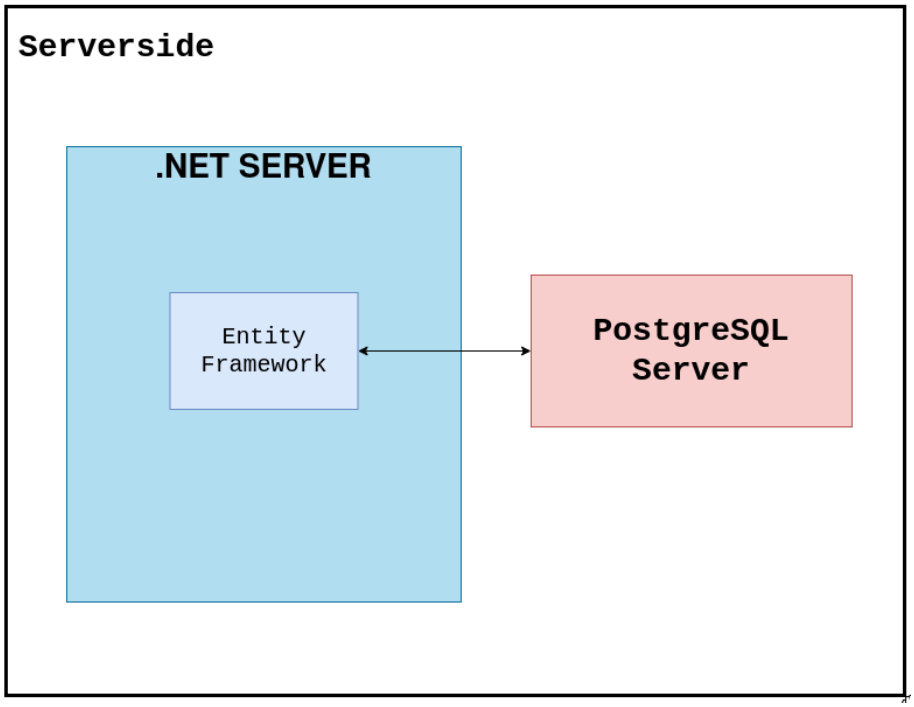


Figur 4-4: Oversikt over serversiden

Figur 4-4 er en grafisk oversikt over serversiden med forskjellige servere og deres underliggende rammeverk. Videre i delkapittelet skal komponentene i figuren redegjøres før prosjektnytten fremlegges. Relasjonene mellom komponentene vil bli modellert trinnvis i delkapittelet, slik at leseren av rapporten oppnår en forståelse for den overordnet sammenhengen på serversiden.

Database

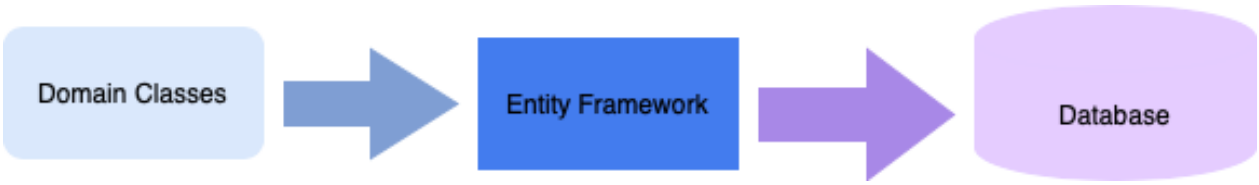
For å implementere databasen er det brukt en kode-først (Code-first) tilnærming som kan implementeres gjennom rammeverket Entity Framework (EF). EF er et Microsoft utviklet rammeverk for .NET som fungerer som et bindeledd mellom database og kodeimplementasjon. Rammeverket muliggjør alle typer databaseoperasjoner fra .NET siden, samtidig som Microsoft har dokumentert rammeverket svært godt (Microsoft, 2021).



Figur 4-5: Oppbygging av serversiden med EF

Figur 4-5 fremstiller serversiden etter implementasjon av EF på en .NET-server, og tilkobling mot en PostgreSQL database server. Figuren er den første iterasjonen av oppbygging mot Figur 4-4.

Kode-først er en tilnærming der databaseobjekter og relasjoner formes av klassemodeller (Domain Classes) som skrives i det aktuelle rammeverket. Det tillater utviklere å endre programvare uten å tilpasse databasen direkte gjennom SQL-setninger (EntityFrameworkTutorial, u.å.). Tilnærmingen lagrer iterative endringer av databasen som et migrasjonsobjekt, som blir brukt for å oppdatere databasen. Disse migrationsobjektene tillater databasen å gå tilbake i tid, noe som er nyttig for prosjektets iterative utvikling, Lean startup. Denne fordelen fikk prosjektgruppen utslag for da det første utkastet av databasediagrammene møtte utfordringer under implementering og måtte justeres.

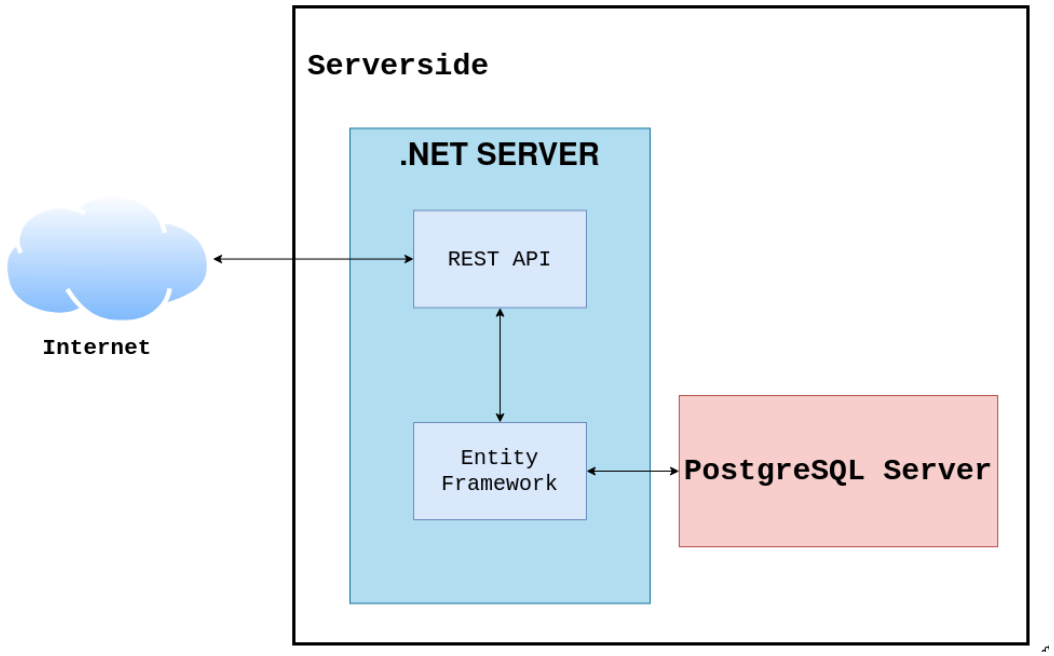


Figur 4-6: Kode først tilnærming i EF (EntityFrameWorkTutorial.net, u.å.)

Figur 4-6 er en illustrasjon av kode først med EF. Klassemodeller blir omgjort gjennom EF til en ferdig relasjonsdatabase. EF er fleksibelt mot alternative relasjonsdatabaser, samtidig som det distanserer behovet for direkte utvikling i databasen (Microsoft, 2021). Prosjektgruppen har derfor opplevd valget av PostgreSQL som lite innvirkende for prosjektet. Oppsummert har EF muliggjort prosjektgruppen å skifte fokus fra databaseimplementering og vedlikehold til utvikling av andre komponenter på serversiden.

REST API

REST API er et populært verktøy for å kommunisere med forskjellige klienter. Med dette verktøyet kan applikasjoner utveksle informasjon over HTTP-protokollen. REST API er designet med hensikt å frakoble (decouple) server implementasjon fra klientimplementasjon. Det tillater UI rammeverk som React å bli utviklet uavhengig av serverimplementasjon og tillater større endringer i serverarkitekturen uten å påvirke motsatt ende. (IBM, u.å.)



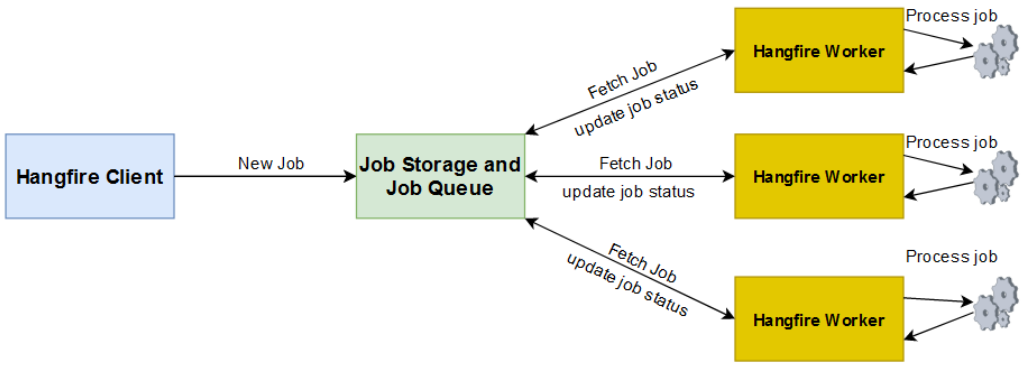
Figur 4-7: Oppbygging av serversiden med REST API og EF

Figur 4-7 bygger videre på figur 4-4 som illustrerer hvordan REST APIet kobler en serverside med EF og PostgreSQL database mot internett. .NET tilbyr implementasjon av API-endepunkter ved bruk av Kontroller objekter (Controllers) og minimal konfigurasjon. Prosjektgruppen har raskt implementert over 30 slike HTTP-endepunkter i 8 Kontroller objekter fra figur 4-1. Bakgrunnen for den raske implementasjonen er god Microsoft dokumentasjon, som har tilført prosjektgruppen mulighet å fokusere på videre implementering av testplattformen. Dette bidro til innhentet utviklingstid

Hangfire

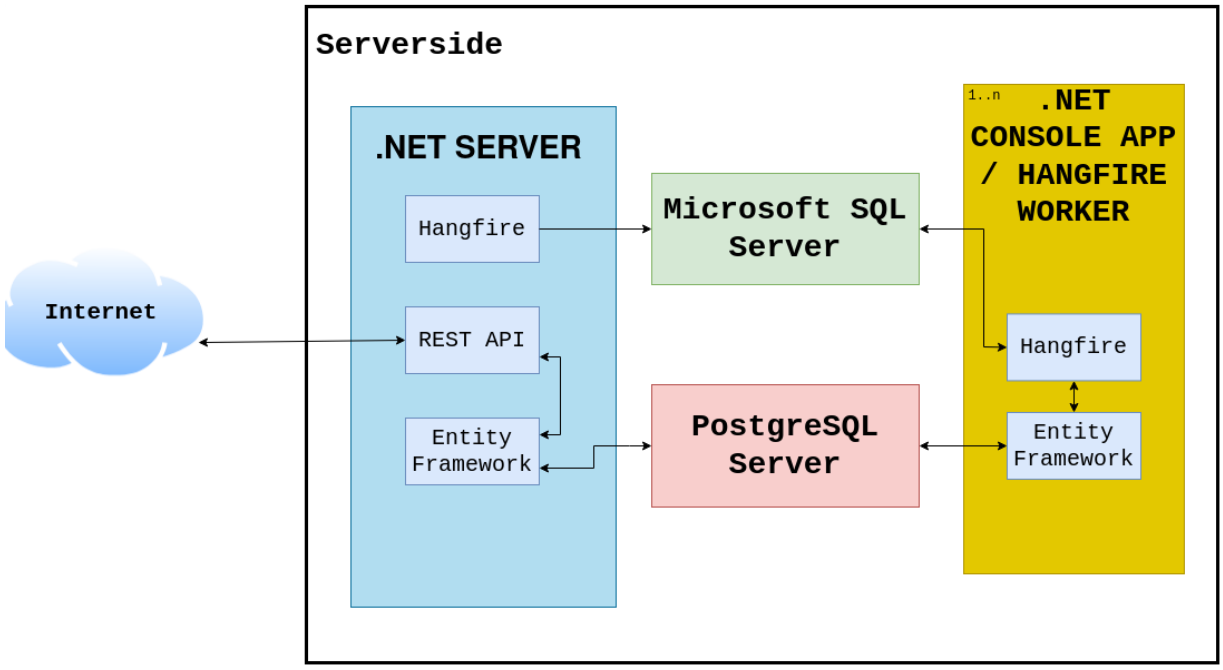
Det ble fremstilt litteratur i punkt 2.4.2 hvordan utførelse av klientkode er en operasjon som gjøres med mistillit, ettersom utførende part ikke vet ressursbehovene til koden. Det ble derfor identifisert behov for å begrense ressurstilgangene til brukerkoden klienter ønsker å utføre på testplattformen. For å oppnå det er Hangfire rammeverket et alternativ som kan allokere en mengde ressurser til en prosess.

Hangfire er et rammeverk for bakgrunnsarbeid i .NET applikasjoner. Rammeverket støtter flere type jobbhåndteringer slik som utførelse umiddelbart, på tidspunkt og etter en mengde tid. Hangfire består av klienter, en database og arbeidende enheter (docs.hangfire, u.å.).



Figur 4-8: Hangfire arkitekturer

Figur 4-8 fremstiller Hangfire sin arkitektur. En Hangfire klient (Hangfire Client) vil være på serversiden og genererer jobber før den setter de på køen i databasen (Job Storage/Queue). Hangfire vil aldri duplisere arbeid, det tillater et mangfold av arbeidende enheter (Hangfire Worker) som utfører arbeid på køen. De arbeidende enhetene kan konfigureres slik at de bare sitter på en fast mengde datamaskin ressurser som ikke kan overskrides. (Hangfire, u.å.). Hangfire fungerer som en slags vektbalanserer til det ressurskrevende arbeidet, som er å utføre kode på serveren. Det gjør serversiden bedre egnet til å skalere horisontalt til flere klienter da antall arbeidende enheter kan økes til flere servere for å møte et større kundebehov. Teorien rundt dette fremstilles i punkt 2.4.2, relevant litteratur.



Figur 4-9: Oppbygging av serversiden med EF, REST API og Hangfire

Figur 4-9 bygger på figur 4-5 og illustrerer serversiden etter implementasjon av Hangfire. Hangfire gir ikke støtte til PostgreSQL som mellomdatabase, så prosjektgruppen valgte en Microsoft SQL Server, som er den standardiserte databasen Hangfire benytter (Hangfire, u.å.). De arbeidende enhetene ligger på .NET Konsoll servere, mens Hangfire klienten er på en .NET-server. Konsoll

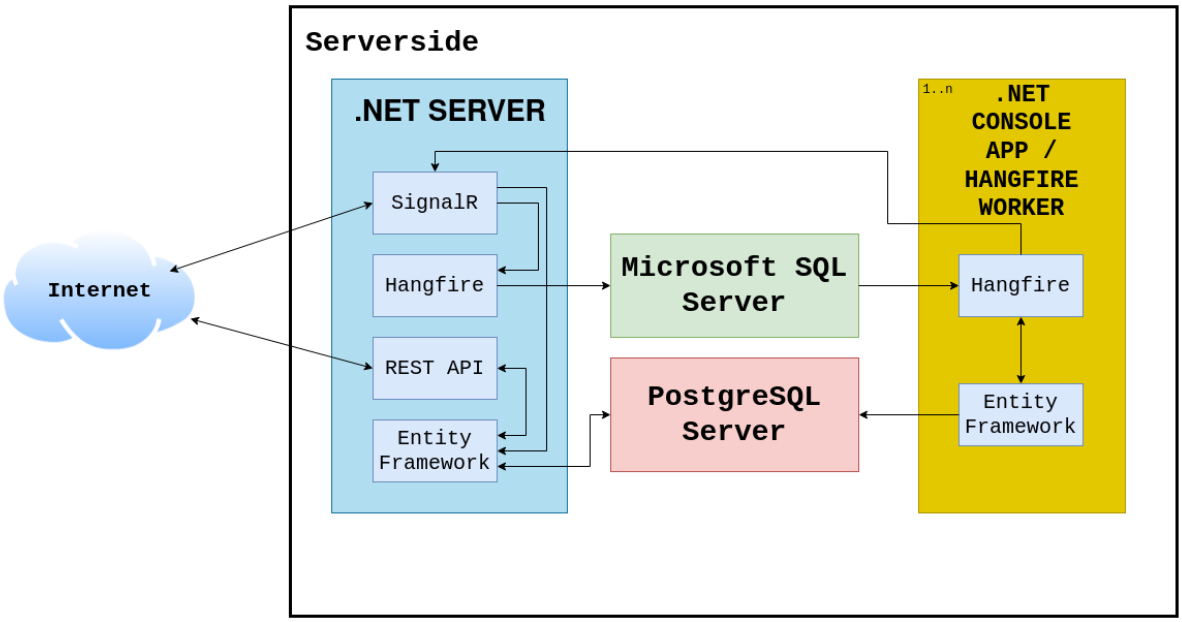
serveren er hvor koden blir utført og lagrer resultatet gjennom Entity Framework i PostgreSQL databasen.

Det er ingen direkte kobling fra internett til Hangfire i figur 4-9. Det er mulighet for å åpne en relasjon mellom REST APIet og Hangfire, så klienter kan benytte HTTP-protokollen til å aktivere den. For at klienter skal kunne hente ressurser over denne protokollen kan det benyttes to tilnærminger, kort polling (Short polling) og lang polling (Long polling). Kort polling er en konstant spørring mot APIet til informasjonen er klar. Dette fører til ekstra trafikk på APIet og skalerer derfor ikke med mange brukere. Alternativt kan lang polling benyttes og fungerer som et synkront kall, der klienten sender en forespørsel og mottar resultatet i samme kall. Ettersom hangfire vil bruke tid for å utregne resultatet vil prosesser bli ventende på REST APIet, som sluker ressurser (Fette & Melnikov, 2011). For å unngå lang og kort pollling skal det i neste avsnitt introduseres en ny kommunikasjonsprotokoll til serversiden.

SignalR

SignalR er et kommunikasjonsrammeverk i .NET der klienter kan asynkront kalle på metoder i mellomvaren på serveren. Rammeverket bygger på Websockets, en teknologi som tillater toveis kommunikasjon mellom klient og server. Det fører til en server som kan oppdatere klienter direkte, noe som blant annet er brukt i direktemeldingstjenester (Microsoft, 2023).

Prosjektgruppen valgte SignalR for å tilby Hangfire en direkte kommunikasjonslink til klienter. Det tillater Hangfire å varsle klienten til en Websocket når koden er ferdigtestet og resultatet er utregnet.



Figur 4-10: Fullstendig overblikk av serversiden med tilhørende relasjoner

Figur 4-10 viser hele serverside arkitekturen med tilhørende relasjoner. Den illustrerer hvordan Hangfire har en kobling til internett gjennom SignalR. Relasjonen mellom internett og SignalR er en mellomvaremetode som tar imot brukerkode som skal testes. SignalR lagrer forsøket - som er

definert i figur 4-1 - i databasen før den videre varsler Hangfire. Relasjonen mellom Hangfire sin arbeidende enhet og SignalR er en metode som varsler klienten om resultatet.

Etter fremstilling av arkitekturen rundt Hangfire sin arbeidsenhet, skal innholdet arbeidsenheten utfører i det prosjektgruppen kaller utførelsesmiljø forklares i nærmere detalj.

Utførelsesmiljø

Utførelsesmiljøet er selve kjernen av testplattformen der kode kompileres og enhetstestes før et resultat blir bestemt. Dette miljøet er utvidbart til flere programmeringsspråk, noe som er mulig ettersom utførelsesmiljøet benytter kommandolinjeverktøyet CLIwrap. Det er et bibliotek som kan importeres til .NET og brukes til å utføre en nødvendig prosedyre. Det gjør det mulig å utvikle nye miljøer for programmeringsspråk ved å laste ned tilhørende avhengigheter direkte på serverens operativsystem.

De fleste programmeringsspråk har sin egen prosedyre for å kompilere og enhetsteste kode. Prosedyren avhenger av egenskapene til programmeringsspråket, da ulike språk bruker forskjellige programmer til å eventuelt kompilere og kjøre koden. Programmeringsspråk har også forskjellige teknikker for å enhetsteste kode, som er noe prosedyren må tilpasse. Prosjektgruppen utviklet bare en JS prosedyre til testplattformen.

En lur bruker av testplattformen kan forfalske et godt resultat ved å hardkode enhetstestene som blir presentert på klientsiden. For å forhindre hardkodete tester å gi falske positiver er det utviklet en ekstra enhetstest som brukeren ikke har innsyn til, som blir kalt for «Hemmelig enhetstest» (Secret Unittest).

Tidlig i prosjektet ble det identifisert problematikk rundt kodeinjeksjon som et sikkerhetshull i løsningen. Et eventuelt tiltak er å kjøre koden på et virtuelt sandkassemiljø (sandbox enviorment) som beskrevet i Goltzsche sin forskningsartikkel (2019). Dette arbeidet ble ikke ferdigstilt og tilstrekkelige tiltak er ikke gjort for å sikre testplattformen.

4.2.2 Klientside

Klientsiden har som oppgave å gi brukeren et grafisk grensesnitt mot den logiske serversiden. Rollen klientsiden har i SPA arkitekturen er å gi brukeren grafiske elementer, struktur og tilhørende JS til å kommunisere med serversiden for informasjon.

For å utvikle reaktive komponenter og brukergrensesnitt er React benyttet, som er begrunnet i punkt 2.4. React er et JS rammeverk for å bygge brukergrensesnitt som er utviklet av Facebook og blir vedlikeholdt av Meta (Arancio, 2021; Thomas, 2021).

Brukergrensesnitt

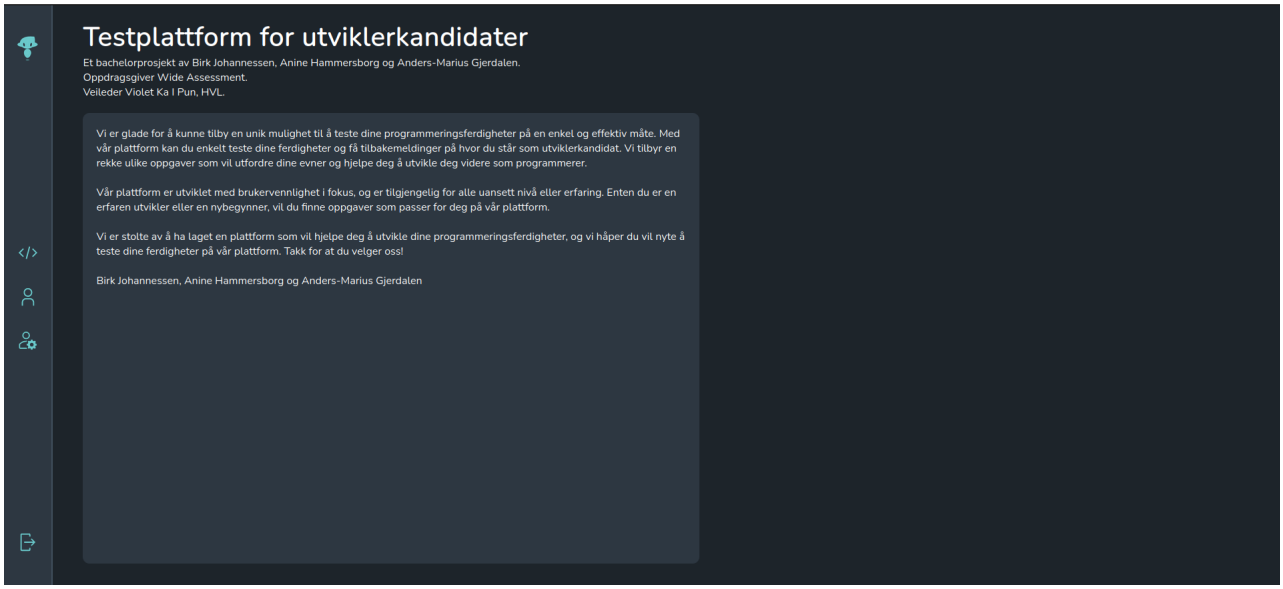
Ettersom React er et veldig populært rammeverk for å utvikle brukergrensesnitt, har flere utviklere publisert åpen kildekode (open source) av egne grensesnitt som andre kan gjenbruke. Prosjektgruppen tok i bruk kun et slikt prosjekt i testplattformen, et koderedigeringsgrensenitt fra Microsoft som heter Monaco. Det er basert på Microsofts Visual Studio Code (VScode), som er det

mest populære koderedigeringsverktøyet ifølge Stackoverflow sin utviklerundersøkelse fra 2021 (Stackoverflow, 2021). Monaco tilbyr en lik opplevelse utviklere får i VScode, som innebærer syntaksutheving (Syntax highlighting) og IntelliSense. Dette verktøyet ble identifisert som svært nyttig for utvikling av testplattformen ettersom det er tidskrevende å utvikle eget koderedigeringsverktøy, samtidig som Monaco støtter et bredt utvalg av programmeringsspråk.

Prosjektgruppen har ellers utviklet komponenter til brukergrensesnitt fra bunn med en kombinasjon av React og CSS. Å utvikle grensesnitt fra bunn har vært tidskrevende, men det har tillatt prosjektgruppen å følge designet fra figur 4-2 og 4-3 i nærmere detaljer enn hva medlemmene ellers kunne oppnådd.

Beskrivelse av brukergrensesnitt per modul

Videre skal modulene som prosjektgruppen har utviklet beskrives i nærmere detalj.



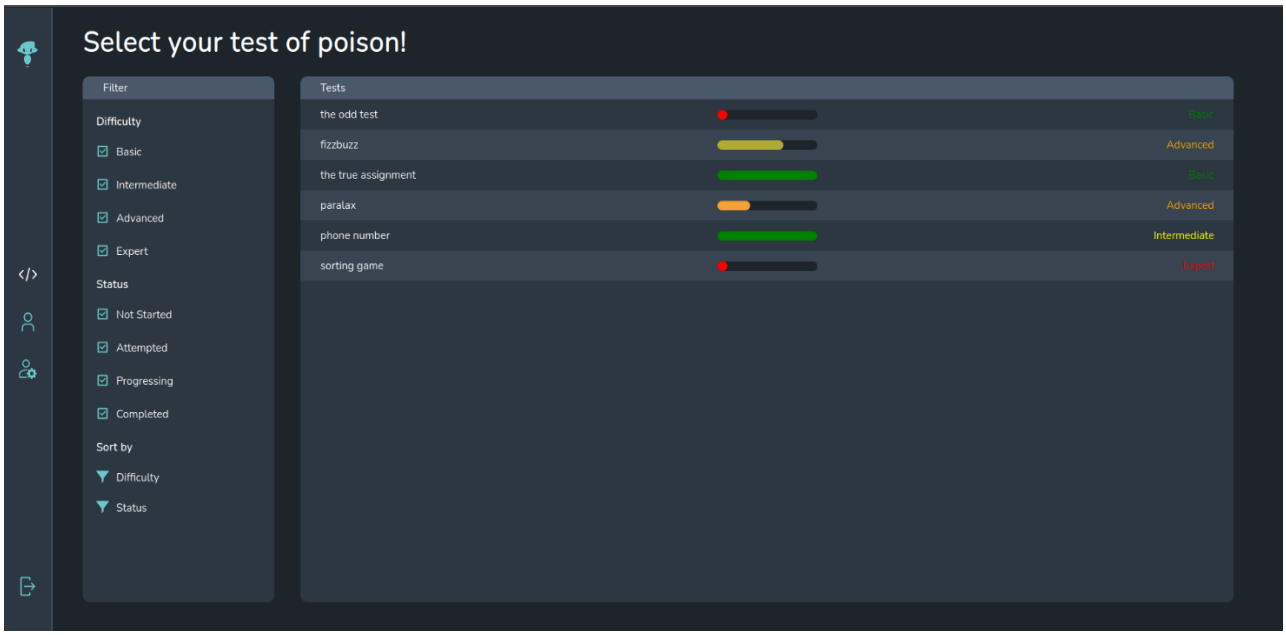
Figur 4-11: Modul for landingsside

Landingssiden vises i figur 4-11 og er brukerens første opplevelse av testplattformen, med hensikt å fortelle brukeren om innholdet på nettsiden. Teksten i modulen er skrevet av en tekstgenererende AI, ChatGPT.

Navigering gjøres i venstre navigeringsmodul, som har logoen til prosjekteier øverst, tre navigeringsknapper videre inn i testplattformen og nederst en knapp for å logge ut.

Brukstilfellene for navigeringsmenyen:

- Prosjekteierlogoen (WA) navigerer tilbake til WA.works sin nettside.
- Øverste midtmenyknapp navigerer til neste modul i rapporten.
- Midtre midtmenyknapp navigerer til brukeroversikt. Denne modulen er ikke utviklet.
- Nederste midtmenyknapp navigerer til en administratorside (adminside). Denne modulen er brukt under utvikling og skal ikke være synlig eller tilgjengelig for vanlige brukere.
- Nederste knapp er en avloggings knapp.



Figur 4-12: Modul for valg av tester

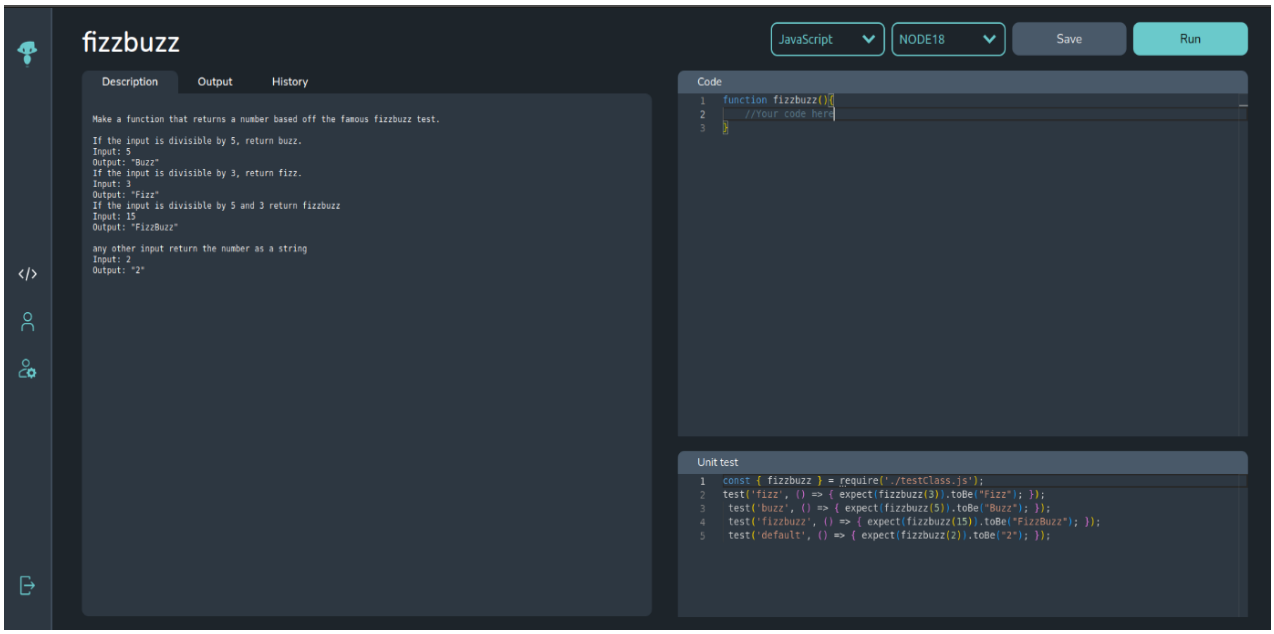
Figur 4-12 viser et utvalg av tester en bruker kan utføre. Hver tilgjengelig test har et navn, en statusbar for brukerens progresjon og en vanskelighetsgrad. Vanskelighetsgraden spenner fra «Basic» til «Intermediate», «Advanced» og «Expert», som gir brukeren mulighet å velge en test som passer egne ferdigheter. Statusbaren spenner fra «Not started» til «Attempted», «Progressing» og «Completed». Den er animert lastende (Loading) fra venstre mot høyre når brukeren laster inn modulen.

Tilfellene for statusbaren:

- Ikke startet (Not started). Rød, brukeren har ikke sendt inn testkode.
- Forsøkt (Attempted). Oransje, brukeren har sendt inn en eller flere kodesnutter
- Utvikler (Progressing). Gul, brukeren har klart noen av enhetstestene i testen.
- Fullført (Completed). Grønn, brukeren har klart alle enhetstestene i testen.

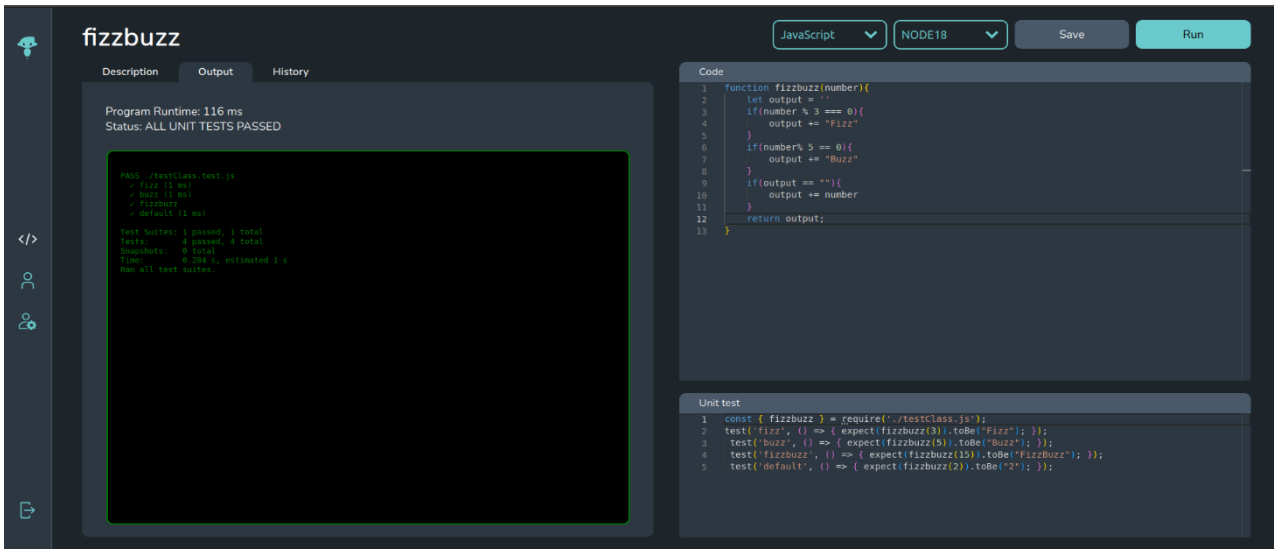
Videre er det et filter for testene slik at brukeren kan begrense mengden tester basert på sin preferanse. Filteret har også synkende sorteringsknapper for status og vanskelighetsgrad.

Når brukeren bestemmer seg for en test, trykker brukeren på testen og blir videre navigert til prosjektets siste modul.



Figur 4-13: En modul der brukeren løser valgt test

Figur 4-13 viser modulen der brukeren løser sin valgte oppgave. Brukeren blir presentert med en kodesnutt som definerer metoden til oppgaven i Monaco-grensesnittet til høyre i figuren. Detaljer om oppgaven ligger forskjellige steder i modulen. Testnavnet er øverst som overskrift, beskrivelsen (Description) til venstre og enhetstester (Unit tests) nederst til høyre. Når brukeren er klar for å kjøre koden trykker brukeren på Kjør (Run) knappen som varsler serversiden over en Websocket.



Figur 4-14: Fremviser en test som er løst riktig

Figur 4-14 fremviser klienten som har blitt varslet om resultatet av kodekjøringen. Resultatet foreligger på venstre side av oppgavesiden i «Output» fanen. Brukeren kan bytte mellom disse fanene etter behov. Det aktuelle resultatet beskriver til brukeren hvordan alle enhetstestene er riktige, som betyr at koden er riktig implementert for oppgaven.

Etter å ha lest kapittelet skal leseren forstå løsningen prosjektgruppen har utviklet og de tekniske detaljene for å oppnå resultatet.

5 RESULTATER

Dette kapittelet vil beskrive i detalj hvordan resultatet fra utviklingen ble evaluert. Det blir også presentert ulike metoder og framgangsmåter som ble benyttet i evalueringen.

5.1 Evalueringsmetode

Videre i delkapittelet vil det utforskes grundig hvordan prosjektgruppen gikk fram for å utføre tester, hvordan brukere ble involvert i evalueringen, og hvordan resultatene ble presentert.

5.1.1 Evalueringsmetode for brukere

For å evaluere brukervennligheten og funksjonaliteten til testplattformen ble det gjennomført en brukertest med et utvalg av brukere. For å sikre at testplattformen traff et større mangfold, hadde deltakerne ulik erfaring og ferdighetsnivåer innenfor programmeringsfaget.

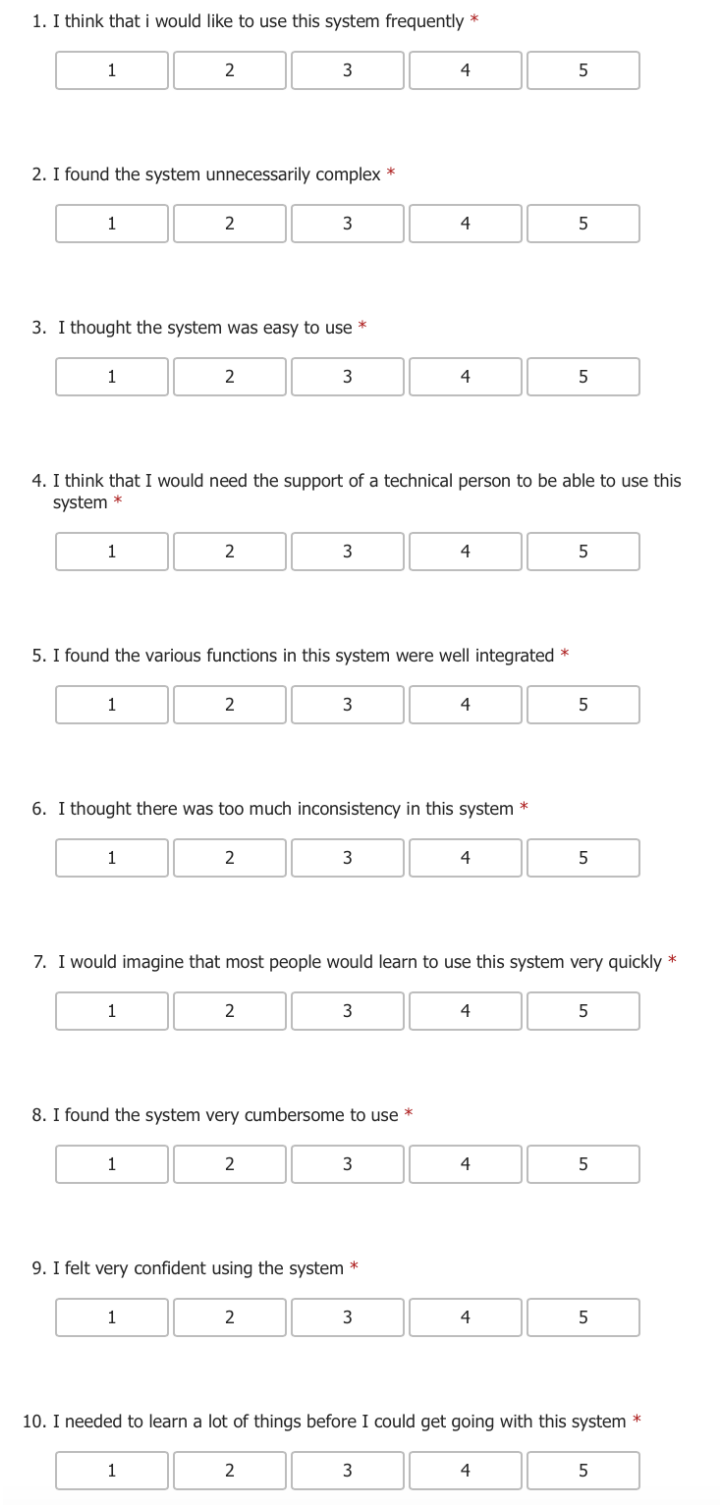
Under brukertestene fikk brukerne en personlig introduksjon til prosjektet og ble bedt om å utføre en praktisk oppgave på testplattformen. Brukerne ble oppfordret til å navigere nettsiden for å velge en oppgave etter ønsket ferdighetsnivå, utføre oppgaven og levere resultatet. Etter gjennomført test mottok hver bruker et evalueringsskjema basert på SUS-standarden. Det ble inkludert tilleggsspørsmål der brukere fikk reflektere i form av fritekst. Tilleggsspørsmålene ble inkludert med den hensikt å oppnå en dypere forståelse av brukerens perspektiv.

Fritekstspørsmålene var følgende:

1. «Hadde du en dårlig opplevelse? Fortell hvorfor»
2. «Var det vanskelig å navigere seg rundt på nettsiden? Fortell hva du ville endret»
3. «Var noe uklart med nettsiden? Fortell hva du synes mangler»
4. «Vi hadde satt pris på din tilbakemelding om plattformen»

SUS – System Usability Scale

Spørreskjemaet som er benyttet heter System Usability Scale og er en bredt anvendt metode for å måle brukervennlighet (Brooke, 1995). Det er også en standardisert metode som gjør det enkelt å sammenligne resultater på tvers av ulike studier eller produkter (Sauro, 2011).



Figur 5-1: SUS skjemaet

I figur 5-1 er SUS spørsmålene fremstilt, og visualiserer et kort spørreskjema som består av 10 enkle påstander om produktet som skal evalueres. Disse påstandene tar for seg aspektene brukervennlighet, effektivitet og tilfredshet. Hver påstand besvares på en 5-punkts likert-skala som går fra «helt uenig» til «helt enig» og blir regnet om til en poengsum (Score). SUS-Poengsummen (SUS-Score) beregnes til å bli et tall mellom 0 og 100, hvor 100 indikerer en høy poengsum knyttet til de 3 nevnte aspektene. Svarene til hver bruker legges sammen for å danne en individuell

poengsum, hvor partalls- og oddetalls -spørsmål har egne regler for utregning. Videre summeres svarene sammen og ganges med en konstant for å skalere fra 0 til 100. Fremgangsmåten for utregningen av poengsummen er vist i vedlegg 5.

5.1.2 Evalueringsmetode for prosjekteier

Prosjekteier sin evaluering av testplattformens funksjonalitet og brukervennlighet var en av de sentrale metodene for evalueringen. Før en funksjonell testplattform var på plass ble det benyttet en MVP i form av en visuell fremstilling av løsningen som vist i figur 4-2 og 4-3. Det ble videre diskutert i hvert møte hva prosjektgruppen skulle gjøre for å imøtekomme krav og ønsker fra prosjekteier. Deretter ble det avtalt et møte med oppdragsgiver for å evaluere testplattformen når funksjonaliteten var på plass. Før de fikk evaluere testplattformen ble det gitt en overordnet presentasjon. To representanter hos prosjekteier fikk muligheten til å gjennomgå prosessen med å velge en oppgave og deretter utføre den. Evalueringen inkluderte en gjennomgang av testplattformens funksjonalitet og brukervennlighet, med vekt på å identifisere eventuelle feil, mangler og forbedringsområder. I tillegg ble prosjekteiers krav og behov vurdert for å sikre at testplattformen oppfylte krav og forventinger til en pålitelig testplattform.

Etter fremgangsmåten var etablert, skal resultatet av evalueringen fremstilles.

5.2 Evalueringsresultat

I dette delkapittelet formidles resultatene fra de ulike evalueringsmetodene som ble brukt for å undersøke testplattformens brukervennlighet, og dens potensiale for å være effektiv i en rekrutteringsprosess. Evalueringen i vedlegg 6 og 7 inkluderer resultat fra brukertestene og prosjekteier.

Tilbakemeldinger fra brukerne ble også samlet inn etter evalueringen for å identifisere eventuelle svakheter og forbedringsområder for testplattformen. Prosjektgruppen hadde en funksjonell evaluerbar testplattform sent i prosjektperioden, og rakk derfor kun en iterasjon med brukertester.

5.2.1 Evalueringsresultat for brukere

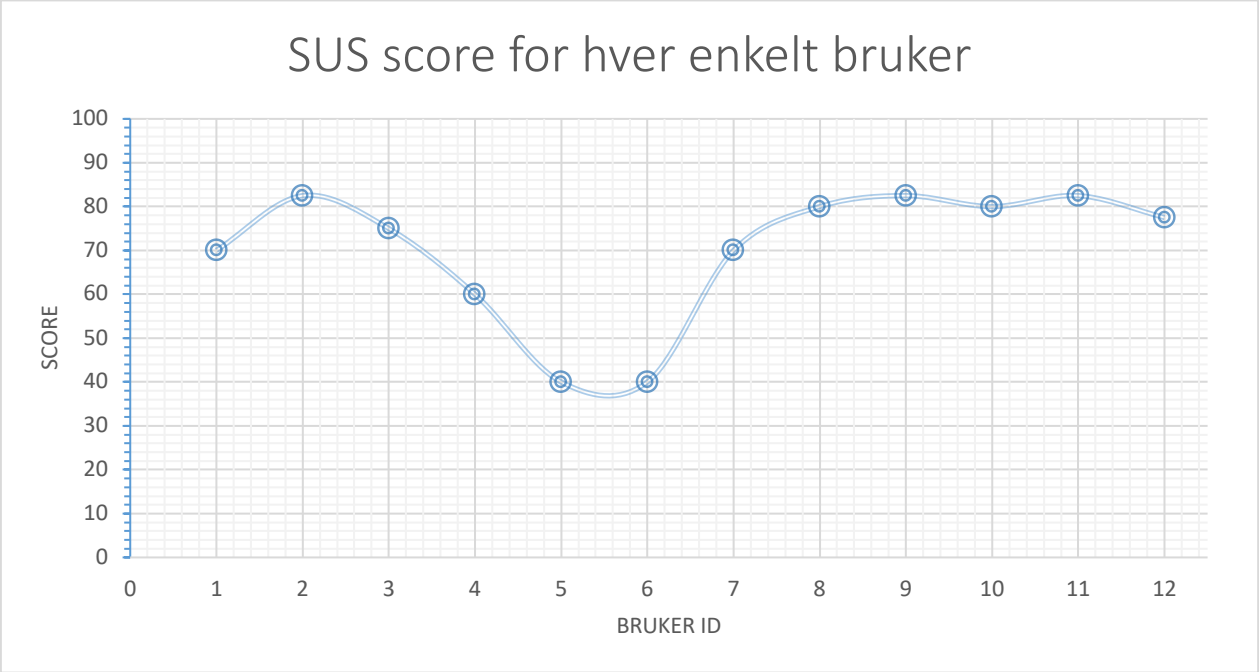
Brukerne som ble testet hadde forskjellige erfaring innenfor programmeringsfaget. I evalueringssamtalen etter testingen hadde brukerne et godt inntrykk av testplattformen. De mente at funksjonalitetene var til stedet og designet var stilfullt.

I svarene fra fritekst evalueringen i vedlegg 8, kom disse forbedringsforslagene frem:

- Gjør siden mer intuitiv for navigasjon mellom de ulike sidene.
- Ikonene burde hatt en pop-up type, for å gjøre det tydeligere hva ikonene gjorde.
- Det var et ønske om en egen tilbakeknapp til oversiktsmodulen fra oppgavemodulen.
- Navigasjonsbaren på venstresiden skulle vært mer representativ, i den forstand at en kunne forstå hva ikonene betydde.

- En bedre måte å komme seg til neste oppgave.

Tilbakemeldingene handlet generelt om brukervennlighet, som er noe prosjektgruppen verdsetter høyt. Punktene ble derfor prioritert til videre arbeid.



Figur 5-2: SUS score for hver enkelt bruker

Det er i figur 5-2 en fremstilling av resultatet fra brukerens svar på SUS skjemaet. Figuren viser en poengsum for hver bruker basert på de 10 spørsmålene, som gjenspeiler brukeropplevelsen. Ut ifra figuren kan det tolkes at majoriteten av brukerne hadde en høy tilfredshet. Det ble identifisert at 2 av brukerne var mindre tilfreds med brukeropplevelsen. Både positive og negative tilbakemeldinger har blitt analysert, og vil bli tatt hensyn til i videre evaluering. Totalt ga den samlede poengsummen til brukerne et gjennomsnitt på 70, som tilsvarte en B på karakterskalaen, vist i vedlegg 9.

Avslutningsvis ble brukertesten gjennomført med suksess og resultatene kan brukes til å optimalisere brukeropplevelsen til testplattformen i videre arbeid. Denne evalueringen viser virkeligheten av å involvere brukere i utviklingen av digitale produkter for å sikre tilgjengelighet og brukervennlighet. Totalt sett viste brukerevalueringene seg å være en viktig evalueringsmetode for å sikre at testplattformen oppfylte kravene til prosjekteier.

5.2.2 Evalueringsresultat for prosjekteier

Prosjekteier evaluerte om prosjektgruppen hadde oppfylt kravene som ble gitt ved starten av prosjektperioden, og resultatet var svært positivt. Prosjekteier var imponert over sluttproduktet og uttrykte at prosjektgruppen hadde løst oppgaven på en god måte. De bemerket spesielt at prosjektgruppen hadde tatt hensyn til aspekter som UX, ren kode og datasikkerhet, selv om det ikke var en del av kravene for prosjektet. Prosjekteier påpekte også at prosjektgruppen hadde utført oppgaven i en større skala og med flere teknologier enn forventet.

Videre bekreftet prosjekteier at alle kravene for prosjektet var oppfylt. Derimot var det forbedringspotensialer knyttet til UX i React:

- Nedtrykksmeny hadde ikke noe musepeker (cursor).
- Skalering på forskjellige skjermstørrelser og mobil.
- Symboler i navigeringsbaren har ikke beskrivelse når musepekeren er over (hover).

Til slutt ble det diskutert om produktet var tilstrekkelig nok implementert. Prosjekteier informerte at produktet kun trengte små endringer før plattformen kunne integreres mot deres allerede eksisterende system.

Etter å ha evaluert resultatene, vil de bli sett i sammenheng med prosjektets hensikt.

5.3 Prosjektresultat

Arbeidet mot testplattformen i et systemperspektiv har vært vellykket, med alle fokusområder implementert på en tilfredsstillende måte. Prosjektgruppen har utviklet en testplattform som kan regne ut et resultat for JS kode og valgt arkitektur som legger til rette for utvidelse av flere. Resultatene som blir regnet ut kan sammenlignes med andre til den grad av hvor mange enhetstester som passerer.

Prosjektgruppen har gjort arkitekturvalg som lar testplattformen skalere med en økende brukerbase, og effektivisert resultatkommunikasjonen over Websocket. Det var ønsket å evaluere skaleringsarbeidet, men prosjektgruppen manglet ressurser til å gjennomføre det i stor nok skala til å trekke en konklusjon. Derimot har prosjekteier sagt seg fornøyd med skaleringsmulighetene som vist i vedlegg 7. Som evalueringen viser, har prosjektgruppen utviklet en forståelig og brukervennlig nettside, men med noe forbedringspotensial for navigering. Videre viser vedlegg 7 hvordan prosjekteier har gitt inntrykk for å ha mottatt kode som direkte kan brukes til videre utvikling og integrasjon mot egne systemer.

Resultatene av prosjektet legger til rette for videre implementering av prosjekteier for å oppnå en testplattform som kan effektivisere en søknadsprosess.

5.4 Prosjektgjennomføring

Prosjektgruppen valgte Lean startup som arbeidsmetodikk, med Gantt-diagram som prosjektplan. Kombinasjonen av metodikkene har ført til en effektiv måte å sikre at prosjektet blir gjennomført oversiktlig og med en bestemt struktur. Det ga samtidig rom for fleksibilitet og tilpasning etter hvert som prosjektet utviklet seg.

5.4.1 Arbeidsmetodikk

Prosjektgruppen valgte å benytte seg av utviklingsmetodikken Lean startup. Som en del av denne metodikken ble det etablert kontinuerlig dialog med prosjekteieren for å gi oppdateringer om fremgangen i prosjektet. I forkant av hvert møte ble det utarbeidet en hypotese som ble diskutert i fellesskap. Det ble også presentert en MVP i form av en skisse som vist i figur 4-2 og 4-3 for å

visualisere prosjektets status. Dette bidro til å identifisere mulige utfordringer og forbedring av design. Prosjektgruppen brukte prosjekteierens tilbakemeldinger for å tilpasse prosjektet. I etterkant ble det opprettet en ny hypotese før prosjektgruppen fortsatte arbeidet i henhold til Lean startup-metodikken.

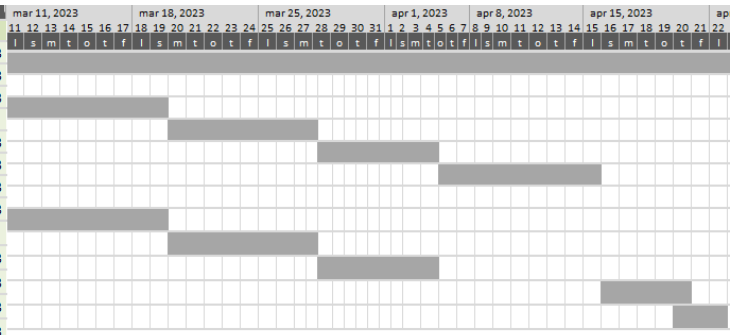
Denne arbeidsmetodikken sørget for en tilbakemeldingssyklus der brukerne kunne gi en rask tilbakemelding og øke kundetilfredsheten.

5.4.2 Prosjektplan

Gantt-diagrammet ble valgt ettersom det kunne gi prosjektgruppen en kontinuerlig oversikt over progresjonen til enhver individuell arbeidsoppgave. Diagrammet ble brukt til å dele opp i fire iterasjoner med arbeidsoppgaver. Hver iterasjon hadde et hovedområde knyttet til prosjektforløpet, der rekkefølgen var avgjørende. Det førte til en bestemt rekkefølge i planen, som ga en oversikt over hva som måtte fullføres før neste iterasjon kunne starte. Fordelen med Gantt-diagrammet er at prosjektgruppen kunne tildele timer fritt underveis i perioden. På den måten fikk gruppen en mer fleksibel arbeidssituasjon.

Arbeidet ble ikke alltid holdt innenfor den satte perioden, det førte til en forskyvning av de resterende arbeidsoppgavene og iterasjonene. Eksempelvis ble ikke testingen av MVP gjennomført til planlagt dato.

TASK	ASSIGNED	PROGR	START	END
Iterasjon 3 Rapport og implementering				
Jobbe med rapport	Alle	90 %	3.12.23	5.16.23
Backend: EntityFramework	Birk Johannessen	100 %	3.12.23	3.18.23
Backend: REST API	Birk Johannessen	100 %	3.19.23	3.27.23
Backend: Hangfire	Birk Johannessen	100 %	3.28.23	4.5.23
Backend: Utførelsesmiljø	Anders-Marius Gjerdalen	100 %	4.6.23	4.14.23
Backend: SignalR	Birk Johannessen	100 %	4.15.23	4.23.23
Frontend: Admin side	Birk Johannessen	100 %	3.12.23	3.18.23
Frontend: Oversikt over tester modul	Anine Hammersborg	100 %	3.19.23	3.27.23
Frontend: Oppgaveside modul	Anine Hammersborg	100 %	3.28.23	4.5.23
Frontend: Monaco Editor	Anders-Marius Gjerdalen	100 %	4.6.23	4.14.23
Frontend: Implementering mot API	Anine Hammersborg	100 %	4.24.23	4.28.23
Frontend: SignalR WebSocket	Anders-Marius Gjerdalen	100 %	4.28.23	4.30.23
Første test av en funksjonell MVP	Alle	100 %	4.28.23	4.30.23



Figur 5-3: Utklipp fra Gantt-diagram for illustrasjon av en sekvensiell inndeling

Derimot ble iterasjon 3, som vist i figur 5-3 utført innen planlagt tidsrom. Dette er et utklipp fra diagrammet som illustrerer perioden for iterasjonen er sekvensielt inndelt. Prosjektgruppen klarte å arbeide med flere arbeidsoppgaver parallelt, samtidig som avhengigheter ble tatt hensyn til. Før implementeringen av SignalR kunne utføres, måtte REST API, Hangfire og utførelsesmilljøet etableres. Innenfor samme tidsperiode foregikk utviklingen av frontend delen, for å fullføre betingelsen til avhengigheten fra SignalR. Dette resulterte i at prosjektgruppen kom i mål med testingen av sin første funksjonelle MVP.

Etter resultatene til prosjektet er presentert, skal det gjennomføres diskusjon om prosjektet.

6 DISKUSJON

I dette kapittelet vil det bli lagt frem diskusjon til prosjektgruppens tilnærming til prosjektet, hvilke konsekvenser tilnærmingen hadde og hvordan det påvirket resultatet.

6.1 Fremgangsmåte

Prosjektgruppen analyserte oppgavebeskrivelsen og brøt den ned i ulike delmål. Beskrevet med korte trekk var kravene at en kandidat skal kunne ta en kodetest, lagre resultatet og gi poengsum som kan sammenlignes med andre. Prosjekteier har et ønske om å gjøre prosessen om å gå fra kandidat til ansatt så enkel som mulig. Det ble dermed naturlig for prosjektgruppen å sette det som sitt mål og problemstillingen ble: «**Hvordan kan prosjektgruppen utvikle en testplattform som effektiviserer en ansettelsesprosess?»**

I startfasen av prosjektet var fokuset å finne den mest optimale løsningen og deretter utvikle en plan. Det ble vurdert å videreutvikle bachelorprosjektet om CSS-testen, men etter å ha analysert arkitekturen valgte prosjektgruppen en ny arkitektur med klient- og serverside. Derimot ble det hentet inspirasjon av designvalg fra prosjektet. Videre ble det valgt verktøy for prosjektet. Prosjektet valgte hovedsakelig verktøy som prosjekteier anbefalte, med et unntak av JS på klientsiden over TS. For å sikre en felles forståelse av arkitekturen, ble det utarbeidet diagrammer og en prosjektplan. Planen ble utformet i form av et Gantt-diagram, hvor prosjektperioden ble delt inn i ulike faser. Rekkefølgen av disse fasene ble bestemt etter avhengighetene mellom arbeidsoppgavene.

Det ble fremlagt to mulige prosjektmetodikker som kunne brukes i prosjektet, der Lean startup ble valgt. Det ble dermed naturlig å starte med design hvor en hypotese skulle utformes basert på prosjektkravene for å oppnå en MVP. De første iterasjonsrundene for evalueringen var med prosjekteier, hvor en hypotese ble presentert. Hvis hypotesen samsvarte med hva prosjekteier ønsket, begynte utviklingen av en MVP basert på den hypotesen. Den MVPen ble dermed brukt videre iterativt, helt til en funksjonell testplattform var ferdig utviklet.

Tid ble sett på som en begrensning. Prosjektgruppen var usikker på om prosjektperioden var lang nok til å nå målene for prosjektet. Tiden før utvikling ble brukt til å tilegne prosjektrelevante kunnskaper, i form av kurs og mindre prosjekter.

Fremgangsmåten for utviklingen var sekvensiell, der de viktigste serverfunksjonalitetene ble implementert først, før klientsiden ble utviklet. Serversiden ble utviklet trinnvis med database, EF og et REST API som en grunnmur i testplattformen. Klientsiden ble så utviklet i henhold til designet og sammenkoblet med serversiden. Videre serverfunksjonalitet som Hangfire, SignalR og utførelsesmiljøet ble implementert, med tilhørende Websocket på klientsiden.

Avslutningsvis i prosjektet evaluerte prosjektgruppen arbeidet som ble utviklet. SUS-skjema ble benyttet i brukerevalueringen, med separat evaluering mot prosjekteier.

6.2 Konsekvens av fremgangsmåte

Prosjektgruppen valgte å benytte en annen tilnærming enn den tidligere bacheloroppgaven. Dette medførte at prosjektgruppen måtte utforme en løsning fra bunn, fremfor å bygge videre på tidligere arbeid. En konsekvens av det valget var at prosjektgruppen ikke hadde en solid grunnmur å bygge videre på. Selv om dette medførte noen utfordringer, stod prosjektgruppen fritt til å finne sin egen tilnærming. Dette resulterte i en ny arkitekturløsning sammenlignet med det tidligere prosjektet. Konsekvensen av dette arkitekturvalget var muligheten for å utvide til flere programmeringsspråk, men åpnet trussel for kodeinjeksjon som diskutert i punkt 3.1.2 og et større behov for verktøy.

En konsekvens av verktøyvalget var mulighet for veiledning fra prosjekteier. Prosjektgruppen fikk utslag tidlig i utviklingen for JS valget ved å ikke bruke for mye tid å tillære seg TS. Som diskutert i punkt 3.3 legger verktøyvalgene til rette for videre arbeid av prosjekteier.

Metodikkvalget av Lean startup gjorde at prosjektgruppen ikke måtte forholde seg til en ukjent metode. Den iterative utviklingen viste seg å være en god nøkkelfaktor for prosjektgjennomførelsen. Det resulterte i gode modeller, som gjorde det enkelt å implementere på server- og klientside. Konsekvensen av dette er en testplattform som ble ferdigstilt med gode tilbakemeldinger fra start.

Valg av løsning påvirket mengden verktøy prosjektgruppen måtte tillære seg for å utvikle testplattformen. Prosjektgruppen oppdaget under utvikling at det hadde vært mer hensiktsmessig å ta kurs parallelt med utviklingen i stedet for å ta dem på forhånd. Konsekvensen av dette var mindre utviklingstid som påvirket den senere evalueringen.

Under utvikling ble det oppdaget områder som innhentet tid. Som diskutert i punkt 4.2.1 var .NET et verktøy som effektivt lot funksjonalitet implementeres, noe som var tidsbesparende for prosjektet. Til tross for denne tidsbesparingen, var det fortsatt tidskrevende å utvikle testplattformen frem til en funksjonell løsning. Konsekvensen av prosjektgruppens valg av løsning førte derfor til at en funksjonell evaluerbar testplattform ikke var klar før sent i prosjektperioden.

Selv om prosjektgruppen hadde kjennskap til prosjektmetodikken, ble tiden en faktor for hvor mange brukeriterasjoner som ble gjennomført. Dette resulterte i at SUS-skjemaet kun ble brukt i en runde med brukertester. Det påvirket planene om flere iterasjoner med tilbakemeldinger fra brukere og deres perspektiv. Derimot ble prosjekteier en sentral del av tilbakemeldingssyklusen. Det var lettere å forholde seg til prosjekteier, ettersom planleggingen var fleksibel. Prosjekteier kunne lettere gi tilbakemelding på et ufullstendig produkt, og var nyttig for å sikre at prosjektgruppen var i riktig retning.

6.3 Vurdering av resultat

Basert på tilbakemelding etter den siste evalueringen med oppdragsgiver, har prosjektgruppen levert et tilfredsstillende produkt. Denne tilbakemeldingen kom på tross av krav som ikke ble oppfylt, blant annet beregning av en brukers poengsum basert på kjøretid. Denne funksjonaliteten ble ikke implementert i systemet, men det ligger til rette for at den kan bli det på et senere tidspunkt. Det ble derimot tatt hensyn til skalerbarhet i forhold til antall brukere som kan benytte plattformen parallelt, se progresjon ved hjelp av en status bar og tester som kan gjennomføres.

Til tross for prosjekteiers gode tilbakemelding, kan ikke plattformene sammenlignes på funksjonalitet og brukergrensesnitt sett opp mot plattformer som Codewars og HackerRank. Dette er plattformer som er godt etablert og tilbyr et bredt spekter programmeringsspråk og tester. Codewars gir brukerne muligheten til å sammenligne seg med andre rundt om i verden gjennom sin rangeringsliste. På den andre siden tilbyr Hackerrank en omfattende plattform som hjelper brukere med å forberede seg til- og gjennomføre -digitale intervjuer med integrerte kodetester. Dette er funksjoner som prosjektgruppen kunne utført som videre arbeid på testplattformen. Selv om den per nå er mindre enn de mer etablerte plattformene som Codewars og HackerRank, er prosjektgruppen overbevist om at testplattformen vil ekspandere og utvikle seg i fremtiden. På sikt er målet at den skal bli en viktig del av rekrutteringen, ved å bidra til å øke effektiviteten i en ansettelsesprosess.

Prosjektgruppen ser tilbake på valgene av fremgangsmåten og systemarkitektur som gode valg. Lean startup la til rette for en fleksibilitet i timeplanen. Det medførte et mer fritt spillerom for gruppen å balansere rapport, utvikling og andre fag. Dette er metodikk prosjektgruppen er kjent med fra tidligere prosjekter, men det kan argumenteres for at resultatet kunne blitt bedre hvis prosjektgruppen hadde benyttet en annen arbeidsmetodikk. Det ble tidlig i prosjektet diskutert om Scrum skulle benyttes. Den kunne gitt prosjektgruppen en mer planlagt prosjektperiode, med flere delmål og frister som måtte overholdes. Det kunne potensielt ført til at prosjektgruppen fikk testet plattformen tidligere i prosjektperioden.

De teknologiske valgene av arkitektur har ført til en skalerbar og utvidbar testplattform. Verktøyene som er valgt har vist seg å være hensiktsmessig. Valgene av React og .NET som utgir SPA arkitekturen har ført til en god brukeropplevelse med reaktive elementer. Det er gjort valg som tar hensyn til skalering av testplattformen i form av å balansere ressurser på serversiden. Prosjektgruppen har analysert Codewars sin nettside og det er funnet informasjon som antyder bruk av lang polling, noe som er diskutert i punkt 4.2.1 som ressurskrevende. Det kan derfor argumenteres for at testplattformen for utviklerkandidater er bedre egnet i arkitekturen enn de allerede eksisterende løsningene.

Når man ser på dette fra et helhetlig systemperspektiv, kan man også vurdere hvordan denne teknologien kan påvirke samfunnet og økonomien på lang sikt. Prosjektgruppens testplattform har en mulig positiv effekt både når det kommer til samfunnsmessige og økonomiske hensikter. Hvis antallet intervjurunder minsker vil det potensielt være kostnadsbesparende for bedrifter, som fremlagt i kapittel 1 om motivasjon og litteratur i punkt 2.4.1.

Det er prosjektgruppen sitt synspunkt at en ferdigstilt testplattform kan benyttes for å effektivisere en ansettelsesprosess. Ferdigstilling av testplattformen i kombinasjon med WA.works sin godt etablerte rekrutteringsplattform og store brukerbase gir et utgangspunkt for å besvare problemstillingen. Årsaken for dette er basert på fremlagt vurdering av systemperspektiv som tar opp hvordan det kan være økonomisk- og tidsbesparende for en bedrift å gjennomgå en rekrutteringsprosess.

Hvis prosjektgruppen kunne gjøre arbeidet på nytt, ville det blitt vurdert å sette av mer tid til evaluering av testplattformen. Prosjektgruppen ville dermed hatt større mulighet til å oppdage flere mangler og feil på plattformen, som igjen kunne påvirke brukervennligheten på en positiv måte. Det er derfor viktig å erkjenne at selv om det er utviklet en funksjonell testplattform, vil det alltid være

rom for forbedringer og nye funksjoner som kan legges til i fremtiden. En testplattform for utviklere er et produkt som alltid vil kreve kontinuerlig videreutvikling, da den teknologiske utviklingen og brukernes behov stadig endres over tid.

7 KONKLUSJON OG VIDERE ARBEID

Avsluttende del av rapporten skal prosjektet konkluderes, og videre arbeid defineres.

7.1 Konklusjon

Prosjektgruppen skulle utvikle en testplattform som kunne effektivisere en ansettelsesprosess. Den skulle være brukervennlig, moderne designet, utvidbart for flere programmeringsspråk og ha tester for et stort mangfold kandidater med forskjellig ferdighetsnivå. Det ble laget delmål ut ifra disse, som ble knyttet til de fire forskningsspørsmålene fra kapittel 1. Videre ble det gjort en evaluering mot målene, forskningsspørsmålene og problemstillingen som prosjektgruppen skulle besvare:

1. «Hvordan utvikle en testplattform som kan konsekvent sammenligne forskjellige løsninger?»

Prosjektgruppen utviklet en testplattform som lar forskjellige løsninger sammenlignes til den grad av hvor mange enhetstester som passerer. Det var ønskelig at den tillot sammenligning av poengsum på kjøretid, men arbeidet ble ikke ferdigstilt. Prosjektgruppens testplattform sammenligner derfor ikke konsekvent forskjellige løsninger.

2. «Er testplattformen egnet til å skalere for en større brukerbase?»

Prosjektgruppen har gjort arkitekturvalg som tilsynelatende skal være gode skalerbare valg, men har ikke hatt ressurser til å evaluere arbeidet i stor skala. Derimot har prosjekteier evaluert arbeidet og sagt seg svært fornøyd om skaleringsvalgene prosjektgruppen har tatt. Prosjektgruppen konkluderer med at testplattformen er egnet for skalering til en større brukerbase.

3. «Hvordan utvikle en testplattform som er utvidbar for flere programmeringsspråk?»

Prosjektgruppen har levert en plattform som fungerer i JS og valgt arkitektur er en etablert grunnstamme for utvidelse av flere programmeringsspråk. Prosjektet viser derfor konkret hvordan en testplattform kan utvides mot flere programmeringsspråk.

4. «Er testplattformen forståelig og brukervennlig?»

I brukerevalueringen ble det tydelig at testplattformen var godt designet med tanke på brukervennlighet siden ti av tolv stilte seg positiv til brukeropplevelsen. Til tross for de gode tilbakemeldingene var det fortsatt karakter B, som betyr at plattformen hadde forbedringspotensialer som nevnt tidligere. Til tross for dette velger prosjektgruppen å konkludere med at testplattformen er god nok etter endt prosjekt og en dag kan bidra til å effektivisere en ansettelsesprosess.

Selv om det kun ble gjennomført en runde med brukertester, resulterte prosessen i utviklingen av et fungerende produkt. Tilbakemeldingene fra SUS-skjemaet var positive og indikerte at brukerne opplevde produktet som brukervennlig og tilfredsstillende. Dette bekrefter at Lean startup var en egnet utviklingsmetodikk for både prosjektet og prosjektgruppen. Dermed velger prosjektgruppen å konkludere at prosjektgruppen valgte riktig utviklingsmetodikk for prosjektet.

Etter å ha konkludert mål og forskningsspørsmål skal prosjektets problemstilling konkluderes. Ut fra et økonomisk perspektiv kan testplattformen føre til kostnadskutt for bedrifter som bruker den, ved å redusere tiden og ressursene som kreves for å ansette de mest egnede utviklerne. Når det gjelder miljømessige konsekvenser, kan testplattformen bidra til å redusere reise- og transportbehovet for jobbsøkere og arbeidsgivere. Ved å gjøre det mulig å teste ferdighetene sine digitalt, kan det føre til at færre mennesker trenger å reise lange avstander til intervjuer og andre fysiske tester. Dette kan potensielt redusere både utslipp og kostnader.

Hvis en bedrift benytter testplattformen til sin ansettelsesprosess, vil muligens en søknadsrunde og et teknisk intervju bli overflødig. Kombinasjonen av testplattformen og WA.works utgjør et digitalt bindeledd mellom kandidater og bedrifter med hjelp av kodetester. Det åpner en toveis kommunikasjon, der kandidater kan ta tester uoppfordret og bedrifter som får en ny kanal inn i rekrutteringsprosessen sin. Ved å kunne ta en tidlig avgjørelse på om en kandidat er egnet for videre vurdering, kan det argumenteres for en effektivisering. Denne effektiviseringen kommer i form av spart arbeidstimer for en bedrift og kandidater. Prosjektgruppen konkluderer derfor at det er utviklet en testplattform som kan effektivisere en ansettelsesprosess i IT-bransjen.

7.2 Videre arbeid

Som nevnt tidligere i rapporten skal videre arbeid utføres av prosjekteier. Videre arbeid er basert på hva prosjektet ikke oppnådde, tilbakemeldinger fra evalueringen og prosjekteiers videre hensikter med testplattformen.

Prosjektet ønsket en testplattform som konsekvent kunne sammenligne løsninger. Dette ble delvis ferdigstilt, men sammenligning av kjøretid mangler. Prosjektgruppen mener arbeidet er viktig for testplattformen videre. I valgt løsning ble det identifisert behov for å sikre testplattformen mot kodeinjeksjon. Derfor er det nødvendig å snarest etablere tiltak mot dette sikkerhetshullet så snart som mulig, et potensielt tiltak er nevnt i punkt 4.2.1.

Tilbakemeldingene fra brukerevalueringen i punkt 5.2.1 viste noen forbedringer i brukeropplevelsen som bør tas hensyn til i videre arbeid. De omhandlet mangel på navigasjonshjelp, noen knapper for funksjonalitet og beskrivelse til ikonene i navigasjonsbaren. Videre er tilbakemeldingene fra prosjekteier i punkt 5.2.2 arbeid som også kan tas hensyn til.

Som nevnt i punkt 4.1 skal prosjekteier gjøre oppkobling fra testplattformen til sine egne systemer. Dette arbeidet innebærer også å legge til autentisering på REST APIet i henhold til deres brukersystem. Som diskutert i punkt 3.1.4 bør antall tilgjengelige programmeringsspråk utvides på testplattformen. Prosjekteier har i denne kontekst uttenkt seg et program som automatisk lager enhetstester til disse programmeringsspråkene. Dette er et stort stykke arbeid å utføre, men er tidsbesparende ved å ikke manuelt skrive enhetstester.

8 REFERANSER

Andreassen, S. (2023). *personlig samtale*.

Arancio, S. (2021) ReactJS: A brief history
Tilgjengelig fra: https://medium.com/@sjarancio/reactjs-a-brief-history-3c1e969a477f
[Hentet 11.05.2023]

BasuMallick, C. (2022). What is Single Page Applications? Architecture, benefits and challanges.
Tilgjenglig fra: https://www.spiceworks.com/tech/devops/articles/what-is-single-page-application/
[Hentet 06.05.2023]

Big Fish (2021) Hva er UI og UX, og hvorfor er det så viktig for nettsiden din?
Tilgjengelig fra: https://bigfish.no/bransjesnakk/hva-er-ui-og-ux-og-hvorfor-er-det-sa-viktig-for-nettsiden-din/
[Hentet 11.04.2023]

Brooke, J. (1995) SUS: *A quick and dirty usabillity scale,* researchgate
Tilgjengelig fra:
https://www.researchgate.net/publication/228593520_SUS_A_quick_and_dirty_usability_scale
[Hentet 05.05.2023]

CodeAcademy. (u.å.) React
Tilgjengelig fra: https://www.codecademy.com/resources/docs/react
[Hentet 02.02.2023]

Codewars. (u.å.) The Codewars Docs
Tilgjengelig fra: https://docs.Codewars.com/
[Hentet 07.03.2023]

docs.hangfire.io. (u.å.). Dokumentasjon.
Tilgjengelig fra: https://docs.hangfire.io/en/latest/
[Hentet 04.05.2023]

EntityFrameworkTutorial. (u.å.). What is Code-first?
Tilgjengelig fra: https://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx
[Hentet 05.05.2023]

Fette, I. & Melnikov, A. (2011). The Websocket Protocol.
Tilgjengelig fra: https://www.rfc-editor.org/rfc/pdfrfc/rfc6455.txt.pdf
[Hentet 09.05.2023]

Finley, K. (2014). The history of the new stack: scale out architecture.
Tilgjengelig fra: https://thenewstack.io/the-history-of-the-new-stack-scale-out-architecture/
[Hentet: 08.03.2023]

Goltzsche, D., Nieke, M., Kapitza, R. og Knauth, T. (2019). AccTEE: A WebAssembly-based Two-way Sandbox for Trusted Resource Accounting.
Tilgjengelig fra: https://www.ibr.cs.tu-bs.de/users/goltzsch/papers/mw19-acctee.pdf
[Hentet 07.05.2023]

HackerRank (u.å.) About Us, HackerRank.
Tilgjengelig fra: https://www.hackerrank.com/about-us/
[Hentet: 03.05.2023]

IBM. (u.å.). REST API.
Tilgjengelig fra: https://www.ibm.com/topics/rest-apis
[Hentet 06.05.2023]

Kirsch, C. M., Payer, H., Röck, H. & Sokolova, A. (2012). Performance, Scalability, and Semantics of Concurrent FIFO Queues
Tilgjengelig på: https://cs.uni-salzburg.at/~anas/papers/Scal-ICA3PP.pdf
[Hentet 19.05.2023]

Microsoft. (2021). Enitity Framework Core.
Tilgjengelig fra: https://learn.microsoft.com/en-us/ef/core/
[Hentet 10.05.2023]

Microsoft. (2023). Overview of ASP.NET core SignalR.
Tilgjengelig fra: https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?WT.mc_id=dotnet-35129-website&view=aspnetcore-7.0
[Hentet 04.05.2023]

Odongo, K. (2021). Object Oriented Programming with TypeScript
Tilgjengelig på: https://dev.to/kevin_odongo35/object-oriented-programming-with-typescript-574o
[Hentet 17.05.2023]

OWASP, (2021). A03:2021 Injection – OWASP Top 10:2021
Tilgjengelig fra: https://owasp.org/Top10/A03_2021-Injection/
[Hentet 15.05.2023]

R. Røtnes, F. W. Eggen, J. Måøy, M. N. Schultz, J. I. Steen, (2021). Behov for og tilbud av IKT-kompetanse (Rapport nr. R1-2021).
Tilgjengelig fra: https://www.tekna.no/globalassets/filer/rapporter/arbeidsmarked/r1-2021behov-for-og-tilbud-av-ikt-kompetanse-v3-190121.pdf
[Hentet 07.03.2023]

Rossen, E. (1997) 'JavaScript blir europeisk standard',
Tilgjengelig fra: https://www.digi.no/artikler/javascript-blir-europeisk-standard/342135
[Hentet 04.05.2023]

Raaheim, J.M. (2012). Lean startup, noe for deg?
Tilgjengelig fra: https://www.tekna.no/fag-og-nettverk/ledelse-og-utvikling/ledelsesbloggen/lean-startup-noe-for-deg/
[Hentet 11.05.2023]

Sarah K. (2015) The Steve Jobs Product Fallacy: "Customers Don't Know What They Want"
Tilgjengelig på: https://www.linkedin.com/pulse/steve-jobs-product-fallacy-customers-dont-know-what-want-sarah-kling
[Hentet 23.04.2023]

Sauro, J. (2011) Measuring Usability with the System Usability Scale (SUS)
Tilgjengelig fra: https://measuringu.com/sus/
[Hentet 06.05.2023]

Sperre. F. E. (2015). Hva koster det å gjøre en rekruttering selv?
Tilgjengelig fra:
https://no.linkedin.com/pulse/hva-koster-det-å-gjøre-en-rekruttering-selv-eirik-f-sperre
[Hentet 12.05.2023]

Stackoverflow. (2021). Stack Overflow Developer Survey – integrated development environment.
Tilgjengelig fra: https://insights.stackoverflow.com/survey/2021#technology
[Hentet 11.05.2023]

Stokke, O.P.B.S. (2021). Ansetter gjerne utviklere uten utdanning i 2021
Tilgjengelig fra: https://www.kode24.no/kodenytt/ansetter-gjerne-utviklere-uten-utdanning-i-2021/73242353
[Hentet 05.03.2023]

Surdal, S. O. og Aadland, R. (2021). Nettbasert testing av kodeferdigheter for rekrutterings formål.
Tilgjengelig fra: https://hvlopen.brage.unit.no/hvlopen-xmlui/handle/11250/3026553
[Hentet 08.03.2023]

Thomas, D. (2021). Facebook changes its name to Meta in major rebrand.
Tilgjengelig fra: https://www.bbc.com/news/technology-59083601
[Hentet 11.05.2023]

Zolotarev, J. (2022). What is Typescript?
Tilgjengelig fra: https://builtin.com/software-engineering-perspectives/typescript
[Hentet 17.05.2023]

8.1 Referanseliste figurer

Figur 4-6 Entityframeworktutorial.net. (n.d.). *What is Code-First?*
Tilgjengelig fra: https://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx.
[Hentet: 15.april 2023]

Figur 3-3 Jimeno, J.L. (2021) Lean Startup: the «Ukemi» of Product Development
Tilgjengelig fra: Innovation https://netmind.net/en/lean-startup-the-ukemi-of-product-development-innovation/
[Hentet: 11.mai 2023]

8.2 Liste over brukte verktøy og teknologier

.NET. (2023). (Version 7.0). Microsoft Corporation.

ChatGPT. (2023). (GPT 3.5). OpenAI.

CodeAcademy. (2023). CodeAcademy.

CSS. (2023). (CSS3). World Wide Web Consortium.

DBeaver. (2023). (Version 22.0.4). DBeaver Corporation.

Docker. (2023). (Version 24.0.0). Docker Inc.

Drawio. (2023). (Version 21.3.1). JGraph Ltd.

Figma. (2023). Figma.

Gantt-diagram. (2023). Microsoft Corporation.

Hangfire. (2023). (Version 1.8.1) Hangfire OU.

Microsoft Excel. (2021). (Version 20211). Microsoft Corporation.

Microsoft Word. (2021). (Version 2021). Microsoft Corporation.

Monaco. (2023). (Version 0.38.0). Microsoft Corporation.

MSSQL. (2023). (Version 16.0). Microsoft Corporation.

Node. (2023). (Version 19.9.0). OpenJS Foundation.

Npm. (2023). (Version 9.6.3). Npm inc.

PostgreSQL. (2023). (Version 15.3). The PostgreSQL Global Development Group.

Postman. (2023). (Version 10.14) Postman Inc.

React. (2023). (Version 18.0). Meta.

SignalR. (2023). (Version 2.4.3) Microsoft Corporation.

Visual Studio Code. (2023). (Version 1.78). Microsoft Corporation.

9 VEDLEGG

Vedlegg 1: Prosjektbeskrivelse

EB-15 Test Plattform (Wide Assessment)

WA.works er en rekrutteringsplattform som er skreddersydd for IT-bransjen og den høye etterspørselen etter teknologer. Plattformen har snudd opp ned på den tradisjonelle ansettelsesprosessen ved å fokusere mer på kandidatjakt enn jobbjakt, og dermed gjort det enkelt for bedrifter å headhunte utviklere, designere, prosjektledere og andre IT-roller.

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. I dag har vi et prosjekt hvor en kandidat kan gjennomføre en test i css. Vi ønsker å videreutvikle den til å støtte javascript

- 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.

Kontaktperson

Andreas Hammerbeck, CTO

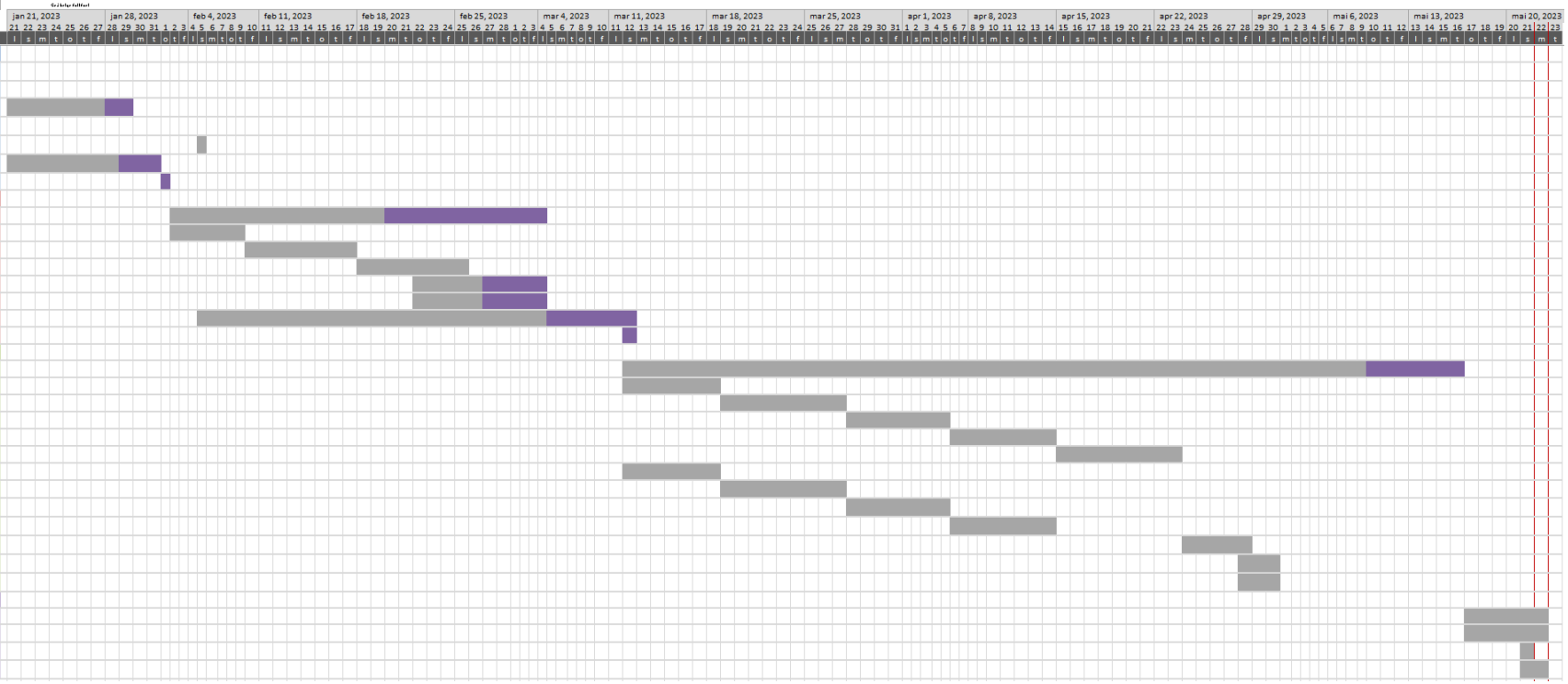
- Andreas@wa.works
- 97661466

Vedlegg 2: Gantt diagrammet

Test Plattform
Wide Assessment AS

Project Start: tir, 10.1.2023
Display Week: 3

TASK	ASSIGNED	PROGR	START	END
Iterasjon 1: Planlegging				
Oppstartsmøte WA og HVL veileder	Alle	100 %	1.10.23	1.10.23
Onboarding	Alle	100 %	1.19.23	1.19.23
Prosjekthåndbok og visjonsdokument i	Alle	90 %	1.19.23	1.29.23
Statusmøte HVL veileder	Alle	100 %	1.20.23	1.20.23
Statusmøte HVL veileder	Alle	100 %	2.5.23	2.5.23
Modeller og diagrammer for prosjekte	Alle	80 %	1.21.23	1.31.23
Første hypotesetest + evaluering	Alle	70 %	2.1.23	2.1.23
Iterasjon 2: Tilegning av kunnskap og rapport				
CodeAcademy	AM og Anine	60 %	2.2.23	3.4.23
Backend: EntityFramework	Birk Johannessen	100 %	2.2.23	2.9.23
Backend: API	Birk Johannessen	100 %	2.10.23	2.17.23
Backend: Database	Birk Johannessen	100 %	2.18.23	2.25.23
Frontend: React - Egne prosjekter	AM og Anine	50 %	2.22.23	3.4.23
Frontend: Javascript - Egne prosjekter	AM og Anine	50 %	2.22.23	3.4.23
Midtveisrapport	Alle	80 %	2.5.23	3.12.23
Første test av MVP med prosjekteier +	Alle	40 %	3.12.23	3.12.23
Iterasjon 3 Rapport og implementering				
Jobbe med rapport	Alle	90 %	3.12.23	5.16.23
Backend: EntityFramework	Birk Johannessen	100 %	3.12.23	3.18.23
Backend: REST API	Birk Johannessen	100 %	3.19.23	3.27.23
Backend: Hangfire	Birk Johannessen	100 %	3.28.23	4.5.23
Backend: Utførelsesmiljø	Anders-Marius Gjerdalen	100 %	4.6.23	4.14.23
Backend: SignalR	Birk Johannessen	100 %	4.15.23	4.23.23
Frontend: Admin side	Birk Johannessen	100 %	3.12.23	3.18.23
Frontend: Oversikt over tester modul	Anine Hammersborg	100 %	3.19.23	3.27.23
Frontend: Oppgaveside modul	Anine Hammersborg	100 %	3.28.23	4.5.23
Frontend: Monaco Editor	Anders-Marius Gjerdalen	100 %	4.6.23	4.14.23
Frontend: Implementering mot API	Anine Hammersborg	100 %	4.24.23	4.28.23
Frontend: SignalR WebSocket	Anders-Marius Gjerdalen	100 %	4.28.23	4.30.23
Første test av en funksjonell MVP	Alle	100 %	4.28.23	4.30.23
Iterasjon 4 Ferdigstilling av rapport og produkt				
Rett- og finskriving	Alle	100 %	5.17.23	5.22.23
Omskriving	Alle	100 %	5.17.23	5.22.23
Oppsummering med veileder	Alle	100 %	5.21.23	5.21.23
Endelig klargjøring av rapport	Alle	100 %	5.21.23	5.22.23



Vedlegg 3: Risikoanalyse

	Hendelse /Risiko	Årsak	Sannsynlighet	Konsekvens	Risiko-produkt	Tiltak
1	Feil implementering av kode	Misforstått kravet fra oppdragsgiver	Høy (4)	Høy(4)	Høy(16)	Iterativ utvikling med oppdragsgiver i sentrum
2	Prosjektet blir for stort og uspesifikt.	Store krav fra oppdragsgiver, høy ambisjon fra prosjektgruppen	Lav (2)	Høy (4)	Middles (8)	Begrense omfanget, sette klart mål for produktet, samtale med oppdragsgiver.
3	Konflikt i prosjektgruppen	Uenigheter eller personlige konflikter	Middels (3)	Middels (3)	Middels (9)	Åpen dialog og ærlighet
4	Ujevn oppgavefordeling	Dårlig kommunikasjon	Lav (2)	Middels (3)	Middels (6)	Gantt diagram utformes i fellesskap
5	Glemmer avtaler	Mangel på struktur	Svært lav (1)	Middels (3)	Lav (4)	Påminne hverandre
6	Opprettholde HVL sine frister	Urealistiske tidsfrister	Lav (2)	Middels (3)	Middels (6)	Følge med og eventuelt sende mail om utsettelse

Vedlegg 4: Risikoutgreningen

Sannsynlighet	Svært Høy (5)	5	10	15	20	25
	Høy (4)	4	8	12	16	20
	Middels (3)	3	6	9	12	15
	Lav (2)	2	4	6	8	10
	Svært Lav (1)	1	2	3	4	5
		Svært Lav (1)	Lav (2)	Middels (3)	Høy (4)	Svært Høy (5)
	Konsekvens					

Vedlegg 5: SUS regler for utregning

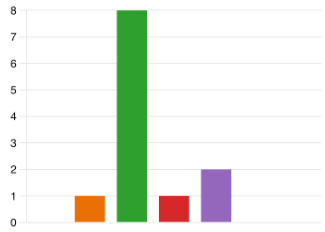
Hvordan utregne SUS score:
Summer totalen av svarene
For oddetallsspørsmål: trekk fra 1 fra scoren
For partallsspørsmål: trekk bruker svaret fra 5
Får dermed to verdier
som summeres sammen og ganges med 2,5
for å bli en skala fra 0-100 istedet for 0-40
Summen av oddetall= summen -5
Summen av partall= summen*-1+25

Vedlegg 6: Evaluering av brukertest - diagrammer

Alder* Hvilken aldersgruppe faller du under? (0 poeng)

Flere detaljer Innblikk

16-18	0
18-21	1
21-25	8
25-30	1
30-45	2
45-60	0
60+	0



Situasjon (0 poeng)

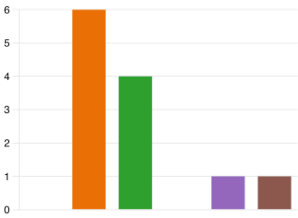
Flere detaljer Innblikk

Student - informatikk	10
Jobber som utvikler	2
Selvlært	0

Om du er student* Hvilket år er du på? (0 poeng)

Flere detaljer Innblikk

1 år	0
2 år	6
3 år	4
4 år	0
5 år	1
5 eller mer	1



Erfaring* Om du har jobber som utvikler. Hvor lenge har du jobbet?* (0 poeng)

Flere detaljer

Nyansatt	3
1-2 år	0
3-5 år	1
5-10 år	0
Over 10 år	1

Har du hørt om Wa.works før? (0 poeng)

Flere detaljer Innblikk

Ja	2
Nei	10

Om du har hørt om Wa.Works før* Har du en brukerprofil der? (0 poeng)

Flere detaljer Innblikk

Ja	0
Nei	12

Vedlegg 7: Tilbakemelding WA

Er oppgaven fullført?	Oppgaven er løst på en veldig bra måte. Kvaliteten på koden er over all forventing vi har fra studenter på dette nivået. Det er tatt hensyn til alt fra UX, clean code og datasikkerhet. Gruppen har tatt godt hensyn til skalering av testplattformen. Det er satt opp bakgrunnsjobber som kan behandle de tunge oppgavene, noe som betyr at høy last for en bruker ikke vil ha noen innvirkning på andre. Plattformen vil fungere utmerket samtidig som brukermassen stiger og være enkel å skalere opp. Gruppen har klart å utføre oppgaven i ett mye større scope og med flere teknologier enn det vi er vandt til fra tidligere bachelorgrupper. Vi i WA er enige om at dette er den beste oppgaven vi har fått levert gjennom våre 8 års erfaringer med bachelorprosjekter.
Oppfyller vi kravene som ble satt i starten?	Alle kravene i som er satt i oppgaven er løst.
Kunne vi gjort noe annerledes?	Noen mindre UX feil i React, men dette er pirking. Vi er veldig fornøyde alt i alt, og kan ikke se hvordan gruppen skulle gjort noe annerledes.
Kunne dette produktet bli tatt i bruk?	Produktet må kobles sammen med vårt system før det kan brukes. Utenom det trengs det minimale endringer for at produktet er klart for bruk.

Vedlegg 8: Evaluering av brukertest – Fritekst

10. Brukeropplevelse* Hadde du en dårlig opplevelse? Fortell hvorfor

8 Svar

ID ↑	Navn	Svar
1	anonymous	NICE!
2	anonymous	Det funket fint for det som var implementert. Ganske god opplevelse
3	anonymous	Det er vanskelig å se hva iconene er. Iconen burde hatt en info pop-up som sier noe om ikonet. Er litt vanskelig å navigere seg rundt. Samtidig virker ikke 'save' button når man har skrevet koden.
4	anonymous	Nei, testene og oversikten over progresjon gjorde at jeg hadde en god opplevelse
5	anonymous	Var ikke helt klart hvor man fant oppgavene i starten
6	anonymous	Det fungerte bra. Bra med skjulte unittester.
7	anonymous	Var litt usikker hvor man skulle trykke for å finne oppgave
8	anonymous	

12. Brukerforståelse* Var det vanskelig å navigere seg rundt på nettsiden? Fortell hva du ville endret

7 Svar

ID ↑	Navn	Svar
1	anonymous	eneste: kunne hatt en knapp ellerno til kodetestene som er lettere å se
2	anonymous	Nettsiden var ganske forståelig, enkel og grei. Noen av ikonene i nav-bar kunne ha vært mer representative
3	anonymous	Det har med iconen fra forrige svar å gjøre.
4	anonymous	Det var oversiktlig, men var litt vanskelig å vite hva symbolene på venstre side kom til å ha som funksjon
5	anonymous	Tydeligere måte å finne oppgavene på siden
6	anonymous	Kanskje litt knot å gå videre til neste test etter man er ferdig med en.
7	anonymous	Hvor jeg skulle navigere fra startsiden for å finne testene. Navnet på testene kan være lite beskrivende, må inn på test for å se hva testen går ut på. Måtte bruke tid på å "tolke" unit testene.

13. Mangler* Var noe uklart med nettsiden? Fortell hva du syntes mangler

10 Svar

ID ↑	Navn	Svar
1	anonymous	eneste: kunne hatt en knapp ellerno til kodetestene som er lettere å se
2	anonymous	nav-bar kunne ha hatt bedre ikoner for hva du trykket på, men funket greit
3	anonymous	Kanskje at man kan se en score på hvor bra en har gjort det.
4	anonymous	Nei, vet ikke hva det skulle være.
5	anonymous	hvor man finner oppgavene ved oppretting av bruker
6	anonymous	
7	anonymous	Menyen med ikoner hadde ikke beskrivelse, så måtte gjette meg frem til hva som var hvor.
8	anonymous	
9	anonymous	Manglende funksjonalitet bare
10	anonymous	

Vedlegg 9: SUS karakterer

SUS Tolkning

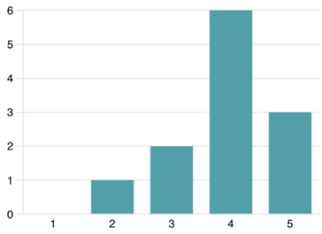
SUS Score	Grade	Adjectival Rating
>80.3	A	Excellent
68-80.3	B	Good
<68>	C	Okay
51-68	D	Awful
<51	F	Poor

Vedlegg 10: Evaluering av brukertest – SUS

1. I think that i would like to use this system frequently (0 poeng)

Flere detaljer Innblikk

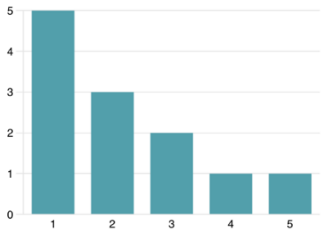
3.92
Gjennomsnittlig vurdering



2. I found the system unnecessarily complex (0 poeng)

Flere detaljer Innblikk

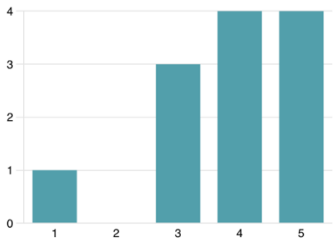
2.17
Gjennomsnittlig vurdering



3. I thought the system was easy to use (0 poeng)

Flere detaljer Innblikk

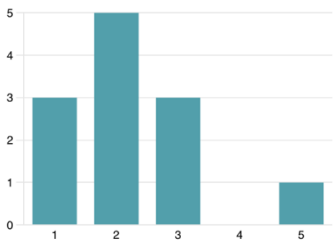
3.83
Gjennomsnittlig vurdering



4. I think that I would need the support of a technical person to be able to use this system (0 poeng)

Flere detaljer Innblikk

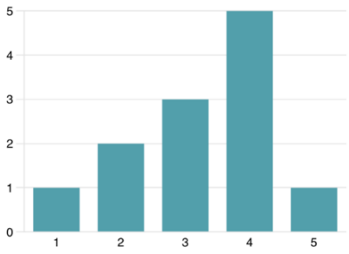
2.25
Gjennomsnittlig vurdering



5. I found the various functions in this system were well integrated (0 poeng)

Flere detaljer Innblikk

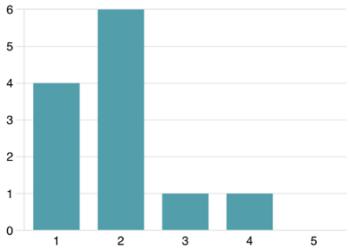
3.25
Gjennomsnittlig vurdering



6. I thought there was too much inconsistency in this system (0 poeng)

Flere detaljer Innblikk

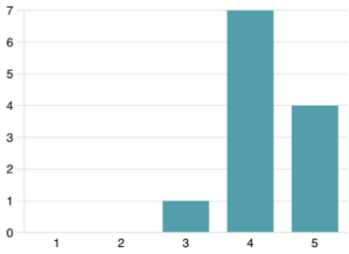
1.92
Gjennomsnittlig vurdering



7. I would imagine that most people would learn to use this system very quickly (0 poeng)

Flere detaljer Innblikk

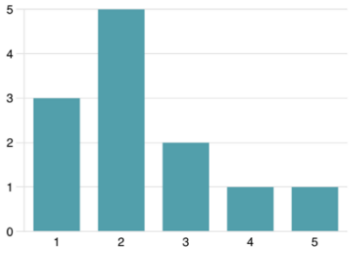
4.25
Gjennomsnittlig vurdering



8. I found the system very cumbersome to use (0 poeng)

Flere detaljer Innblikk

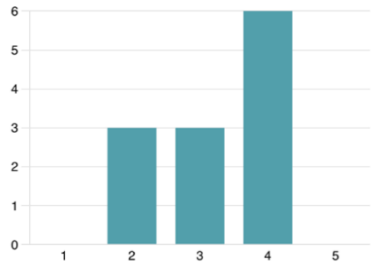
2.33
Gjennomsnittlig vurdering



9. I felt very confident using the system (0 poeng)

Flere detaljer Innblikk

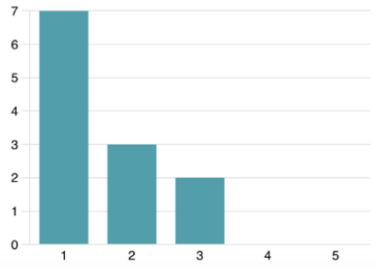
3.25
Gjennomsnittlig vurdering



10. I needed to learn a lot of things before I could get going with this system (0 poeng)

Flere detaljer Innblikk

1.58
Gjennomsnittlig vurdering



Vedlegg 11: SUS Score utregning med graf

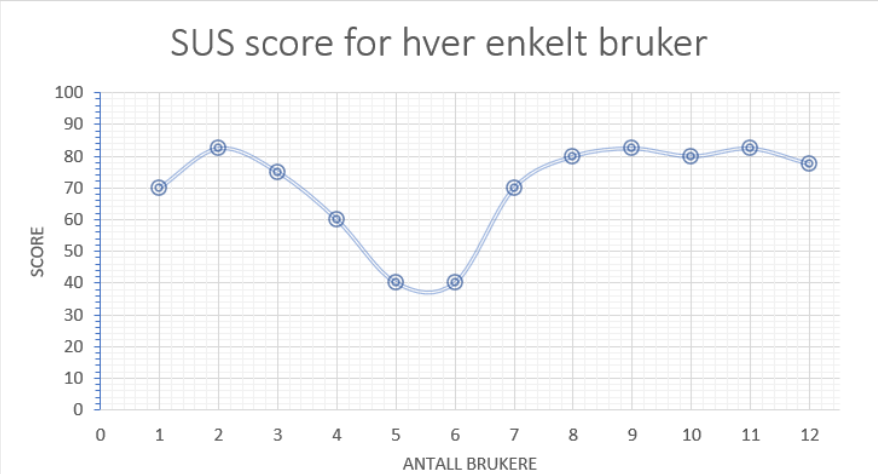
SUS Undersøkelse

ID	1	2	3	4	5	6	7	8	9	10	SUS Score
1	2	2	3	3	5	1	4	2	3	1	70
2	3	1	5	1	3	2	4	1	4	1	82,5
3	3	2	3	2	4	1	4	2	4	1	75
4	3	3	2	3	3	2	4	3	4	1	60
5	5	4	1	3	2	4	4	4	3	4	40
6	4	1	5	2	2	3	4	3	3	1	40
7	4	2	4	1	4	2	4	2	4	1	70
8	4	1	5	1	4	2	5	2	3	2	80
9	3	1	4	2	4	1	5	2	3	1	82,5
10	5	3	5	2	4	1	4	1	4	2	80
11	5	2	4	2	3	2	5	1	4	3	82,5
12	4	1	4	2	4	1	5	2	4	3	77,5

Hvordan utregne SUS score:

Summer totalen av svarene
For oddetallsspørsmål: trekk fra 1 fra scoren
For partallsspørsmål: trekk bruker svaret fra 5
Får dermed to verdier
som summeres sammen og ganges med 2,5
for å bli en skala fra 0-100 istedet for 0-40

Summen av oddetall= summen -5
Summen av partall= summen*-1+25



AVG:
70

SUS Tolkning

SUS Score	Grade	Adjectival Ratin
>80.3	A	Excellent
68-80.3	B	Good
<68>	C	Okay
51-68	D	Awful
<51	F	Poor

Summen av oddetall= summen -5	12	14	13	11	10	10	13	15	16	14	17	16
Summen av partall= summen*-1+25	16	19	17	13	6	6	15	17	17	18	16	15
SUS score:	70	82,5	75	60	40	40	70	80	82,5	80	82,5	77,5