OSLOMET

oxx

Gamification av ansettelser

Bachelorprosjekt 2021

Gruppe 31

Christian Dyrli

Jørgen Borander

Maximian Stiegler Sharoyan

Sondre Næbb Bjarkum

Institutt for Informasjonsteknologi
Postadresse: Postboks 4 St. Olavs plass, 0130 Oslo
Besøksadresse: Holbergs plass, Oslo

PROSJEKT NR.
31

TILGJENGELIGHET
Offentlig

Telefon: 22 45 32 00

BACHELORPROSJEKT

HOVEDPROSJEKTETS TITTEL	DATO
Gamification av ansettelser	26.05.2021
	ANTALL SIDER / BILAG 143
PROSJEKTDELTAKERE Bjarkum, Sondre Næbb Borander, Jørgen Dyrli, Christian Sharoyan, Maximilian Stiegler	INTERN VEILEDER Yamashita, Aiko

OPPDRAGSGIVER	KONTAKTPERSON
OXX AS	Venold, Anders

SAMMENDRAG

Gruppen har utviklet et spill designet for bruk på stands, og øvrig promotering av bedriften OXX. Spillet er laget for å kunne assistere OXX under ansettelsesprosessen så vel som promotering, og inneholder derfor evnetester underveis. Prosjektet har også en administratorside for gjennomgang av registrerte kandidater.

Løsningen er laget i ASP.NET Core 5 og inneholder en evnetest-del, en administrator-del samt API for lagring og henting av resultater fra database. Løsningen er laget for, og har blitt testet med hosting i Azure.

3 STIKKORD
Gamification
Ansettelse
Webapplikasjon

Forord

Denne rapporten tar for seg hele prosessen fra planlegging til sluttprodukt, av vårt bachelorprosjekt ved OsloMet – Storbyuniversitet våren 2021. Bachelorprosjektet består av en webapplikasjon for spillifisering av ansettelsesprosessen til oppdragsgiveren OXX.

Vi ønsker å takke OXX for muligheten til å gjennomføre et veldig spennende prosjekt og veldig fagrelevant oppgave. Samtidig takke for oppfølgning og veiledning underveis i prosjektet, og spesielt takk vår interne veileder Aiko Yamashita og vår eksterne veileder Anders Venold for god hjelp.

Til slutt vil vi takke samboere og venner som har vært en god hjelp på humør og motivasjon under en veldig spesiell bachelor periode.

Innholdsfortegnelse

1. Innledning

Gamification av ansettelser

Oslomet – Gruppe 31 – vår 2021

Skrevet av

Christian Dyrli

Jørgen Borander

Maximilian Stiegler Sharoyan

Sondre Næbb Bjarkum

1.1 Forord

Denne rapporten tar for seg hvordan det utvikles et supplement til OXX sin ansettelsesprosess. Det ble laget en spillifisert webapplikasjon med evnetester for å hjelpe OXX til å fremstå som en attraktiv arbeidsgiver for eventuelle kandidater, samt teste kandidaters kunnskap for vurdering om ansettelse.

Prosjektet har tatt i bruk relevante teknologier fra IT-bransjen, og spesielt OXX, for å utvikle et produkt bedriften enkelt kan implementere i eksisterende systemer.

Løsningen er implementert i rammeverket .NET Core 5. Gruppen har ved hjelp av teknologier som Phaser, Javascript og C#, laget et spill hvor kandidater blir introdusert til OXX, gjennomfører evnetester, og kan melde sin interesse for jobb hos OXX.

Prosjektet har utfordret og utviklet gruppens kunnskap og erfaring, og har gjennom diverse teknologier og arbeidsmetoder gitt relevant erfaring mot arbeidslivet. Løsningen demonstrerer resultatet av vår bachelorutdanning ved OsloMet.

1.2 Gruppemedlemmer

Christian Dyrli

Christian studerer informasjonsteknologi, med bakgrunn fra sykepleiestudiet og interesse for data og spill siden videregående skole. I løpet av studiet har han jobbet som promotør for Oslomet sine tre it-linjer, samt som Orakel (Studentassistent for førsteårsstudenter med fokus på å lære nye studenter hvordan de selv kan finne frem til svarene de trenger)

Jørgen Borander

Jørgen studerer dataingeniør og har som de andre en stor interesse for data, teknologi og spill. I løpet av studiet har han tatt et verv som studentkontakt i Tekna, hvor oppgavene har gått i å stå på stand og arrangere ulike kurs som gjør studielivet til medstudentene litt lettere.

Maximillian Stiegler Sharoyan

Max studerer anvendt datateknologi og har gjennom hele livet hatt en interesse for data, teknologi og spill. Han har tidligere studert statsvitenskap og har ved siden av studiene jobbet som Orakel og har vært veldig engasjert i aksjegruppen på OsloMet.

Sondre Næbb Bjarkum

Sondre studerer Anvendt Datateknologi og spesialiserer seg på webutvikling, det være innenfor .Net Core eller Javascript. Lidenskapen er frontendutvikling, men stiller med gode ferdigheter i backend og anses å være en fullstack webutvikler. Utover å jobbe som ølspesialist hos Gulating Günerløkka, har han blant annet jobbet som Orakel og Studentmentor ved siden av studiene.

1.3 Oppdragsgiver

OXX startet opp i 2000 og leverer rådgivning, UX design og teknisk utvikling av avanserte løsninger. De har siden oppstart hatt en god og solid vekst i deres markedssegment.

OXX lager og leverer totaldigitale løsninger under tett samarbeid med sine kunder. Målet deres er å skape konkurransefortrinn og forretningsverdier ovenfor deres kunder.

OXX sitt mål med dette prosjektet er å kunne bruke løsningen som et hjelpemiddel for å skape oppmerksomhet rundt bedriften, samtidig teste og evaluere kandidater for mulig ansettelse. Oppdragsgiver valgte denne gruppen av flere årsaker. Blant annet dekker gruppen hele spekteret av IT-studier ved OsloMet, gruppen har sterke akademiske resultater og fagkompetansen matchet OXX sin teknologistack.

1.4 Veiledere

1.4.1 Veileder fra OsloMet

Aiko Yamashita

Stillinger:

- **Førsteamanuensis ved OsloMet (Fakultet for teknologi, kunst og design. Institutt for informasjonsteknologi Menneske-maskin interaksjon og universell utforming av IKT)**
- **Senior data scientist ved CoE Advanced Analytics - DNB**

Aiko er veilederen vår gruppe ble tildelt av OsloMet. Hun stiller sterkt med over 20 års erfaring fra blant annet doktorgrad i informatikk ved Universitet i Oslo, samt mange års arbeidserfaring innenfor IT. Aiko har bidratt med gode tilbakemeldinger og tips underveis i prosjektet.

1.4.2 Veileder fra OXX

Anders O. Venold

Stilling: Development Manager

Anders har fungert som vår veileder og kontaktperson fra OXX. Han har jobbet hos OXX i over 13 år fordelt på to perioder, og har nå stilling som Development manager i bedriften. Han har hjulpet gruppen med tekniske spørsmål ift. bl.a. Microsoft Azure, samt satt oss i kontakt med øvrige ansatte i bedriften i tilfellende der deres kompetansenivå overgår hans eget.

1.5 Hvorfor valgte gruppen denne oppgaven?

Gruppen var tidlig ute med å kontakte bedrifter for å finne oppgave. Flere oppgaver ble vurdert, og gruppen besluttet i plenum at oppgaven til OXX var den mest spennende og utfordrende. En faktor som bidro til valget, er at gruppen hadde tidligere erfaring med teknologistacken til OXX. Dette ga gruppen en trygghet i å kunne utvikle et best mulig produkt. I tillegg fremsto OXX som genuint interessert i å følge opp gruppen og tilby grundig veiledning underveis. Noe et fåtall av bedrifter gruppen var i kontakt med hadde kapasitet til.

1.6 Gruppens mål for prosjektet

Gruppens mål er å utvikle en webapplikasjon som sorterer ut kandidater via en poengsum, regnet utfra evnetester, som et supplement til ansettelsesprosessen til oppdragsgiver. Samtidig skal løsningen skape oppmerksomhet rundt oppdragsgiver. Med oppmerksomhet menes det at løsningen oppleves som underholdende i kontrast med andre bedrifters løsninger. Dette er ment for å motivere og tiltrekke flest mulig kandidater.

1.7 Prosjektet

OXX ønsker med dette prosjektet at det lages en webapplikasjon som de kan benytte seg av til bruk på stands for å bistå med promotering av bedriften, og samtidig teste kunnskapen til potensielle kandidater for ansettelse. Ettersom OXX er en liten aktør med kun 35 ansatte er det ønskelig å bevisstgjøre målgruppen om bedriften og presentere OXX som en attraktiv og spennende arbeidsgiver.

1.7.1 Problemstilling

Den originale oppgaven som definert av oppdragsgiver: "Målsetningen med oppgaven er å gjennomføre et prosjekt hvor det skal utvikles en webløsning for å teste aktuelle kandidater i en ansettelsesprosess. Selv om sluttproduktet er viktig, vil selve prosjektgjennomføringen være en stor del av oppgaven. Webløsningen trenger ikke være komplett, men bør minimum være en fungerende prototype som kan benyttes som utgangspunkt for videre utvikling."

Under tidlige samtaler med bedriften forklarte OXX at problemstillingen kun er satt som en generell retning for oppgaven. OXX ønsket ikke å begrense gruppen med oppgaveteksten, derimot ønsket de input og ideer utenifra rundt hvordan ansettelsesprosessen kunne justeres og eventuelt kombineres med promotering. I samarbeid ble det derfor satt rammer for kravspesifikasjon som kan leses i kapittel 6. Kravspesifikasjoner. Etter flere gruppemøter og samtaler ble det konkludert med at oppgaven skulle være en webløsning med fokus rundt følgende punkter:

1. Teste nye kandidater i grunnleggende forståelse rundt programmering
2. Skape oppmerksomhet rundt selskapet
3. Fremstille OXX som en attraktiv arbeidsgiver
4. Introdusere kandidaten til selskapet

Disse punktene ble valgt da de tydeliggjør målet med prosjektet og samtidig fortsatt svarer på den originale oppgaven.

1.7.2 Målgruppe

Prosjektet retter seg mot IT-studenter og nyutdannete innenfor IT. For å nå ut til denne målgruppen er det ønskelig å bruke løsningen på stands hos forskjellige universiteter, og promotere den i sosiale medieplattformer som oppdragsgiver benytter.

1.8 Gruppens ønske med oppgaven

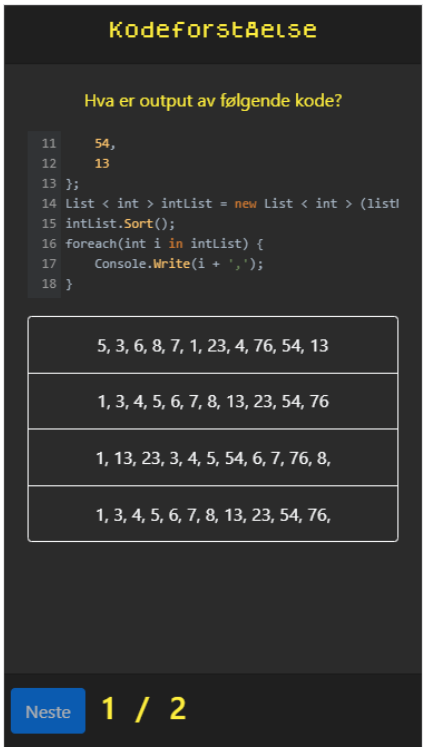
Gruppens ønske var å kunne stille med et ferdig produkt, men dette viste seg å være urealistisk med tanke på tidsaspektet. Derav ble det definert et mer realistisk omfang av oppgaven. Det ble da bestemt at de mest essensielle funksjonalitetene for en fungerende løsning, skulle fullføres. Eventuelt en HiFi-prototype av den ønskede løsningen.

1.9 Sluttprodukt

Dette punktet vil kort gå over resultatet av oppgaven for å gi leseren et bilde av prosjektet Punktet inneholder et utvalg illustrasjoner samt medfølgende forklaring. Nøyere dokumentasjon og gjennomgang vil finne sted under produktdokumentasjonen

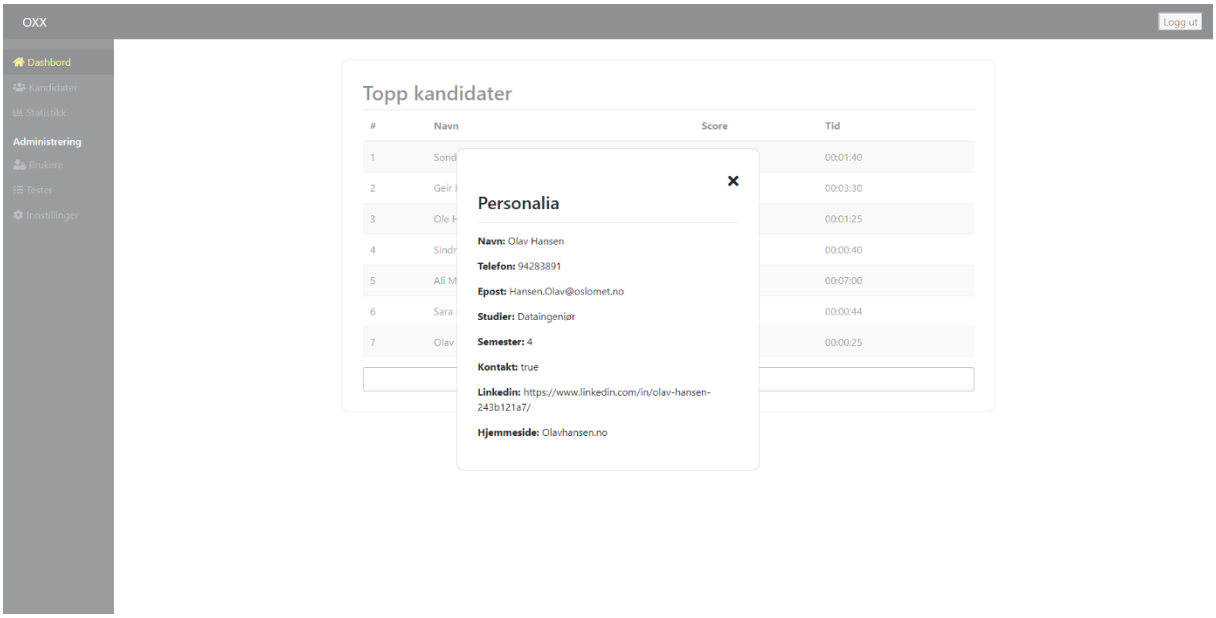
Figur 1: Oversiktsbildet over kartet i spillet.

Figur 1 illustrerer kartet i spillet. Innenfor disse rammene kan bruker bevege seg fritt og interagere med andre karakterer. Dialogene er en kombinasjon av kort informasjon om OXX og innledning til oppgaver karakterene trenger hjelp med. Sistnevnte fungerer som kontekst og insentiv til hvorfor spilleren skal gjennomføre tester.



Figur 2: Et eksempel på en oppgave.

Figur 2 viser et eksempel på en oppgave som brukeren kan få under gjennomgangen. Oppgavene dekker varierende kunnskap innen utvikling fra gruppens bachelorutdanning ved OsloMet. Etter gjennomføring blir kandidaten presentert med et registrerings skjema. Dersom kandidaten ønsker det, lagres total score, tid brukt på oppgavedelene og kandidatens personalia i database og hentes ut på administratorsiden, slik at OXX kan vurdere ulike kandidater enkelt.



Figur 3: Viser Dashbord med kandidat detaljer på administratorsiden.

Figur 3 viser hvordan administratorsiden er tiltenkt brukt for ulik funksjonalitet som gjennomgang av kandidater og resultater, samt registrering og redigering av administratorer og tester.

1.10 Teknologier og verktøy

Tabell 1 viser hvilke teknologier og verktøy som ble benyttet i løpet av prosjektet, se kapittel 2 for definisjoner av teknologiene og verktøyene.

Tabell 1: Oversikt over verktøy og teknologier.

Frontend	Beautify.js CodeMirror CSS HTML Javascript JSON Phaser.js
Backend	ASP.NET Core 5 C# Entity Framework
Database	Microsoft SQL Server
Verktøy	Adobe Illustrator GitHub Jira Lucidchart.com Microsoft Azure Microsoft SQL Server Management

	Pixilart.com Postman Tiled Trello Visual Studio 2019 Visual Studio Code
Kommunikasjon	Facebook Messenger Microsoft Teams

2. Teknologier og verktøy

Gamification av ansettelser

Oslomet – Gruppe 31 – vår 2021

Skrevet av

Christian Dyrli

Jørgen Borander

Maximilian Stiegler Sharoyan

Sondre Næbb Bjarkum

2.1 Forord

I samarbeid med OXX valgte gruppen å utvikle applikasjonen med Microsoft sin softwarestack, da dette samsvarer med teknologiene OXX anvender. Denne seksjonen inneholder en liste av de viktigste teknologiene fra punkt 1.10. Øvrige teknologier står beskrevet i ordlisten under punkt 9. Det anbefales å se beskrivelsen av teknologiene leser ikke kjenner til. For øvrig kan leser gå videre til punkt 3.

2.2 Viktige teknologier og verktøy

ASP.NET core 5

.NET Core er et open source rammeverk utviklet av Microsoft for utvikling av en hel rekke applikasjoner, dette være alt fra webapplikasjoner, konsollapplikasjoner til spill. Fordelen med .Net Core er at det fungerer på tvers av alle store operativsystemer som Windows, MacOs og Linux.

Beautify.js

Beautify.js er et JavaScript-bibliotek som korrigerer syntaks og tar inn mellomrom og andre elementer der det er naturlig for å visuelt utbedre kode.

Codemirror

Codemirror er et JavaScript-bibliotek som brukes til å implementere en kodeeditor på nettsider.

Entity Framework Core

EF er et verktøy utviklet av Microsoft for å automatisere aktiviteter med databaser. Fremfor å jobbe direkte med databasetabeller omgjør EF data fra databaser til objekter. Dette kalles object-relational mapping (O/RM). Dette verktøyet gjør i praksis at en kan lage C# objekter,

som brukere og deres relaterte informasjon, og EF vil oversette dette til et språk databasen forstår. I praksis gjør dette at utviklere kan jobbe med domenespesifikke modeller og data uten å forholde seg til databasens underliggende strukturer som tabeller og kolonner, noe som er en fordel som forenkler utviklingen.

GitHub

GitHub er en plattform for opplasting og versjonskontroll av kode. Brukes til å holde kontroll på ulike versjoner av et program og forenkler samarbeid når flere jobber med samme kode samtidig.

Jira

Jira er et program som gir mulighet for å holde oversikt over hva som skal gjøres og hvem som har ansvar og skal gjøre jobben. Samt å holde kontroll over hvor mye tid en bruker på de gitte oppgaver.

Microsoft Azure

Microsoft Azure er en skytjeneste som tilbyr en rekke tjenester relatert til hosting av webapplikasjoner. Tjenestene Azure tilbyr er for tallrike for denne listen og kan leses mer om på Azure sin hjemmeside (Microsoft Azure, 2021).

Phaser

Phaser er et JavaScript-rammeverk designet spesifikt for utvikling av todimensjonale nettbaserte spill, og inneholder mye av det en kan ønske seg av funksjonalitet til et slikt prosjekt. Rammeverket kan brukes til å lage alle typer spill en typisk finner på tidligere

flashbaserte nettsider som nitrome.com, så vel som mange mobilspill. I dette prosjektet brukes Phaser 3, som er den nyeste utgaven av Phaser. Utviklerene til Phaser har publisert dokumentasjon for alle innebygde metoder på deres egen nettside, og dette oppdateres hver gang en ny release kommer ut.

Pixilart.com

Pixilart.com er en online editor designet for å lage pikselkunst. Verktøyet gjør det enkelt å designe bilder og modeller piksel for piksel, ved hjelp av en rekke verktøy og farger, samt lagring i flere formater og redigering av tidligere prosjekter.

Postman

Postman er en applikasjon som brukes til å teste API-er. Applikasjonen legger til rette for å sende forespørsel til nettserveren ved å fylle inn hvilken metode som skal kalles og hva du ønsker å sende til metoden. Postman vil så presentere responsen fra API-et.

Tiled

Tiled er en gratis 2D level editor som hjelper deg til å utvikle komponenter til spillet ditt. Dens hovedfunksjon er å utvikle kart til rutebasert videospill i ulike former, men den støtter også gratis bildeplassering samt kraftige måter å kommentere nivået ditt med ekstra informasjon som brukes av spillet. Tiled fokuserer på generell fleksibilitet mens det prøver å holde seg intuitiv (Tiled, 2021).

Trello

Trello er en nettbasert tjeneste som brukes til å holde orden på hva som skal gjøres og hva som er blitt gjort i Kanban-stil.

3. Prosessdokumentasjon

Gamification av ansettelser

Oslomet – Gruppe 31 – vår 2021

Skrevet av

Christian Dyrli

Jørgen Borander

Maximilian Stiegler Sharoyan

Sondre Næbb Bjarkum

3.1 Forord

I dokumentasjonen under beskrives hvordan utviklingen av prosjektet er blitt gjennomført og prosessen bak utviklingen av produktet. Her gjennomgås hele arbeidet rundt ideen, problemene og løsningene som dukket opp underveis, samt dokumentasjon rundt utformingen av prototypen. Denne delen foregikk fra prosjektet startet i januar og frem til innlevering i mai.

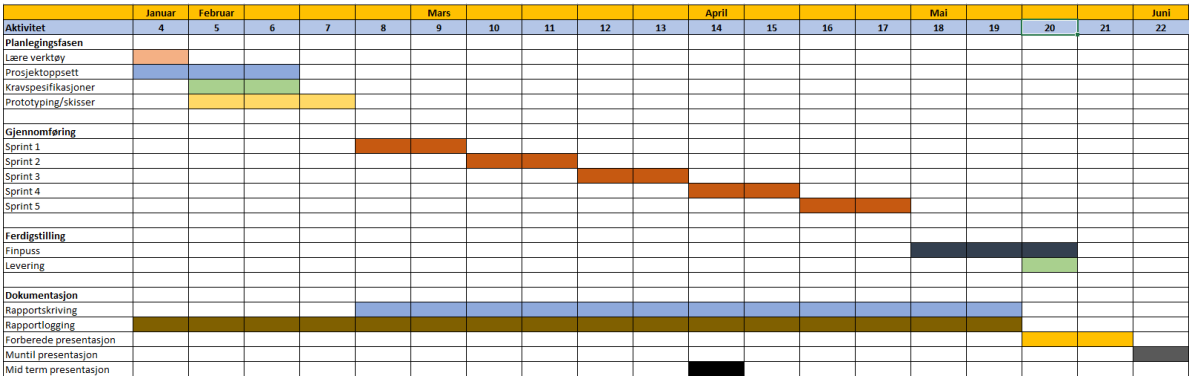
3.2 Metodikk og Planlegging

OXX har en smidig tilnærming til utvikling ved bruk av SCRUM. Dette er en veldig vanlig tilnærming å ha til et prosjekt og er brukt mye i næringslivet, spesielt i teknologiselskaper (Petrova, 2019). Derav var det naturlig at gruppen tok i bruk samme metode når oppgaven skulle løses. Ettersom SCRUM er en metodikk ofte brukt i utvikling, anser gruppen det som relevant og verdifull erfaring etter endt oppgave.

3.2.1 Fremgangsmåte

Gruppen og OXX ble enige om hva som skulle være satsningsområdet for den ferdige løsningen av prosjektet, og at det skulle jobbes med de mest essensielle kravene først. Dette for å få til et minimum fungerende løsning eller en HiFi-prototype. Det ble bestemt at de ulike deloppgavene skulle fordeles innad i gruppen, avhengig av hvem i gruppen som hadde sterkest kompetanse i de ulike feltene. På grunn av koronasituasjonen og høyt smittetrykk i Oslo, hadde ikke gruppen anledning til å møtes fysisk. Det kom da en enighet om at gruppen skulle møtes på teams hver dag klokken 10 for å diskutere gårsdagens arbeid, hva som skulle gjøres den kommende dagen og om en trengte hjelp videre.

3.2.2 Sprintplanlegging



Figur 4: Originale fremdriftsplan.

Figur 4 viser den originale fremdriftsplanen. Den opprinnelige planen var å ha fem sprinter, der oppgavene for de neste ukene skulle bli satt i forkant av hver sprint. For å holde styr over de ulike sprintene ble det først benyttet Jira. Grunnen for valget av Jira var at dette er programmet OXX benytter seg av i sin interne sprintplanlegging. Underveis i prosjektet ble Jira byttet ut med Trello. Dette fordi Jira opplevdes for komplisert i forhold til gruppens nytteverdi.

Gruppen valgte å bytte til Trello som organiseringsverktøy, da samtlige gruppemedlemmer hadde tidligere erfaring med applikasjonen. På denne måten slapp gruppen å bruke unødig tid til opplæring i nytt verktøy.

Som konsekvens av endringen i organiseringsverktøy og uforutsette utfordringer med nye teknologier, var det vanskelig å følge sprintplanen og backlog ble raskt utvidet. På bakgrunn av dette valgte gruppen å gå over til Kanban da dette også er metodikken Trello er designet for.

GitHub ble brukt som versjonshåndteringsverktøy for prosjektet. Her opprettet gruppen flere brancher underveis når ny funksjonalitet skulle implementeres. GitHub gir samtidig hvert gruppemedlem mulighet til å ha all kode tilgjengelig hele tiden.

3.2.3 Utfordringer underveis COVID-19

Koronasituasjonen i Norge medførte en del utfordringer i forbindelse med gjennomføringen av prosjektet. Måten gruppen jobbet på ble sterkt påvirket av nedstengningen i desember. Fysiske møter ble byttet ut med teams, og arbeid på kontoret til bedriften ble flyttet til hjemmekontor på hybelen. Dette senket hastigheten til prosjektet, da møter på nett medbringer kommunikasjonsutfordringer som ikke medfølger fysiske møter. I tillegg til dette fikk ikke gruppen veileder før midten av januar, som gikk ut over støtte under forberedelsene i høstsemesteret og informasjonsflyten mellom universitetet og studentene.

3.3 Utviklingsprosessen

I begynnelsen av prosjektet ble frontend av spilldelen prioritert, da gruppen anså dette som den viktigste delen. Dette ble gjort fordi det både var den mest utfordrende delen med tanke på ny teknologi, og fordi gruppen opplevde det som enklere å brukerteste. Med prototype av spilldelen ferdig, gir det testbrukere et klarere bilde av det tenkte sluttproduktet. Backend ble også jobbet med underveis, men prioritert lengere ut i prosjektet. Bakgrunnen for dette er at frontend måtte nå et visst nivå for å kunne implementere backendfunksjonalitet.

3.3.1 Tidlig planlegging

Tidlig i prosessen ble det diskutert med oppdragsgiver hvilke teknologier og verktøy som skulle brukes i prosjektet. Oppdragsgiver satte ingen krav, men fremmet ønske om at løsningen skulle være enklest mulig for OXX å kunne ta i bruk eller videreutvikle etter at prosjektet ble ferdigstilt. På dette grunnlaget var det enighet om å benytte rammeverket ASP.NET Core 5 med C#. Det ble videre utarbeidet nøyaktige og punktlige kravspesifikasjoner for å kunne utføre prosjektet på best mulig måte (se 6.Kravspesifikasjoner).

Etter samtaler og gjennomgangen av kravspesifikasjonene ble det bestemt at det skulle lages et spill med evnetester. Hvordan testene skulle uttrykkes, var opp til gruppen å finne ut av.

Det ble da satt av tid til en idemyldring. Det var ønsket at evnetestene skulle være en form for spillifisering, samtidig som de testet relevante kunnskaper av kandidatene.

Det ble fremmet fem forskjellige ideer. Disse var hovedideene før gruppen kom fram til en avgjørelse på hvordan gruppen ønsket de endelige testene skulle se ut.

- Et Tetrisspill for å teste kodeforståelse, hvor kodeord faller ned og må flyttes for å treffe på riktig sted i kodelinjen nederst.
- Shufflespill hvor det er flere bilder av kodesnutter som en skal sortere riktig ved å dra og dytte rundt på bilder for å gi et riktig totalt bilde som et puslespill.
- Ballongtest, en test hvor det er flere ballonger som ved en knapp fyller de med luft. Disse eksploderer hvis de fylles med for mye luft. I tillegg sprekker hver ballong ved ulike luftmengder.
- Flippkort spill der kode står på forskjellige kort hvor du snur to og to kort for å treffe samme kode eller kode som har samme output.
- En quiz som tester forskjellige aspekter som kodeforståelse, teorikunnskap om programmering og teknologier som er relevante for oppdragsgiver

Samtidig som dette ble diskutert, ble det vurdert hvordan alt dette skulle henge sammen. Skulle kandidaten komme til en nettside hvor en bare ble bedt om å ta tester, eller en mer konkret historie hvor en blir sendt rundt for å løse oppgaver for ulike personer på kontoret til OXX.

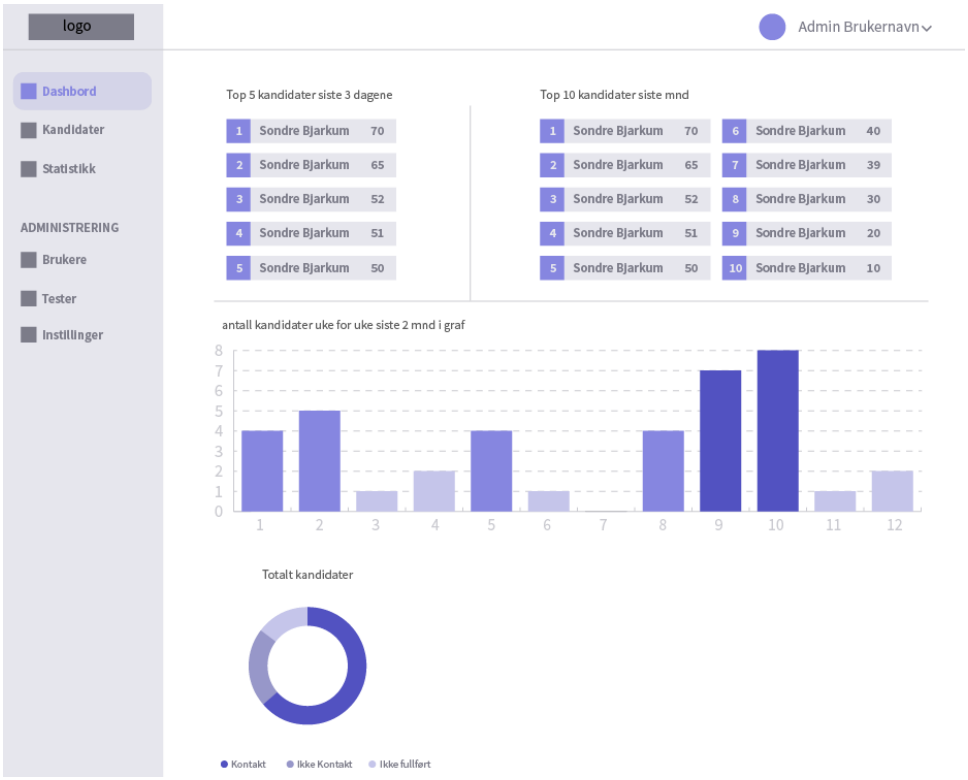
3.3.1.1 Skisser og moodboard

Før gruppen visste hvordan spillet skulle uttrykkes lagde gruppen et moodboard for å visualisere hvordan det var tiltenkt å se ut. Denne kom i form av en kollasj med bilder av spill i lignende stil som gruppen ønsket.

For å skape et visuelt inntrykk av spillet og administratorsidens design, lagde gruppen skisser. Disse skissene ble utformet i Adobe Illustrator. Gruppen var enige om å gjenskape den retro pikselart stilen fra spill som Pokemon og Zelda. Disse spillene er også i fugleperspektiv, slik gruppen så for seg prosjektet. Se Figur 5

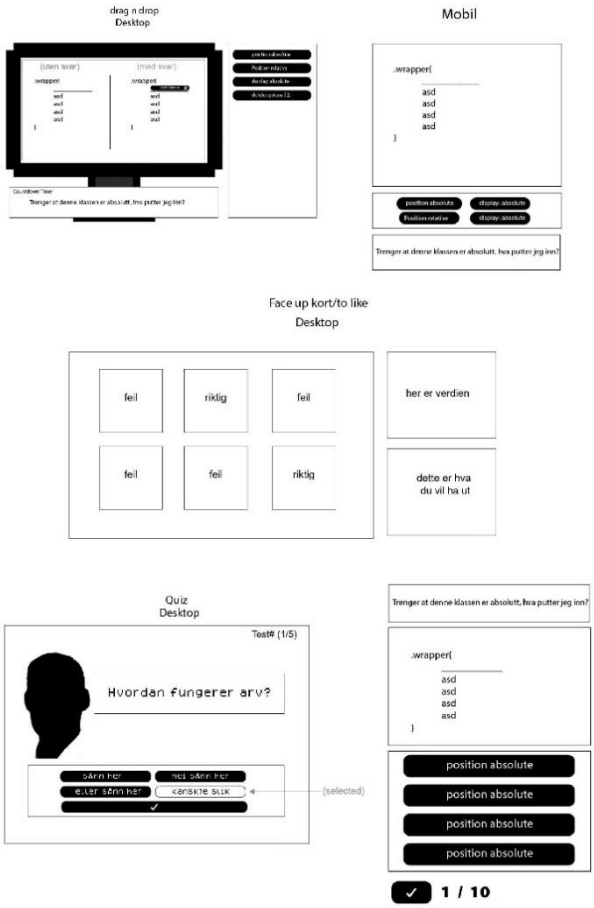
Figur 5: Skisse av spill i fugleperspektiv.

Administratorsiden skulle kunne visualisere dataene om tester og kandidater fra databasen. Det var også tiltenkt at den skulle vise statistikk over hvor mange som hadde gjennomført evnetestene innenfor en gitt periode (se Figur 6).



Figur 6: Skisse av administratorsiden.

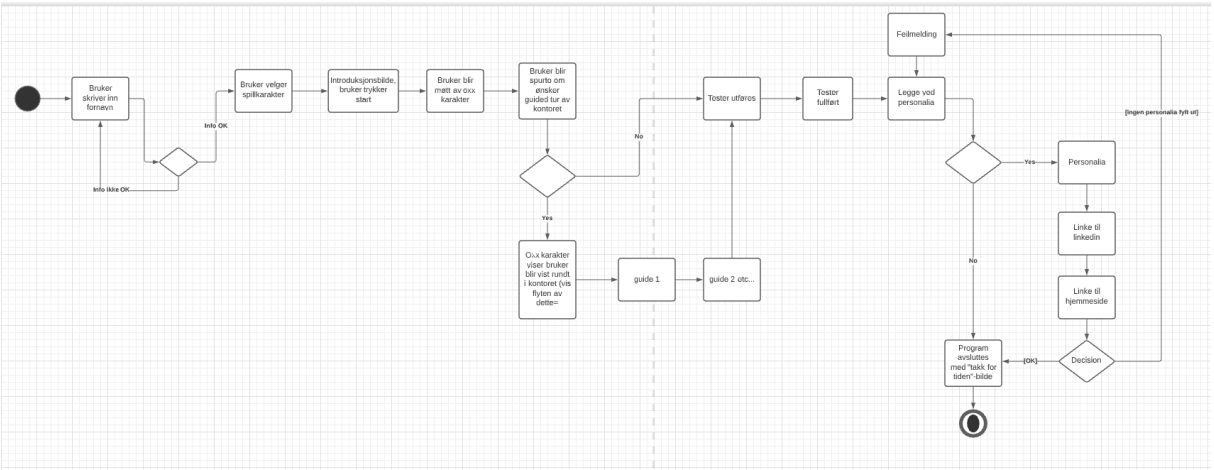
Skissene var et godt utgangspunkt for å begynne utvikling (se Figur 5 og Figur 6 og Figur 7).



Figur 7 Skisser av evnetester

3.3.1.2 Utvikling av flyten

Videre ble det diskutert innad i gruppen hvordan flyten av spillet skulle se ut. Det ble derfor tegnet opp et forslag til aktivitetsdiagram (se Figur 8) for å gi gruppen en pekepinn på hvordan flyten i spillet skulle ende opp med å se ut. Flyten endret seg noe utover oppgaven og endte med å se noe annerledes ut til slutt.



Figur 8: Skisse av aktivitetsdiagram.

Det ble tenkt at kandidaten begynte med å skrive navnet sitt inn slik at dette kunne bli brukt underveis i spillet, for å gi kandidaten en personifisert opplevelse under bruk. Det var også ønsket at bruker skulle få mulighet til å enten lage sin egen karakter, eller velge mellom et gitt utvalg karakterer. Dette for å få hele opplevelsen til å oppfattes som mer inkluderende og brukervennlig. Denne funksjonaliteten ble nedprioritert til fordel for viktigere aspekter av prosjektet.

Aktivitetsdiagrammet (se Figur 8) viser at brukerne ville bli møtt av en karakter som introduserer de for OXX og ønsker de velkommen. Karakteren skulle vise spilleren rundt på kontoret, før de ble videresendt til testene. Det ble derimot bestemt og heller la brukeren utforske kontoret på egenhånd. Å kunne snakke med de forskjellige karakterene var tenkt å skape større innlevelse under gjennomføringen, enn hvis brukeren ble geleidet gjennom hele prosessen.

Etter at alle evnetester er fullført, sendes bruker til et registreringsskjema for personalia, og deretter til en side som takker for oppmerksomheten.

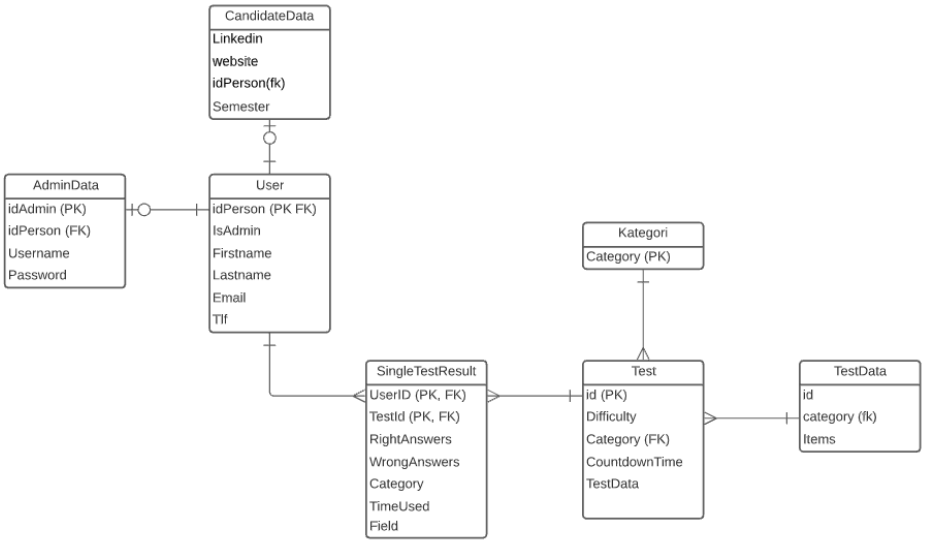
Underveis i prosessen gjennomførte gruppen en del endringer som devierer fra den originale flyten. Som nevnt tidligere ble det ikke prioritert at brukerne kunne legge inn navn eller bestemme figur. Spillet begynner istedenfor ved at du havner på en landingsside som forklarer hvordan du beveger deg i spillet. Når brukerne starter spillet, blir en tildelt en fast

figur og navn blir ikke nevnt i samtalene med de forskjellige figurene. Ved start blir en møtt av en karakter som ønsker velkommen og forteller litt om OXX. Videre herfra blir kandidaten bedt om å bevege seg videre til flere karakterer. Der blir kandidaten bedt om hjelp til å løse gitte problemer, før det avsluttes med at kandidaten dirigeres tilbake til første karakter for å registrere informasjon om seg selv.

Hvilken informasjon som skulle lagres om kandidatene ble det diskutert med OXX i forhold til deres behov. Første idé var at kandidatene skulle kunne laste opp CV i tillegg til navn, e-post, telefon, studieretning og hvilket semester de var på. Dette gikk gruppen vekk fra da løsningen ble planlagt benyttet på stands og med mobiltelefon, ettersom CV ofte ikke lett er tilgjengelig fra mobil. Gruppen konkluderte med at brukerne heller skulle ha mulighet til å legge ved LinkedIn-lenke til sin egen profil. På denne måten har OXX mulighet til å finne CVen til kandidater de ønsker å kontakte, via LinkedIn. Ettersom ikke alle har LinkedInprofil tenkte gruppen at informasjonen som navn, email etc. skulle registreres separat.

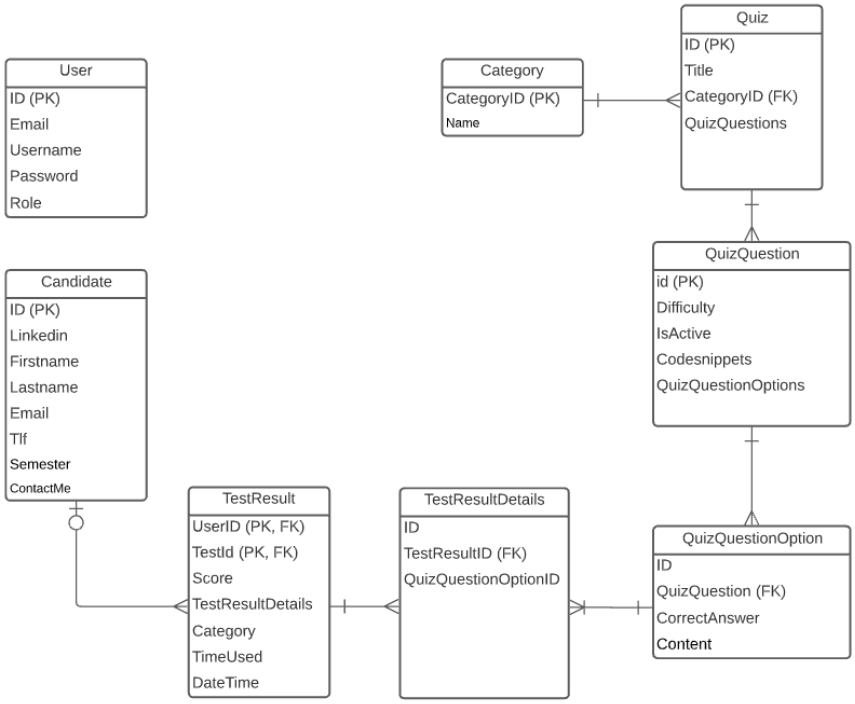
3.3.1.3 Utforming av databasen

Første steg i utvikling av backend var å utforme databasen og dens struktur. Dette ble gjort tidlig i utviklingen da databasen er en viktig del å få riktig tidlig, ettersom feil databasestruktur vil kunne by på utfordringer senere. På bakgrunn av dette satt gruppen stort fokus på at kravene som følger med tredje normalform var oppfylt. Lucidchart ble anvendt for å utforme databasemodellen og dens tabeller, hvor et tidlig utkast så slik ut (Figur 9)



Figur 9: Utkast databasestruktur.

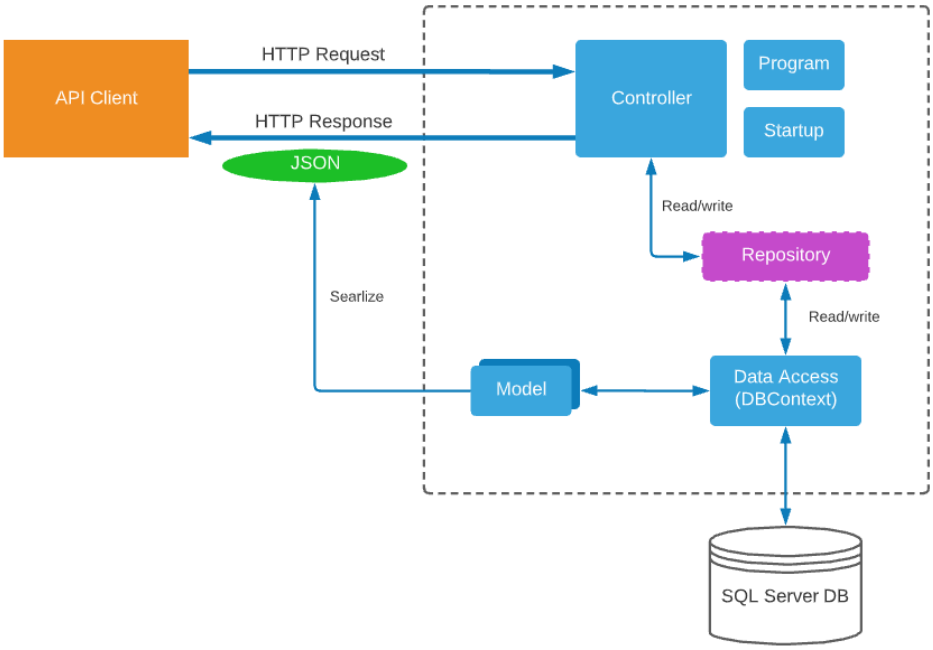
Derimot etter flere revisjoner og konklusjoner rundt hvilke teknologier vi skulle bruke, endret modellen seg til å se slik ut (Figur 10)



Figur 10: Endelig databasestruktur.

3.3.1.4 Struktur

Før majoriteten av planlegging og utforming av backend satt gruppen en plan for hvordan webapplikasjonen skulle struktureres og lagdeles i henhold til mapper og prosjekter. Øvre struktur i applikasjonen er delt opp i tre deler: API, Data og Domain (Figur 11). API inneholder controllere og grensesnittet for sluttbruker og fungerer som et bindeledd mot databasen. Under Data ligger logikken til controllerne samt Context for databasen, og i Domain ligger modellklassene som refereres av Data. Alle disse leddene snakker sammen for å danne den fulle applikasjonen.



Figur 11: Webapplikasjonstruktur.

3.3.2 Spillet

Figur 12: Skissert pixelmann med animasjonssteg.

Pikselkartet og pikselfigurene er laget ved bruk av pixilart.com. Først skisserte gruppen en pikselmann og satte opp figuren i alle steg av bevegelsesanimasjonen (Figur 12), slik at den kunne brukes i første testing av bevegelsesfunksjonaliteten.

Deretter designet gruppen et førsteutkast av kartet (Figur 13). Inspirasjonen til designet er hentet fra plowgames.com (Plow Games, 2021) sin retrotour og Oxx sin egen promovideo, som viser deler av kontorlokalene deres i bakgrunnen (OXX, 2021). Videre designet gruppen flere karakterer bruker kan interagere med.

Figur 13: Skisser pixelart

Den neste delen av prosjektet som ble ferdigstilt var kartet, som er omgivelsene kandidatens karakter befinner seg i. Det oppsto her et problem med hvordan karakteren beveget seg, og

interagerte med omgivelsene. Det var ønsket at karakteren skulle bevege seg på en mer naturlig måte, og at figuren skulle kollidere med andre objekter på kartet. Dette ble løst ved å gjøre nødvendige endringer i Phaser.

På dette tidspunktet i utviklingen fokuserte gruppen på optimalisering mot mobile enheter. Løsningen så langt skalerte ikke riktig og brukeren manglet muligheten til å bevege seg rundt på kartet og starte evnetester. Denne oppgaven medførte mange utfordringer, da det var tidkrevende og vanskelig å implementere. Det ble implementert en joystick, og en knapp. Dette med hensikt å gi brukere på mobil mulighet til å bevege seg og aksessere evnetestene.

Knappen, når benyttet nær enkelte karakterer, henter HTML for enten evnetester eller registreringsskjema og presenterer dette for brukeren. Hvilken HTML som hentes ut baserer seg på hvilket punkt i spillet bruker har nådd.

I felleskap med OXX ble diskusjonen rundt antall tester konkludert med at tre evnetester ville være tilstrekkelig, der hver evnetest tester ulike aspekter og kunnskap om utvikling. Det var viktig at gjennomføringen, fra start til slutt, ikke varte lenger enn 10 minutter. Dette begrunnes med applikasjonens fokus på stands, som medfører en begrensning av hvor mye tid kandidater kan sette av til å gjennomføre tester.

3.3.2.1 Valg av evnetest

Under punkt 3.3.1 er fem ulike muligheter for å teste ferdighetene til kandidatene listet opp. Etter flere samtaler med OXX kom det til enighet at evnetestene skulle tilrettelegges for å skille en førsteklassestudent fra en tredjeklassestudent. Ballongtesten ble derfor fjernet da denne ikke kunne skille forskjellige kandidater ferdigheter inn utvikling.

Originalt var planen at flere typer evnetester skulle implementeres. Gruppen innså derimot at den gjenværende tiden ikke ville være tilstrekkelig til å implementere dette på en god måte. Gruppen kom derfor til enighet om å velge en type test, fleksibel nok til å brukes flere ganger med forskjellige kategorier.

Basert på alternativene under punkt 3.3.1 konkluderte gruppen med at en quizløsning derfor var best egnet. Dette ble begrunnet med at en quiz enkelt kan tilpasses og oppdateres etter behov. Det var også foretrukket av oppdragsgiver å ha mulighet til å kunne gjøre dette.

Med utgangspunkt i pensum fra IT-studiene ved OsloMet ble kategoriene som følger: kodeforståelse, programmeringsteori og systemutviklingsteori. Disse tre kategoriene dekker et bredt kunnskapsgrunnlag hos en IT-student, og er et godt utgangspunkt for å skille studenters kunnskapsnivå.

For å skille kandidater som har fullført evnetester besluttet gruppen å beregne en poengsum. Denne poengsummen tar utgangspunkt i antall riktige svar, veier dette opp mot totalt antall spørsmål og regner ut en prosentandel av dette. Som et supplement til dette ble det implementert en stoppeklokke som tar tiden fra kandidat begynner en evnetest til evnetesten er over. Dette gir oppdragsgiver mulighet til å skille kandidater på mer enn kun poengsum og gir derav et mer helhetlig resultat. Kandidaten kan bruke så lang tid en vil, men tiden en bruker vil gi en indikasjon på hvor godt en kandidat kjenner stoffet.

3.3.2.2 Utforming av administratorside

OXX uttrykte et ønske om en administratorside. Denne siden skal kunne liste ut kandidatene og filtrere disse etter poengsum eller tidsbruk, se detaljer personalia til kandidater og kunne liste de 10 beste kandidatene hver måned og de 10 beste totalt. De ønsket også muligheten til å fjerne kandidater hvis de har blitt intervjuet og kunne se, endre og slette spørsmål fra de forskjellige testene.

Under utvikling ble prioriteringene å få inn de mest essensielle funksjonene. Dette ble konkludert med å være muligheten til å kunne se detaljert personalia, og liste ut de 10 beste kandidatene fra foregående måned.

3.3.3 API-et

Denne seksjonen krever god kunnskap rundt databaseoppsett og backendprogrammering, for å få verdi fra lesningen. Dersom ord eller uttrykk er ukjent, anbefales det å se beskrivelse under kapittel 2.Teknologier og Verktøy, eller under kapittel 10.Ordliste.

3.3.3.1 Databasen

MSSQL server ble valgt som databasemodell, ettersom denne er godt dokumentert. For å lese og lagre data til databasen besluttet gruppen å bruke Entity Framework Core (heretter EF Core). Alternativet til å bruke EF Core ville vært å skrive SQL og lage relasjoner mellom tabellene manuelt. For å effektivisere utviklingen tok gruppen derfor i bruk EF Core. Derimot var ikke gruppen altfor godt kjent med hvordan rammeverket brukes riktig. Etter samtale med ekstern veileder ble gruppen referert til et videokurs (Lerman, 2020) som ble grunnlaget for oppbygningen av prosjektet samt oppsett av EF Core og bruk deretter.

3.3.3.1.1 Enitity Framework Core

For å ta i bruk EF Core lagde gruppen modellklasser med utgangspunkt i modellen fra 3.3.1.3 som gjenspeiler tabeller i databasen. Under utvikling av dette støtte gruppen på flere utfordringer i henhold til relasjoner mellom tabellene. Mer spesifikt mellom testresultatene og svaralternativer. Det viste seg at EF Core lagret fremmednøkler for å danne relasjonene, men at disse fremmednøklene skapte en nærmest endeløs løkke av referanser. Dette ble løst ved å lage tabellen TestResultDetails som brøt opp mange-til-mange relasjonen som EF Core lagde. En annen utfordring gruppen støtte på, var at ved forsøk på sletting av en kandidat så kastet EF Core et unntak. Dette ble senere rettet ved å definere at fremmednøkkelen opprettet av TestResultDetail i QuizQuestionOption skulle settes til nullverdi.

Under utvikling ble modellene også justert en del. Attributter ble endret og nye løsninger måtte implementeres. Blant annet ble det opprettet hjelpeklasser for å sende data mellom frontend og backend, hvor disse brukes til å lage nye objekter av gitt type. Dette måtte implementeres ettersom ASP.NET Core 5 ikke klarer å oversette JavaScript-objekter til komplekse C# objekter, mer spesifikt modellklassene.

3.3.3.1.2 DbContext

Når modellene er laget, må disse gis kontekst. Med dette menes det at man instansierer modellklassene i en contekstfil og setter disse til å representere tabeller i databasen. Med DbContext-klassen tillater man å gjøre transaksjoner og spørringer til databasen (EntityFrameworkTutorial, 2020). I dette tilfelle laget gruppen GamificationContext.cs. Her oversetter programmet modellene til tabeller med metoden DbSet, definerer noen egendefinerte relasjoner og injiserer en superbruker til databasen. EF Core ordner resten bak panseret ved å generere SQL-setninger.

3.3.3.1.3 Lazy Loading

For å enklest mulig hente data fra databasen ble lazy loading tatt i bruk. Dette gjør at alle data som henger sammen blir hentet ut. Dette er riktignok ikke optimalt for release (Wildermuth, 2018). Gruppen brukte dette for enkelhetens skyld, men gruppen kan da ikke definere nøyaktig hvilke data som returneres fra controller. Med dette viser det seg at attributten, som definerer om et svaralternativ er riktig eller galt også blir returnert ved spørring. Ettersom dette er en prototype er ikke dette kritisk, men ved release ville man heller brukt eager loading, definert med .Include() hvilke datafelt som skulle blitt returnert og derav utelatt denne attributten.

3.3.3.2 Utforming av API

3.3.3.2.1 Controllere

Så fort databasen og dens modeller var fastsatt, begynte gruppen å utvikle API-leddet. Det første som ble planlagt var controllerne. Hvordan controllere webapplikasjonen trengte og hvilke metoder og endepunkter som skulle være med i disse måtte defineres. Gruppen visste at det kom til å ta i bruk CRUD-modellen (Create, Read, Update and Delete) som utgangspunkt for utforming av API-en. Derimot ønsket gruppen å opprette metoder og endepunkter utover CRUD operasjoner, slik som å hente ut top 10 kandidater den siste måneden. For å planlegge dette brukte gruppen kravspesifikasjonene (Kapittel 6) og noterte hvilke metoder de forskjellige kontrollerne kom til å trenge utover CRUD operasjonene.

Kontrollerne ble deretter opprettet basert på modellenes kategorier, som i praksis betyr mappene modellene ligger i. Som et utgangspunkt ble egendefinerte controllere opprettet, via en innebygd funksjon i Visual Studio som genererer CRUD-metodene automatisk. Dette var et godt utgangspunkt for å utvikle egne metoder.

Utviklingen av egendefinerte metoder i kontrollerne gikk henholdsvis smertefritt. Ettersom gruppen planla hva metodene skulle gjøre, sto det mest på å få logikken for disse på plass. PostMan kom godt med her. Av mer utfordrende integrasjoner var controlleren for autorisering. Gruppen undersøkte hvordan en slik løsning kunne implementeres og landet på en veiledning fra c-sharpcorner (Saseendran, 2020). Ettersom logikken for å ta i bruk Microsoft Identity er såpas avansert, valgte gruppen å implementere denne løsningen. Dette utdypes i punkt 3.3.3.2.3.

3.3.3.2.2 Lagdeling

Når controllerne var på plass var det ønskelig å lagdele de av sikkerhetsmessige årsaker (Walpita, 2019). For å flytte logikk ut av controlleren og inn i et eget repository ble det implementert et interface. Både QuizsController og UsersController ble lagdelt, derimot AuthenticateController ble det avgjort å la være ettersom koden viste seg å være utfordrende å flytte over i et egen repository.

3.3.3.2.3 Autorisering

Som nevnt i punkt 3.3.3.2.1 ble det implementert en løsning for autorisering. Prosessen for å implementere autorisering var til dels utfordrende og komplisert. Forskjellige kilder har veldig forskjellige meninger om hva som utgjør en sikker og gjennomført autorisering (Degges, 2017). Til slutt endte gruppen opp med en løsning som ikke er hundre prosent robust, men som fungerer. Utfordringen med hva som er implementert er at JWT Token, som blir returnert etter vellykket innlogging, blir lagret i session i nettleseren og vil kunne være sårbar mot csrf-angrep (Hasan, Anderson, & Smith, 2019) (Copes, 2018). En alternativ

løsning som ble vurdert var å opprette sessions i selve applikasjonen ved bruk av Microsoft.Sessions rammeverket, men dette ble prioritert bort på grunn av tidsbegrensninger.

Det ble besluttet å ta i bruk en kombinasjon av rammeverket Microsoft.Identity for identifikasjon og brukerroller, og middlewaren Authentication.JwtBearer for tokens. Kombinasjonen av disse gjorde at gruppen slapp å definere egne modeller for brukere, deres innloggingsfunksjonalitet og opprettelse av tokens med tidsavgrensing. Ved å implementere disse ville det meste gjøre seg selv automatisk, som hashing av brukers passord og opprettelse av tokens.

For å sikre mot uautorisert tilgang til gitte endepunkter i API-leddet brukes et attributt kalt [Authorize] over funksjonene som skal beskyttes. I disse defineres hvilken rolle en bruker må inneha for å få tilgang. Gruppen definerte to roller som er gjennomgående brukt gjennom applikasjonen: Super og Admin. Det ble også opprettet en tredje bruker som er tiltenkt en standardbruker, men er ikke blitt brukt. Denne er tiltenkt for videreutvikling om det skulle bli aktuelt å for eksempel lage en personlig brukerside for kandidater.

3.3.4 Integrering

Frontend og backend ble som vist over utviklet separat. Det som deretter gjensto, var å integrere disse delene til ett og samme prosjekt. For å gjøre dette besluttet gruppen å møtes fysisk for enklere samarbeid. Denne prosessen gikk smertefritt for seg. Gruppen hadde konsekvent laget forskjellige grener i GitHub, og ved integrasjon ble de forskjellige grenene merget til Main.

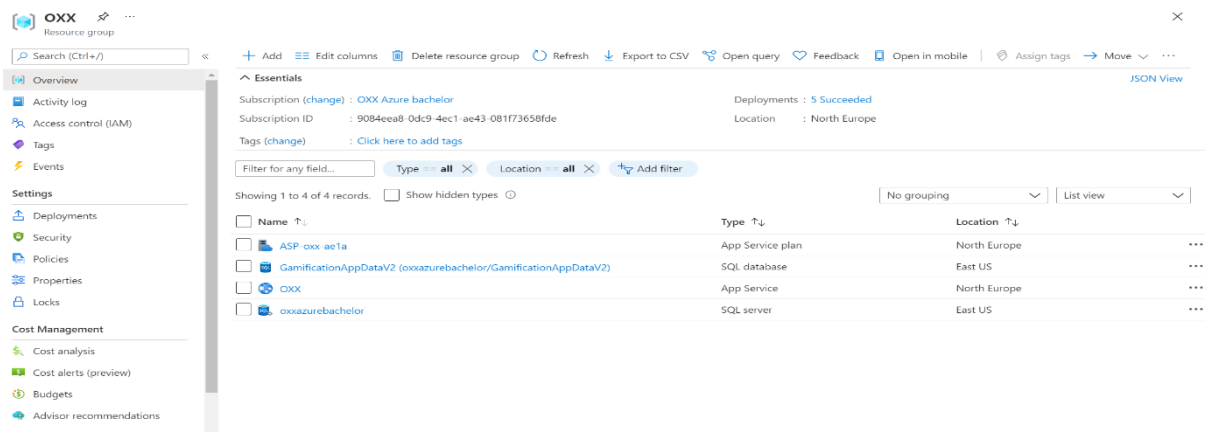
Deretter måtte javascriptet som driver evnetestene videreutvikles. For å hente quizer under utviklingen var det blitt brukt et tredjeparts API (Open Trivia, 2021). Javascriptet måtte da endres til å hente fra gruppens eget API. Ettersom JSON-dataen fra gruppens API var bygd opp annerledes enn tredjeparten sin, måtte koden som omgjør JSON-data til HTML skrives om for å ta hensyn til andre relasjoner og annen oppbygging.

For autorisering og loginsiden var det nødvendig å sette opp ett kall til API-et som tar imot brukernavn og passord. Returnert token ble lagt til i session, og brukt i foregående forespørsler som krevde autorisering.

3.3.5 Hosting i Azure

For at oppdragsgiver skulle kunne benytte seg av løsningen var et av kravspesifikasjonene med høy prioritering at løsningen skulle være kompatibel med å kunne hostes i Azure (Figur 14). Dette ble lagt til rette for, i tilfellet OXX ønsket å videreutvikle produktet. Azure ble valgt som plattform ettersom OXX allerede benytter seg av denne teknologien i sin utvikling. Derav kunne OXX også bistå med veiledning, noe gruppen benyttet seg av.

For å kunne hoste gjennom Azure måtte det lages en konto og en resource group. Alle medlemmene av gruppen ble invitert til denne gruppen, og utdelt autorisasjoner for å kunne publishe. Det ble opprettet en App Service hvor selve applikasjonen ble tjent.



Figur 14: Portalvinduet i Azure.

Det neste steget var å lage en database. Men for å kunne lage en enkelt database måtte det opprettes en egen SQL Server hvor databasen skulle ligge.

Disse ble laget ved å klikke på "Add" i Azure menyen inne på resource group panelet. Da SQL serverne på Azure starter med å være ip beskyttet, brukte gruppen et program som heter Microsoft SQL Server Management for å sjekke og legge til en IP-adresse. Videre ble det

brukt en undermeny inne på Azure SQL server “Firewalls and Virtual Networks”, for å hviteliste IP-adresser slik at alle i gruppen skulle få mulighet til å publishe og aksessere applikasjonen.

Microsoft SQL server management ble også benyttet for å sjekke databasens tabeller og innhold. Dette gjorde at gruppen kunne sikre at innholdet fra databasen i prosjektet ble lastet opp i Azure og lagret.

Azure medførte også utfordringer og frustrasjon. Ettersom gruppen aldri hadde brukt Azure i løpet av studiet, var det forvirrende å finne frem og forstå hva vi trengte for prosjektet. Problemene gruppen støtte på var som følger:

- Å gi alle gruppemedlemmer tilgang til å kunne publisere.
- Lage kostnadsbudsjett for å begrense utgifter.
- Åpne opp tillatelser for hosting fra forskjellige steder, hviteliste forskjellige IP adresser.
- Autorisering mellom Azure og ASP.NET Core.
- Skrive til databasen opprettet i Azure.

Som nevnt tidligere, ble disse problemene løst ved hjelp av ekstern veileder fra OXX.

3.3.6 Brukertesting av prototype

Det var ønsket fra alle parter at applikasjonen skulle ha brukertesting underveis i utviklingen. Det ble diskutert hvordan en på best mulig måte kunne gjennomføre testene, da den vanlige måten gruppen hadde testet på tidligere ikke lenger var mulig på grunn av Covid-19.

Testmetoden gruppen kom frem at til var best egnet, ble å hoste applikasjonen online og deretter sende lenke til testbrukere. Ved å klikke på lenken ble brukerne sendt til en forside med forklaring av brukertesten, samt hva de burde være oppmerksomme på underveis. Etter gjennomført test ble brukere sendt videre til et Google Docs dokument, for å svare på

spørsmål relatert til hvordan de opplevde applikasjonen. Kapittel 11.Appendiks inneholder spørsmål brukt under brukertestene.

3.3.6.1 Første brukertest

Den første brukertestingen ble bestemt at skulle teste de enkleste delene av spillet. Altså muligheten til å bevege seg rundt, hvordan videresendingen til en ny fane blir gjort og hvordan brukerne opplevede gjennomføringen.

Testen ble sendt til 10 testdeltagere, alle studenter i en aldersgruppe mellom 18-30 år. Alle ble tilsendt lenke på Facebook Messenger, og ble spurt om de kunne tenke seg å delta. Alle tilbakemeldingene var anonyme.

Før selve testen ble sendt, brukte gruppen mye tid å sørge for at alt skulle fungere, og på å fjerne potensielle problemer som kunne oppstå. Det ble antatt at tilbakemeldingene ville hovedsakelig være positive, da gruppen ikke støtet på noen problemer selv og at tilbakemeldinger om feil ville være minimale.

Det viste seg derimot å være feil, da 60% av tilbakemeldingene rapporterte problemer. Gruppen hadde feilet i å forutse problemene som kunne oppstå, som følge av den store variasjonen av både hardware og software. Forskjellige telefoner og nettlesere forårsaket at forklarende tekst fra snakkebobler underveis ikke ble visst, i tillegg til at streker fra snakkeboblene til hjørnet av skjermen gjorde det vanskelig å se deler av spillet.

Først ble dette sett på som en litt mislykket brukertesting, før gruppen innså at dette er nettopp hensikten med å gjennomføre brukertesting. Nemlig å finne feil slik at disse kan lukes ut og utbedre applikasjonen.

3.3.6.1.1 Videreføring til neste brukertest

Tilbakemeldinger fra testerne antydet at brukertesten var vellykket. Dette er noe som blir tatt videre med til de neste brukertestene.

I tillegg tar gruppen med seg videre det store mangfoldet av enheter og nettlesere en bruker kan benytte seg av. Det ble derfor viktig å teste så mange nettlesere og enheter som mulig, for å finne feil relatert til dette.

3.3.6.2 Andre brukertest

Målet med andre brukertest, var å teste den ferdige historien, hvordan overgang til nye faner opplevdes og finne generelle feil med selve spillet. Brukertesten ble sendt til 10 kandidater som testet løsningen. Kandidatene er alle i en aldersgruppe mellom 18-30 år og er studenter. Dette fordi den endelige målgruppen som skal benytte løsningen vil være i dette aldersspennet og vil hovedsakelig være studenter.

Det ble valgt å utvide antall spørsmål fra tidligere gjennomførte brukertester, for å få mer detaljer rundt hvilke enheter og nettleser som ble benyttet og hvorvidt kandidatene brukte en reklameblokker. Dette ble gjort for å avdekke en sammenheng mellom enhet, nettleser, reklameblokker og problemene som tidligere oppsto. Det visste seg at reklameblokker i noen tilfeller hindrer Phaser i å laste inn visse moduler riktig, mer spesifikt snakkeboblene til karakterer. Det er et gjennomgående problem i Phaser og er en feil kildekoden til Phaser (Molchan, 2021). Løsning på dette problemet vil være å oppfordre brukere til å skru av reklameblokker.

Andre brukertest var mer vellykket enn første. Kandidatene opplevde få ødeleggende problemer. Én kandidat opplevde at tekstboblene forsvant fra personene den skulle snakke med, et problem som også oppsto tidligere i første brukertest. Høyst sannsynlig var dette på grunn av en reklameblokker. I tillegg opplevde tre andre kandidater den veiledende teksten i tekstboblene som misvisende da de ble bedt å “trykke på Space eller knappen på skjermen”. Benyttet testerne seg av pc ville de ikke få opp skjermknappen ettersom denne er kun er synlig ved bruk av mobiltelefon.

Applikasjonen er tiltenkt for bruk på mobil, men nærmere 66,7% av testbrukerne benyttet seg av datamaskin fremfor mobil. Dette var ikke overraskende ettersom det ikke ble gjort

rede for at testbrukerne helst skulle benytte seg av mobil. Neste brukertest tok dette i betraktning for å sette søkelys på feil og problemer som kan oppstå ved bruk av mobiltelefon.

3.3.6.2.1 Videreføring til neste brukertest

Et av spørsmålene i brukertesten definerte en skala i spørsmålet, men selve skalaen under ble rangert i motsatt rekkefølge. I forklaringsteksten sto det at 5 var mest oversiktlig, mens på skalaen sto det at 5 var minst oversiktlig.

En tilbakemelding om brukertesten sa at skalaer burde være standardisert. Med dette menes at dersom en skala rangerer fra 1-10, så burde samtlige skalaer gjøre dette. I tillegg ble det påpekt at spørsmål burde ha et "vet ikke", eller "ønsker ikke svare" alternativ der dette er relevant.

3.3.6.3 Tredje brukertest

Tredje brukertest ble relativt lik som andre brukertest, bortsett fra at det ble endret hva brukeren skal bruke for å teste. Tidligere har det vært opp til hver enkelt bruker hva de ønsker å benytte for å teste. Selve gjennomføringen av testen var helt lik som andre brukertest, da denne er tilsvarende den tenkte endelige prototypen. Den eneste tiltenkte endringen for den ferdige prototypen vil være at testene er implementert i stedet for de sorte pauseskjermene testerne blir sendt til. Etter tilbakemelding fra enkelte testere om at skalaene i spørsmålsdelen burde være uniforme, ble det valgt å endre disse. Tidligere var det en kombinasjon av 1-10 og 1-5 vurderinger. Disse ble endret til å alle være 1-10 skalaer for samtlige spørsmål med skala som svarmetode.

Brukertesten ble utført med 10 personer, av disse var det en bruker som opplevde noen problemer med testingen, problemet som oppsto var at deler av teksten forsvant og gjorde det vanskelig å gjennomføre testen og få med seg hva poenget med brukertesten var. Dette var et kjent problem som er blitt tidligere opplevd av brukere som benytter seg av reklameblokker-tjenester. Denne brukeren mente at reklameblokkeren ikke var blitt

benyttet. Men det ble benyttet en eldre Android mobil med Chrome nettleser. Da dette er en eldre mobil mistenkes det at det kan være andre faktorer som har forårsaket denne feilen.

3.3.6.4 Planlagt brukertest - Testing av hypotese

Gruppen ønsket å teste om kandidater som gjennomgikk brukertest med spillet mellom hver test, ville prestere bedre enn kandidater som gjennomgikk kun testene, rett etter hverandre.

En av hensiktene med spillifisering, er å stimulere motivasjon og fokus ved å gjøre oppgavene mer underholdene for kandidater. (White, 2015) Derfor ønsket gruppen å undersøke om forskjellen på resultatene ville støtte opp dette. I tillegg til quizresultatene, ønsket gruppen å se om tilbakemeldingene fra brukertesterne som gjennomførte den spillifiserte brukertesten, ville være klart mer positive, enn tilbakemeldinger fra brukertesterne som bare tok quizer.

Den planlagte fremgangsmåten gikk ut på å dele brukertesten i to versjoner. Den ene versjonen ville ha den fulle opplevelsen, der brukeren ville navigere gjennom kartet mellom evnetestene og oppleve interaksjonen med karakterene rundt på kontoret. Den andre versjonen var planlagt å kun være evnetestene etter hverandre.

I etterkant ville spørreskjemaet inneholde spørsmål fokusert rundt brukerens opplevelse på temaer som fokus, underholdning og interesse.

På denne måten kunne gruppen fått en indikasjon på om produktet gir resultatene gruppen har siktet mot underveis, både med tanke på testresultater og brukeropplevelse.

På toppen av dette ville denne brukertesten gi gruppen en tilbakemelding angående hvorvidt applikasjonen håndterer å ta imot, og lagre registrert data fra flere kandidater.

4. Produktdokumentasjon

Gamification av ansettelser

Oslomet – Gruppe 31 – vår 2021

Skrevet av

Christian Dyrli

Jørgen Borander

Maximilian Stiegler Sharoyan

Sondre Næbb Bjarkum

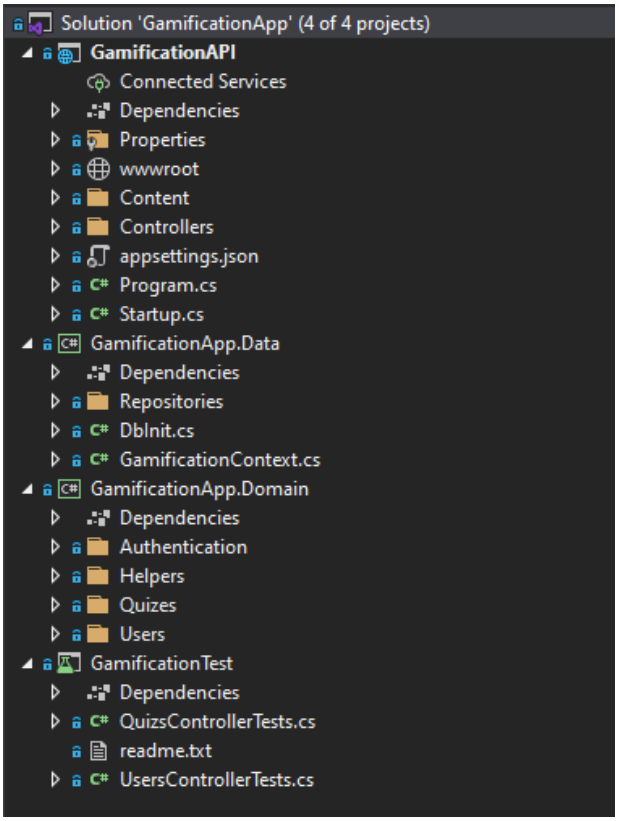
4.1 Forord

Den neste delen av rapporten har til hensikt å gi leseren et teknisk innblikk i det utviklede produktet. Det er anbefalt å lese tidligere deler før denne lesningen. Teksten inneholder en del ord og utrykk som kan være vanskelig å forstå omen ikke har IT-kompetanse. Alle ord og utrykk som kan oppleves fremmed vil være skrevet i punkt 2. Teknologier og Verktøy samt punkt 10. Ordliste i rapporten. Denne delen er laget for de som ønsker å sette seg dypere inn i prosjektet eller som skal videreutvikle løsningen.

4.2 Beskrivelse av applikasjonen

4.2.1 Prosjekt- og mappestruktur

Prosjektet er lagdelt etter “Three Layering Archtecture” (Pal, 2013). Disse tre lagene er API, Data og Domain. API-laget snakker med Data som igjen snakker med Domain. Denne lagdelingen gjøres for både sikkerhetshensyn, å sikre at sluttbruker ikke har tilgang til logikken i backend, og for å ha en struktur for utvikling, se Figur 15.



Figur 15 Mappestruktur

4.3 Frontend

Prosjektets frontend er en interaktiv webapplikasjon i form av et spill i fugleperspektiv. Hovedformålet med denne webapplikasjonen var å utvikle et brukergrensesnitt som kan teste og finne nye mulige kandidater på en spillifisert måte som er både morsom og engasjerende for brukeren. Brukergrensesnittet må kunne kommunisere med applikasjonens backend for å motta og sende data, og det skal være lett å ta opp og bruke for en førstegangsbruker.

4.3.1 Spillet

Brukergrensesnittet til webapplikasjonen består av flere deler. Selve spillet, evnetestene, personaliaskjema og administratorside. Spillet ble utviklet i Phaser. Som nevnt tidligere er det et Javascript-rammeverk laget for utvikling av web baserte spill.

4.3.1.1 Main.js

```
/** @type {import("../typings/phaser")} */

import { LoadScene } from './scenes/LoadScene.js';
import { MenuScene } from './scenes/MenuScene.js';
import { PlayScene } from './scenes/PlayScene.js';

var config = {
    type: Phaser.AUTO,
    width: 800,
    height: 800,
    backgroundColor: "#111111",
    parent: 'phaser-example',
    dom: {
        createContainer: true
    },
    scene: [LoadScene, MenuScene, PlayScene],
    render: {
        pixelArt: true
    },
    physics: {
        default: "arcade",
        arcade: {
            debug: false
        }
    },
    scale: {
        autoCenter: Phaser.Scale.CENTER_BOTH,
        mode: Phaser.Scale.RESIZE
    }
};

var game = new Phaser.Game(config);
```

Figur 16: Scriptet som står for opprettelse av Phaserprosjekt.

Main.js er scriptet som oppretter selve Phaser prosjektet (Figur 16). Den gjør dette ved hjelp av en konfigurasjonsfil. I konfigurasjonsfilen deklareres de ulike parameterne som danner

grunnlaget prosjektet er bygget opp på. Her blir det deklarert og gitt prosjektet tilgang til de ulike scenene som brukes. I konfigurasjonsfilen deklareres også hvilken plugin prosjektet skal bruke for håndtering av fysikk i spillet.

4.3.1.2 CST.js

```
export const CST = {
    SCENES: {
        LOAD: "LOAD",
        MENU: "MENU",
        PLAY: "PLAY"
    },
    IMAGE: {
        LOGO: "Logo.png",
        PLAY: "play_button.png",
        TERRAIN: "oxxKontor.v3.png",
        TEXT: "enDag.png",
        QUIZ: "testSporsmal.png"
    },
    SPRITE: {
        PERSON: "person2.png",
        NPC: "npc.png",
        NPC2: "walking.png",
        NPC3: "person3.png",
        NPC4: "person4.png",
        NPC5: "person5.png"
    }
}
```

Figur 17: Opprettelse av CST.Js.

Det opprettes en CST.js (Figur 17) Javascript fil som blir brukt til å holde oversikt over alle «key's» til scene, bilder og sprites.

4.3.1.3 Scenes

Spillet er bygget opp av scenes. Dette prosjektet består av 3 ulike scenes: LoadScene, MenuScene og PlayScene.

LoadScene er den første scenen som prosjektet tar i bruk. Den laster inn og gir Phaser tilgang til alle nødvendige filer, som for eksempel bilder og sprites fra assetsmappen i prosjektet. De lastes inn og gir nøkkel til CST slik at de blir lett tilgjengelige fra andre scenes i prosjektet.

MenuScene blir startsiden til spillet. Etter at alle filer er lastet inn i LoadScene går programmet videre til MenuScene. MenuScene lastes ikke inn før LoadScene sine oppgaver er fullført. Dette sikrer at alle filer er klare før spillet starter.

PlayScene er scenen spillet kjøres fra, ved bruk av Phaser sine innebygde metoder og klasser samt elementene lastet inn i LoadScene. Gjennomgangen av PlayScene vil kun dekke det som er nødvendig for å få oversikt over hvordan spillet er fungerer. Dette grunnet størrelsen til PlayScene.

Det første som skjer i scenen, er at alle de ulike objektene blir opprettet.

```
//laster inn personene
this.person = this.physics.add.sprite(620, 70, CST.SPRITE.PERSON).setScale(2).setDepth(-1);

this.npc1 = this.physics.add.sprite(450, 60, CST.SPRITE.NPC2, 0).setScale(2).setImmovable(true).setDepth(-2);

this.npc2 = this.physics.add.sprite(75, 65, CST.SPRITE.NPC3, 0).setScale(2).setImmovable(true).setDepth(-2);

this.npc3 = this.physics.add.sprite(220, 220, CST.SPRITE.NPC4, 4).setScale(2).setImmovable(true).setDepth(-2);

this.npc4 = this.physics.add.sprite(630, 470, CST.SPRITE.NPC, 12).setScale(2).setImmovable(true).setDepth(-2);

this.npc5 = this.physics.add.sprite(610, 205, CST.SPRITE.NPC5, 0).setScale(2).setImmovable(true).setDepth(-2);
```

Figur 18: Phaser innebygde sprite klasser.

Ved bruk av Phaser sin innebygde sprite klasse (Figur 18) opprettes alle de ulike sprite'ene som blir brukt i spillet. Det blir her deklarert at disse skal håndteres av nevnt fysikk plugin. Dette gir mulighet for definering av hver sprite sin hitbox.

```
// Dialog npc1
// Første dialog
this.Npc1BubbleTest1 = new SpeechBubble(this, {
    'width': 200,
    'height': 100,
    'x': 440,
    'y': -90
}, false)
// laster inn tekst
this.Npc1ContentTest1 = this.add.text(0, 0, "Hei Velkommen til oss hos OXX. Visste du at vi leverer i dag

this.Npc1ContentTest1.depth = 1
// finner posisjonen til bobblen
var b = this.Npc1ContentTest1.getBounds();

// setter riktig posisjon til teksen i forhold til bobblen
this.Npc1ContentTest1.setPosition(440 + (200 / 2) - (b.width / 2), -90 + (100 / 2) - (b.height / 2));

// setter at den er usynelig
this.Npc1ContentTest1.visible = false
```

Figur 19: klassen SpeechBubble.js.

All dialog i spillet opprettes på lik måte. Ved å bygge videre på Phaser sin innebygde Graphics klasse, opprettes klassen SpeechBubble.js (Figur 19). I klassen lages det en hvit snakkeboble i gitt størrelse og posisjon. Deretter anvendes Phaser sin text-klasse til å opprette teksten til snakkeboblen. Når objektene er opprettet finner vi posisjonen til snakkeboblen og gir teksten riktig posisjon i forhold til snakkeboblen, så setter vi begge objektene som usynlig i spillet inntil videre.

Etter oppretting av objektene deklareres posisjon for snakkebobler og tilhørende tekst, og settes til usynlig.

```
//oppretter kart
let mappy = this.add.tilemap("mappy");
let terrain = mappy.addTilesetImage("oxxKontor.v3", CST.IMAGE.TERRAIN);

//layers
let botLayer = mappy.createStaticLayer("bot", [terrain], 0, 0).setDepth(-1);
let topLayer = mappy.createStaticLayer("top", [terrain], 0, 0).setDepth(-3);

//kart kollisjon
this.physics.add.collider(this.person, topLayer);

topLayer.setCollisionByProperty({collides:true});

//setter at kameraet følger personen
this.cameras.main.startFollow(this.person);

//setter at man ikke kan gå ut av kartet
this.physics.world.setBounds(0,0, mappy.widthInPixels, mappy.heightInPixels)
this.person.setCollideWorldBounds(true);
```

Figur 20: Opprettelse av kartet.

Denne koden deklarerer og oppretter kartet (Figur 20) i spillet. Ved hjelp av Tiled kart-editor genereres en JSON fil som definerer hvor spillermodellen kolliderer på kartet. Kartet lastes inn ved å bruke Phaser sin innebyde tilemap-klasse. Ved å kombinere JSON filen, bildet av kartet og fysikk plugin defineres hvor spilleren kan, og ikke kan bevege seg. Kameraet settes til å følge spilleren, og kartets ramme settes som ytre hitbox.

```
//legger inn at man kan bruke piltastene
this.keyboard = this.input.keyboard.addKeys("LEFT, UP, RIGHT, DOWN, ENTER")
```

Figur 21 Oppretelse av bevegelse

```
// setter at hvis det er en touch skjerm så kommer det en joystick og en knapp som de kan bruke
if (IS_TOUCH) {

    // laster inn joystick
    // joystick plugin https://rexrainbow.github.io/phaser3-rex-notes/docs/site/virtualjoystick/
    this.joyStick = this.plugins.get('rexvirtualjoystickplugin').add(this, {
        x: (this.game.renderer.width / 3.5),
        y: (this.game.renderer.height / 1.4),
        radius: 50
        // dir: '8dir',
        // forceMin: 16,
        // fixed: true,
        // enable: true
    });

    this.cursorKeys = this.joyStick.createCursorKeys();

    let button = new BasicButton({
        'scene': this,
        'key': 'button',
        'up': 0,
        'over': 0,
        'down': 1,
        'x': (this.game.renderer.width / 1.5),
        'y': (this.game.renderer.height / 1.4)
    })
```

Figur 22 Kode for å legge til joystick og knapp

For at spillet skal kunne benyttes på pc, benyttes Phaser sin innebygde keyboard-klasse (Figur 21). Her defineres det at Phaser skal bevege spillermodellen etter hvilken piltast som blir trykket, og at enter er satt som interaksjonsknapp.

For mobile enheter bruker vi en joystick plugin (Figur 22) som har blitt utviklet av «Rex» (rexrainbow, 2021), til å bevege seg. Klassen BasicButton bygger på sprite-klassen til Phaser. Denne brukes til å definere en knapp som erstatter enter som interaksjonsknapp. Denne plasseres på linje med joysticken.

```
if (this.keyboard.RIGHT.isDown || this.cursorKeys.right.isDown) {
    this.person.setVelocityX(84)
}
```

Figur 23 Setter bevegelseshastighet

Figur 23 demonstrerer koden som beveger spriten rundt i spillet, spesifikt til høyre. Ettersom sprite'en lastes inn med nevnt fysikk plugin har den tilgang til setVelocity metoden for å bevege seg.

```
//stopper personen hvis man ikke presser
if (this.keyboard.LEFT.isUp && this.keyboard.RIGHT.isUp && this.cursorKeys.right.isUp && this.cursorKeys.left.isUp) {
    this.person.setVelocityX(0);
```

Figur 24 Stopper bevegelsen

Figur 24 viser at setVelocity blir satt til 0 når programmet ikke mottar input fra noen av piltastene. Dette er nødvendig da setVelocity ikke automatisk reverteres i etterkant av bevegelsesinput.

```
//gå høyere animasjon
this.anims.create({
    key: "rigth",
    frameRate: 6,
    frames: this.anims.generateFrameNumbers(CST.SPRITE.PERSON, {
        start: 7,
        end: 4
    })
})
```

Figur 25 kode for bevegelsesanimajson

```
//setter at man har animasjon når man går
if(this.person.body.velocity.x > 0){
    this.person.play("rigth", true)
```

Figur 26 Spiller av animasjon

For å lage beveglsesanimasjonen (Figur 25) anvendes Phaser sin anims-klasse. Her brukes metoden create som oppretter animasjonen. Ved bruk av en sprite tilsvarende Figur 12 vises seksjoner av sprite'en etter hverandre for å etterligne at spillermodellen beveger seg. Hvilken animasjon som spilles av, bestemmes av brukerens piltast-input.

Figur 26 Spiller av bevegelses animasjonen av karakteren når den går mot høyre

```
// snakkebobbler npc2
if (this.checkOverlap(this.person, this.npc2)) {
    this.turn(this.person, this.npc2)
    if (test1) {
        if (test2) {
            if (test3) {
                this.Npc2BubbleTest4.visible = true;
                this.Npc2ContentTest4.visible = true;
            } else {
                this.Npc2BubbleTest3.visible = true;
                this.Npc2ContentTest3.visible = true;
            }

        } else {
            this.Npc2BubbleTest2.visible = true;
            this.Npc2ContentTest2.visible = true;
        }

    } else {
        this.Npc2BubbleTest1.visible = true;
        this.Npc2ContentTest1.visible = true;
        if (this.keyboard.ENTER.isDown === true || buttonIn) {
            test1 = true
            this.Npc2BubbleTest1.visible = false;
            this.Npc2ContentTest1.visible = false;
            category = 1; //sett kategori
            processQuiz();
            this.scene.pause();

        }
    }

} else {
    this.npc2.setFrame(0)
    this.Npc2BubbleTest1.visible = false;
    this.Npc2ContentTest1.visible = false;
    this.Npc2BubbleTest2.visible = false;
    this.Npc2ContentTest2.visible = false;
    this.Npc2BubbleTest3.visible = false;
    this.Npc2ContentTest3.visible = false;
    this.Npc2BubbleTest4.visible = false;
    this.Npc2ContentTest4.visible = false;
}
```

Figur 27 Eksempel på snakkeboblers faser

Interaksjonen mellom spilleren og de statiske spritene i spillet har 4 forskjellige faser (Figur 27), med ulik dialog ut ifra hvilken fase av spillet brukeren har nådd. Spillet bytter fase hver gang brukeren gjennomgår en brukertest.

```
// metode som sjekker om 2 sprites sin bounds overlapper, gir ut true hvis de gjør det, false hvis ikke
checkOverlap(spriteA, spriteB){
    let bounds1 = spriteA.getBounds();
    let bounds2 = spriteB.getBounds();

    return Phaser.Geom.Rectangle.Overlaps(bounds1, bounds2);
}
```

Figur 28 Kode som sjekker overlapping av karakterer

Snakkebobler med tilhørende tekst blir satt til synlig, når spillermodellen beveger seg innenfor et satt rektangulært areal rundt de andre karakterene i spillet (Figur 28).

4.3.1.4 Evnetester og brukerregistrering

Grensesnittet for quizene er definert i en HTML-fil ved navn index.html under mappen "quiz". Denne filen og tilhørende funksjonalitet blir kalt på i underliggende JavaScriptet når bruker interagerer med en karakter (Figur 29).

```
if (this.checkOverlap(this.person, this.npc4)) {
    this.turn(this.person, this.npc4)
    if (test1) {
        if (test2) {
            if (test3) {
                this.Npc4BubbleTest4.visible = true;
                this.Npc4ContentTest4.visible = true;
            } else {
                this.Npc4BubbleTest3.visible = true;
                this.Npc4ContentTest3.visible = true;
                if (this.keyboard.ENTER.isDown === true || buttonIn) {
                    test3 = true
                    this.Npc4BubbleTest3.visible = false;
                    this.Npc4ContentTest3.visible = false;
                    category = 3;
                    processQuiz();
                    this.scene.pause();

                }
            }
```

Figur 29 Kall for evnetest

Quizens kategori settes utfra hvilken fase brukeren har nådd i spillet, og processQuiz() blir kalt. Phaser-delen av spillet pauses og logikken for quizen stares via quiz.js. Først blir en forespørsel sendt for å hente en tilfeldig quiz innen gitt kategori. Dataene blir deretter tatt med videre og HTML-en blir bygget opp.

For å unngå at denne koden kjøres gjentagende ganger, er det blitt tatt i bruk en debounce med et callback. Når processQuiz() kalles, kjøres egentlig koden vist i Figur 30.

```
function debounce(func, timeout = 300){
    let timer;
    return (...args) => {
      clearTimeout(timer);
      timer = setTimeout(() => { func.apply(this, args); }, timeout);
    };
}

 const processQuizes = debounce(() => processQuiz());
   function processQuiz(){
     console.log("process quiz");
       $("#quiz-overlay").load( "../game/quiz/quiz.html" );
     const quiz = document.getElementById("quiz-overlay");
       quiz.classList.add("display-quiz");
       getQuiz(category);
 }

 const processPersonalias = debounce(() => processPersonalia());
function processPersonalia() {
    console.log("process personalia");
    $("#personalia-overlay").load("../game/personalia/personalia.html");
    const personalia = document.getElementById("personalia-overlay");
    personalia.classList.add("display-personalia");

}
```

Figur 30 Kode for debounce

Brukerregistrering fungerer på samme måte som quizer, men funksjonaliteten ligger under mappen personalia. I tillegg er det laget regulære uttrykk som sjekker brukers input før skjema sendes inn.

4.3.1.5 Codemirror & Beautify.js

Disse to bibliotekene brukes for å presentere kodesnutter i evnetestene. CodeMirror lager et grensesnitt for koden, som er gjenkjennelig fra programmeringseditorer. Beautify.js formaterer kodesnuttene fra databasen for å sikre riktig syntaks før det legges ut i CodeMirror-modulen, som vist på Figur 32.

CodeMirror initieres i JavaScriptet og legges ut i quizen i de tilfellende der en kodesnutt følger med fra API-kallet (Figur 31). Beautify.js formatterer deretter kodesnutten.

```
function manageCodemirror() {
    /*
    -> i html er det en textarea med id "snippets"
       -> Codemirror tar inn dette textareaet og injiserer egen kode
       -> setter dette til globale variabel editor
       -> satt metode for syntaxhighlight til javascript, den funker bra på det meste
    */
    editor = CodeMirror.fromTextArea
        (document.getElementById('snippets'), {
            mode: "javascript",
            theme: "darcula",
            lineNumbers: true,
            readOnly: true
        });
    document.querySelector('.CodeMirror').classList.add('toggle-codemirror');
}
```

Figur 31 Kode for initiering av CodeMirror

```
function formatCodesnippet(snippet) {
    /*
    -> henter codemirror og fjerner klassen som gjemmer denne
    -> setter verdien med kodesnippeten, bruker beautify for å formattere syntax
    */
    document.querySelector('.CodeMirror').classList.remove('toggle-codemirror');
    editor.getDoc().setValue(js_beautify(snippet));
    editor.refresh();
}

//timer
```

Figur 32 Kode for bruk av Beautify.js

4.3.1.6 Administratorportal

Administratorportalen har en innloggingsside med inputfelt for brukernavn og passord. Disse autoriseres i AuthenticateController ved innlogging. Ved aksessering av administratorportalen vil Javascriptfunksjoner som sender forespørsel til AdminController.cs. For å hente innholdet til de forskjellige sidene kalles loadHtml() automatisk fra en onload-attribut i sidens body-tag. Metoden som kalles er vist i Figur 33, som videre kaller på metoden i Figur 34.

```
function loadHtml() {

    //get admin html
    let myHeaders = new Headers();
    myHeaders.append("Authorization", `Bearer ${sessionStorage.token}`);

    let requestOptions = {
        method: 'GET',
        headers: myHeaders,
        redirect: 'follow'
    };

    fetch("http://localhost:40062/admin", requestOptions)
        .then(response => response.text())
        .then(result => document.getElementById("App").innerHTML = result)
        .catch(error => alert('error', error));

    getTopUsers();
}
```

Figur 33 Kode for henting av HTML fra API-et

```
function getTopUsers() {
    let myHeaders = new Headers();
    myHeaders.append("Authorization", `Bearer ${sessionStorage.token}`);

    //get top users html
    let requestOptions = {
        method: 'GET',
        headers: myHeaders,
        redirect: 'follow'
    };

    fetch("http://localhost:40062/api/users/GetTopUsers", requestOptions)
        .then(response => response.text())
        .then(result => formatTopUsersList(JSON.parse(result)))
        .catch(error => console.log('error', error));

    function formatTopUsersList(result) {
        console.log(result)
        let html = result.map((aResult, index) => {

            return `
                    <tr data-value="${aResult.id}">
                        <td>${index+1}</td>
                        <td>${aResult.firstname} ${aResult.lastname}</td>
                        <td>${aResult.quizResults[0].score}</td>
                        <td>${new Date(aResult.quizResults[0].timeUsed * 1000).toISOString().substr(11, 8)}</td>
                    </tr>
                    `
        });
        $("#topResultsList").html(html);

        $("#topResultsList tr").click(function () {

            getCandidateDetails($(this).data('value'));
        });
    }
}
```

Figur 34 Metodekall som henter topp kandidater

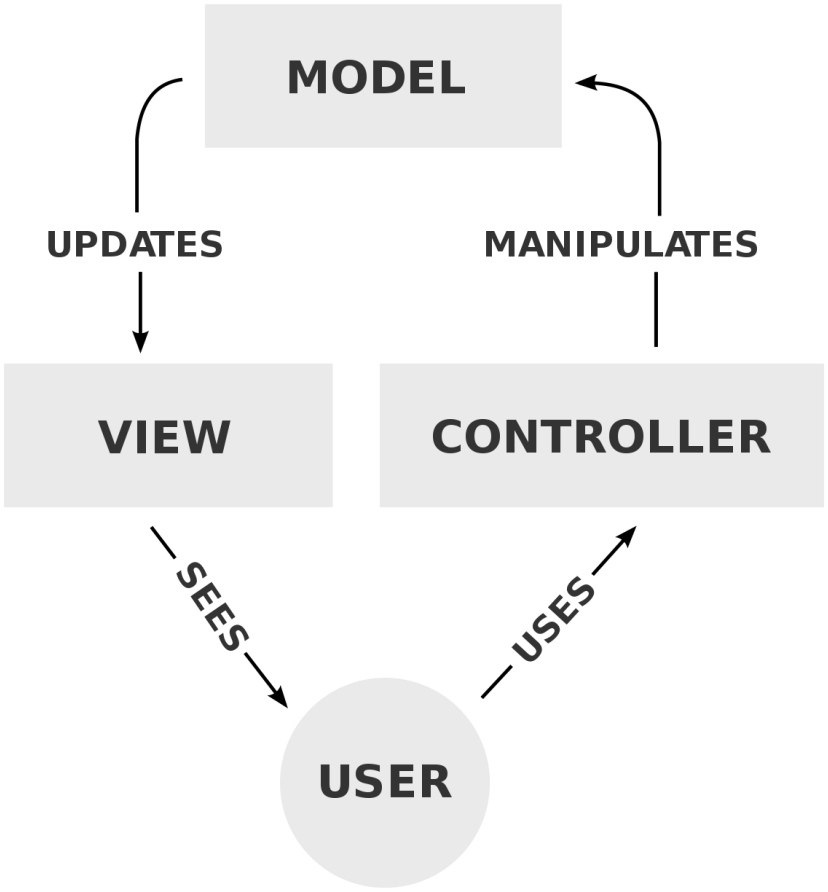
Ved å trykke på logg ut øverst i høyre hjørne vil bruker omdirigeres til innloggingssiden og session vil slettes.

4.4 Backend

Backend i prosjektet er bygget på rammeverket ASP.NET Core 5 for å utvikle et API som skaper bindeleddet mellom frontend og backend. API-et henter og lagrer informasjon til en database og muliggjør utveksling av disse dataene. Prinsippene som er fulgt er REST og CRUD.

4.4.1 Arkitektur

Arkitekturstrukturen til prosjektet er MVC (Model View Controller) og er delt opp i tre deler: GamificationAPI, Data og Domain . Figur 35



Figur 35: Figur hentet fra https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller (Wikipedia, 2021 D). Viser MVC modellen

4.4.2 Database

Databasen er av type MSSQL. Denne inneholder tabeller som rommer informasjon om alt fra kandidater og deres resultater til evnetester/quizer og deres kategorier. Databasen er blitt initiert med forhåndslagde data om evnetester, adnministratorbruker samt kandidater og deres testresultater, for å gi et generelt datagrunnlag. Databasen har også blitt migrert til Azure som en Azure SQL Server med disse dataene. Dette er et proof of concept for at databasen og programmet som er utviklet kan hostes i Azure. Databasen er satt opp etter tredje normalform (3NF).

4.4.2.1 Models

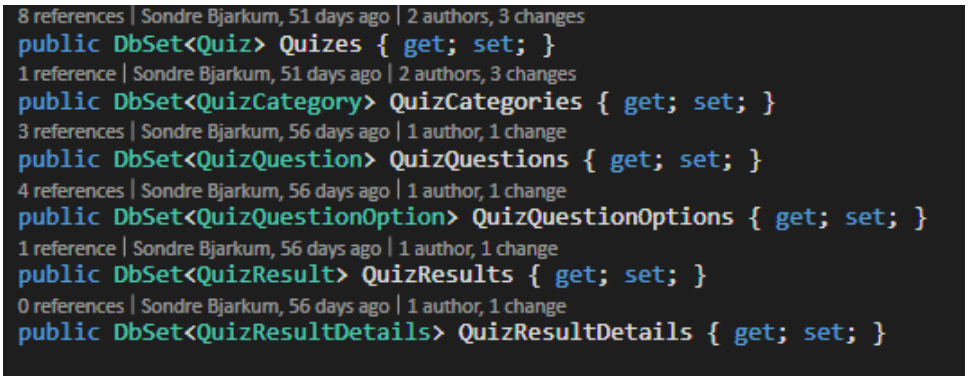
For å definere tabeller i databasen og hva disse skal inneholde, er det laget modellklasser. Disse klassene har attributter som blir oversatt til rader og kolonner. Et eksempel på en slik modell i vår løsning er Candidate (Figur 36). Denne modellen inneholder attributter som fornavn, etternavn, telefon og en referanse til testresultatene. Disse modellene danner grunnlaget for tabeller og blir brukt som refereanser til DBContext.

```
19 references | Sondre Bjarkum, 7 days ago | 2 authors, 4 changes
public class Candidate
{
    5 references | Sondre Bjarkum, 50 days ago | 1 author, 1 change
    public int Id { get; set; }
    10 references | Sondre Bjarkum, 7 days ago | 1 author, 1 change
    public string Firstname { get; set; }
    10 references | Sondre Bjarkum, 7 days ago | 1 author, 1 change
    public string Lastname { get; set; }
    10 references | Sondre Bjarkum, 7 days ago | 1 author, 1 change
    public string Phone { get; set; }
    10 references | Sondre Bjarkum, 7 days ago | 1 author, 1 change
    public string Email { get; set; }
    10 references | Sondre Bjarkum, 7 days ago | 1 author, 1 change
    public string Studies { get; set; }
    10 references | Sondre Bjarkum, 7 days ago | 1 author, 1 change
    public string Semester { get; set; }
    10 references | Sondre Bjarkum, 7 days ago | 1 author, 1 change
    public bool ContactMe { get; set; }
    10 references | Sondre Bjarkum, 56 days ago | 1 author, 1 change
    public string LinkedIn { get; set; }
    10 references | Sondre Bjarkum, 56 days ago | 1 author, 1 change
    public string Webpage { get; set; }
    11 references | Sondre Bjarkum, 7 days ago | 2 authors, 3 changes
    public virtual List<QuizResult> QuizResults { get; set; } = new List<QuizResult>();
}
```

Figur 36: Modellen Candidate.

4.4.2.2 DBContext

DBContext-klassen er implementert for at databasen skal kunne forstå ASP.NET Core 5 sine entiteter. I prosjektet er det en Context-klasse kalt GamificationContext for kandidater og evnetester, samt administratorbrukere. Disse klassene arver en rekke metoder fra DBContext som konfigurerer blant annet tabeller og databasetilkoblingen, og inneholder referanser til modellklassene. For å initiere disse tabellene/modellene brukes en metode som heter DbSet<> i Context-klassene. Hver enkelt modell i Context-klassene settes som en verdi i metoden og gis deretter et navn, eks. DbSet <Candidate> Candidates, hvor tabellen i databasen nå vil hete "Candidates". Dette deklarerer en "Candidates"-tabell i databasen. Eksempel fra prosjektet for bruk av DbSet<> (Figur 37)



Figur 37 Viser DbSet metoden i aksjon i GamificationContext

Context-klassen har også blitt brukt for å injisere en superbruker vist i Figur 38

```
//SEEDER SUPERBRUKER
// any guid
const string ADMIN_ID = "a18be9c0-aa65-4af8-bd17-00bd9344e575";
// any guid, but nothing is against to use the same one
const string ROLE_ID = ADMIN_ID;
builder.Entity<IdentityRole>().HasData(new IdentityRole
{
    Id = ROLE_ID,
    Name = "Super",
    NormalizedName = "SUPER"
});

var hasher = new PasswordHasher<ApplicationUser>();
builder.Entity<ApplicationUser>().HasData(new ApplicationUser
{
    Id = ADMIN_ID,
    UserName = "Admin",
    NormalizedUserName = "ADMIN",
    Email = "some-admin-email@nonce.fake",
    NormalizedEmail = "some-admin-email@nonce.fake",
    EmailConfirmed = true,
    PasswordHash = hasher.HashPassword(null, "Admin-123"),
    SecurityStamp = string.Empty
});

builder.Entity<IdentityUserRole<string>>().HasData(new IdentityUserRole<string>
{
    RoleId = ROLE_ID,
    UserId = ADMIN_ID
});
```

Figur 38 Injisering av superbruker i GamificationContext

4.4.2.3 Migrations

Migrasjoner er skrevet hver endring av databasens modeller, for å sikre at databasen er oppdatert med EF Core sine modeller. Dette er gjort i terminalen for GamificationApp.Data med linjen: “Add-Migration *migrasjonsnavn*”. Denne migrasjonen legges da i Migrations-mappen under GamificationApp.Data. Mappen inneholder filer for hvordan databasen og dens relasjoner til tabeller bygges.

4.4.2.4 DBInit

For å initiere data i databasen under utvikling, brukes klassen DBInit.cs. Denne injiserer verdier til databasen ved bruk av manuelt opprettede objekter basert på modellklassene. Deretter håndterer EF Core relasjonene mellom disse. Bruken av DBInit.cs deklareres i Program.cs.

4.4.2.5 Endepunkter

Alle endepunkter starter med: http://localhost:40062/api/ etterfulgt av kontrollernavn og metode (Tabell 2). Eksempel http://localhost:40062/api/quizs/GetQuizes returnerer alle quizer.

Tabell 2 Oversikt over endepunkter til API-et

Kontroller	Metode	Input	Funksjon	Autorisering
Quizs	GetQuizes	Ingen	Reurnerer alle quizer med relatert data	Ja
Quizs	GetQuizesWithCategory	Kategori {id}, int	Henter en tilfeldig quiz innen kategori	Nei
Quizs	GetQuiz	{id}, int	Henter quiz med id	Ja
Quizs	PostQuiz	Quiz-modell	Lagrer en quiz til databasem	Ja
Quizs	DeleteQuiz	{id}, int	Sletter quiz med relatert data	Ja
Users	GetUsers	Ingen	Henter alle kandidater	Ja
Users	GetUser	{id}, int	Henter kandidat med id	Ja
Users	GetTopUsers	ingen	Henter kandidater	Ja

			med beste score siste måneden	
Users	PostCandidateResults	CandidateHelper	Lagrer en kandidat med resultater til DB	Nei
Users	GetUserQuizResult	{id}, int	Henter gitt kandidat sine evnetestresultater	Ja
Users	DeleteUser	{id}, int	Sletter kandidat og relaterte testresultater	Ja

4.4.3 Controllers

For å gjøre kall til API-et er det laget controllers. Disse controllerne håndterer forespørsler fra nettleseren, sender forespørslene videre til databasen som igjen returnerer en respons. Dette skjer uavhengig av om det er en liste over alle kandidater eller en gitt quiz/evnetest. Prosjektet inneholder fire controllere: AdminController, AuthenticateController, QuizsController og UsersController. Hver av disse controllerne håndterer forskjellige spørringer fra nettleseren, henholdsvis aksessering av innholdet til administratorsiden, brukerautorisering og innlogging, håndtering av quizer og håndtering av kandidater.

4.4.3.1 AuthenticateController

AuthenticateController kan lage administratorbrukere og logge de inn. Den bruker Microsoft sitt Auth-rammeverk for å automatisere prosessen. Ved innlogging vil denne controlleren ta imot brukernavn og passord, ta dette videre til databasen og returnere en JWT Bearer token. Denne brukes gjennomgående i applikasjonen, der autorisering er nødvendig. Slike token's har en automatisk utløpstid på en time. For å sjekke om en bruker er logget inn, er det for alle controllere der endpoint skal være beskyttet lagt til "[Authorize(Roles = UserRoles.Admin)]". Denne sjekker opp mot Microsoft Auth, ved bruk av den returnerte JWT Bearer token, om innloggede bruker har tilgang.

4.4.3.2 AdminController

For å hindre uautorisert tilgang til innholdet på administratorsiden bruker denne controlleren JWT Bearer token fra AuthenticateController, for å autorisere brukeren. Forskjellige endpoints vil returnere forskjellige HTML-filer som deretter blir lagt ut i front-end.

4.4.3.3 QuizsController

QuizController har ansvaret for å håndtere henting og posting av quizer. GetQuizesWithCategories() (Figur 39) er det mest brukte endepunktet. Denne er til for å hente ut tilfeldige quizer innenfor en gitt kategori. Når en bruker skal gjennomføre en quiz, er det dette endepunktet som blir kalt, og hvilken kategori er med som en parameter. Utover dette er det generelle endepunkter for bruk med en CRUD api: get, set, post og delete. Disse ligger bak en autorisering ettersom de kun tiltenkt brukt av en administrator.

```
public async Task<Quiz> GetQuizesWithCategory(int category)
{

    //hent quizer
    List<Quiz> get = await _context.Quizes.Select(q => new Quiz
    {
        Id = q.Id,
        QuizCategory = q.QuizCategory,
        QuizQuestions = q.QuizQuestions,
        Title = q.Title,
        TotalScore = q.TotalScore
    }).ToListAsync();

    //sorter på kategori
    List<Quiz> sorted = new List<Quiz>();
    foreach (var q in get)
    {
        if (q.QuizCategory.Id == category)
        {
            sorted.Add(q);
        }
    }
    //random tall opptill lengden på "sorted"
    var random = new Random();
    int index = random.Next(sorted.Count);

    //returner tilfeldig quiz
    return sorted[index];
}
```

Figur 39 Viser koden til metoden GetQuizesWithCategory

4.4.3.4 UsersController

Rollen til UsersController er hovedsakelig å gi administratorsiden informasjon om gitt brukers resultater samt deres personalia. Alle metodene er låst bak autorisering utenom metoden PostUserQuizResult, som lagrer en ny kandidat med deres quizresultater. Utover dette er det mulig for administrator å opprette kandidater via PostUser eller PutUser, samt oppdatere eller slette gitt kandidats informasjon og resultater.

4.5 Accessibility

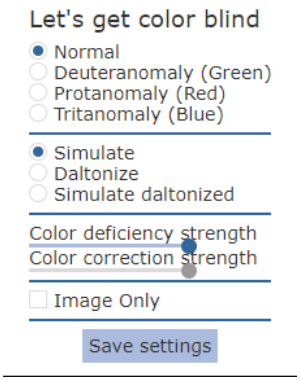
I Norge er det vedtatt å følge WCAG 2.0-standarden (§1, 2019). WCAG er en forskrift med regelverk og krav om universell utforming av nettløsninger og automater, og har vært lovbundet i Norge siden 2014 (Tilsynet for universell utforming av ikt, 2021). Loven har til hensikt å gjøre nettsteder og øvrige teknologiske løsninger tilgjengelig og funksjonelt for flest mulig, uten at det går for mye utover bedriften som står bak. (Henry, 2021)

4.5.1 Dynamisk skalering

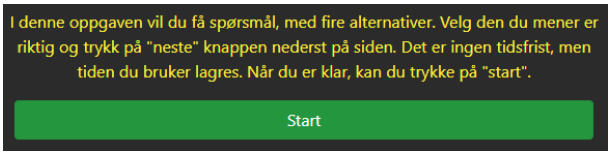
Da spillet i prosjektet i hovedsak er tiltenkt mobilbrukere, er det av høy prioritet at applikasjonen fungerer på flest mulige enheter, både med tanke på størrelse og operativsystem. Med begrunnelse i dette er dynamisk skalerende frontend i forhold til skjermstørrelse implementert. Dette sikrer at applikasjonen er tilpasset, både mobil, nettbrett og pc i varierende størrelser.

4.5.2 Fargevalg

Fargevalget i quizer-, registreringsskjema- og på administratorsiden er basert på profilen til OXX. En kombinasjon av grå bakgrunn med hvit og gul oppleves som god kontrast og leselighet. Det smale utvalget av farger og størrelsesjustering av font gir også sidene god leselighet for fargeblinde. Dette er testet med en fargeblindsimulator-utvidelse i nettleseren Google Chrome (Figur 40).



Figur 40 Fargeblind simulator



Figur 41 Grønn start knapp

Quizen har to ekstra farger. På forklaringssiden til quizene er startknappen grønn (Figur 41). Dette er Bootstrap sin btn-success. Valget er gjort for å tydeliggjøre hva knappen gjør, da grønn som «startfarge» er en vanlig konnotasjon. F.eks. i lyskryss. «Neste»-knappen er markert med btn-primary. Dette gir en klar blåfarge og er tenkt å tydeliggjøre hvor brukeren går videre etter å ha svart på et spørsmål. Knappen er på bakgrunn av dette deaktivert før brukeren velger et svaralternativ. Når brukeren velger et alternativ blir knappen aktivert, som gjør at blåfargen blir lysere og kraftigere.

Dette fargevalget har en svakhet i at grønn og blåfargen er tilnærmet helt like for personer med blå-grønn fargeblindhet. Dette er noe vi ønsker å utbedre i videre utvikling.

I tillegg til fargeblindfilter har vi også testet med et svart-hvitfilter (Figur 42). Leseligheten er fortsatt god. Det er vanskeligere å skille karakterer med mørkere hudtone fra bakgrunnen der den også er mørk, men heller ikke dette fremstår som kritisk for sidens leselighet

Figur 42 Karakterer i svarthvit

4.5.3 Bilder

```
<img src="test-eksempel.png" alt="Eksempel på test" width="100%"
height="auto"> == $0
```

Figur 43 Alt tekst i inspiser

Bilder i applikasjonen har en alternativ tekst (Figur 43), som vises dersom det er noe galt ved innlasting av bildet. Denne beskriver hva bildet var ment å vise.

4.5.4 Språk

Språket på sidene er satt til Norsk, da dette er språket hovedsakelig brukt av ansatte og søkere. Utviding av tilgjengelige språk er en utbedring som burde implementeres, da mange studenter og jobbsøkere i Norge kommuniserer på engelsk.

4.5.5 Inkluderende design

Figur 44 Skisser av karakterer

Figurene i spillet er designet for å dekke flere forskjellige hår og hudfarger (Figur 44) der det er mulig å representere med begrensningene som kommer med antallet piksler og gruppens ferdigheter innen pikseldesign. I fremtidig utvikling er det tenkt at brukere skal kunne velge mellom et utvalg av karakterer.

5. Testdokumentasjon

Gamification av ansettelser

Oslomet – Gruppe 31 – vår 2021

Skrevet av

Christian Dyrli

Jørgen Borander

Maximilian Stiegler Sharoyan

Sondre Næbb Bjarkum

5.1 Forord

I denne delen av rapporten blir det redegjort hvordan gruppen har foretatt seg testing av applikasjonen, hvilke programvare som er brukt og fremgangsmåte. Denne delen inneholder begreper og uttrykk som kan oppleves ukjent. Disse begrepene finner leser i ordlisten.

5.2 Fremgangsmåte

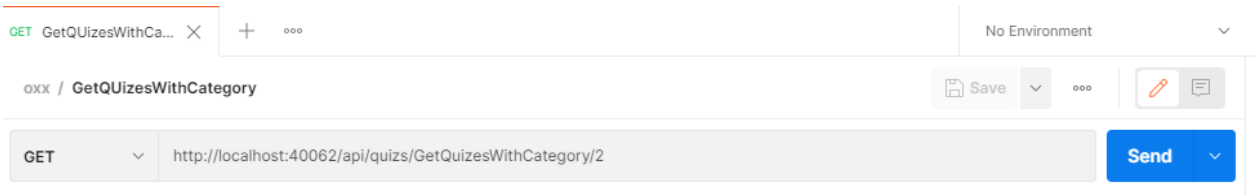
Hensikten med å teste er å sikre hvorvidt funksjonaliteten til programvaren har forventet utfall. Dette oppnås ved å teste så tidlig som mulig slik at mulige feil også blir oppdaget så tidlig som mulig. Gruppen tok i bruk PostMan tidlig i utviklingsfasen av backend, for deretter å lage enhetstester. Dette er en noe utradisjonell tilnærming ettersom man gjerne lager enhetstestene først. Dette forklares med noe manglende kunnskap innen testing i ASP.NET Core 5, men fremgangsmåten viste seg å fungere godt til vårt formål.

5.3 PostMan

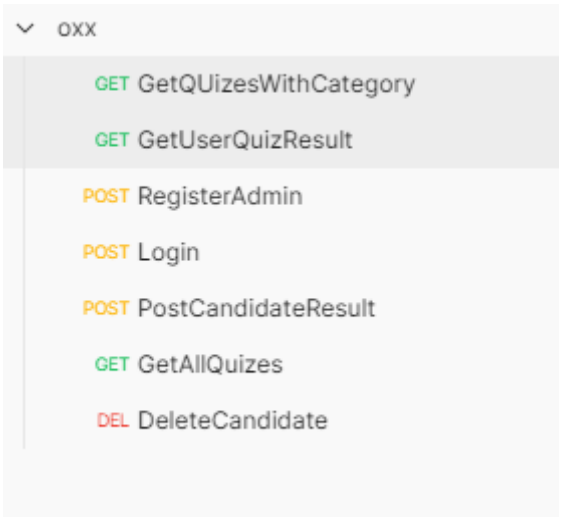
For å komme raskt i gang med testing av backend, besluttet gruppen å ta i bruk verktøyet PostMan. Gruppen brukte PostMan for manuell testing, ved å selv analysere resultatet i rådata fra API-et. Dette til forskjell fra automatisk testing som automatisk sjekker resultatet.

5.3.1 Spørringer

Spørringer i PostMan gjøre ved å definere endepunkt og spørremetode vist i Figur 45. For å organisere spørringene ble det laget en kolleksjon av nødvendige endepunkter vist under i Figur 46. Til slutt, etter å ha sendt en spørring, vil responsen fra API-et legges ut som rådata (JSON) vist i Figur 47.



Figur 45 En spørring i PostMan

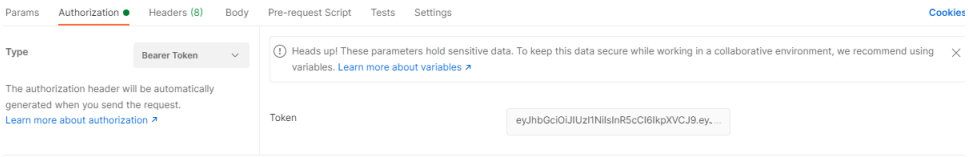


Figur 46 Oversikt over spørring

Figur 47 Respons fra spørring

5.3.2 Autorisering

Ettersom funksjonaliteter som krever autorisering må integrasjonstestes for å tas i bruk i enhetstester, besluttet gruppen å teste dette manuelt ved bruk av PostMan. For å gjøre kall til endepunkter som krever autorisering, er man nødt til å logge inn som administrator i applikasjonen. Deretter kopierer man token som blir returnert og limer det inn under autorisering som bearer token, vist under i Figur 48.



Figur 48 Eksempel på autorisering

5.4 Enhetstesting

Enhetstesting er programvaretesting der enkeltstående enheter og komponenter i programvaren blir testet. Hensikten er å validere at en gitt enhet av programvaren fungerer på tiltenkt måte. I programvaren er en enhet den minste testbare delen og har gjerne få inputs og en enkel returverdi.

I dette tilfelle har kun backend blitt enhetstestet og kun logikk gruppen anser som hensiktsmessig å teste. Dette vil si at metoder som kun er til for videreutvikling eller som ikke fremtrer vesentlig viktig, ikke er blitt tatt med.

Mønsteret som er brukt for å lage testene er “Arrange, Act, Assert”, som er ansett for å være en grundig metode å teste på (Automation Panda, 2021).

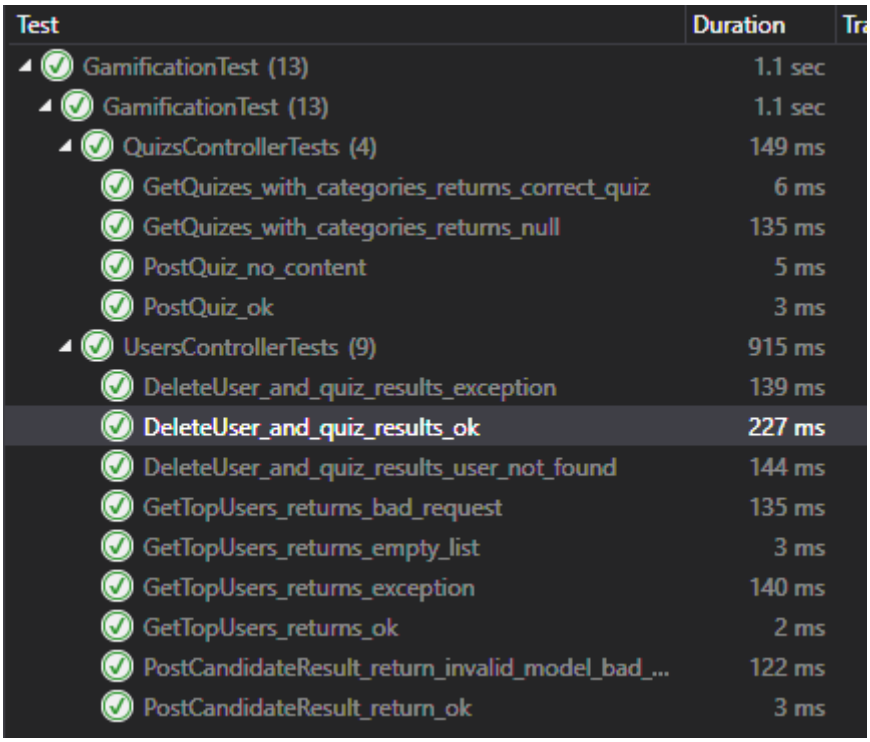
5.4.1 Testmiljø

Testene ble utført i Visual Studio 2019 ved bruk av xUnit. Klasser og metoder ble mocket ved bruk av biblioteket Moq. Standard oppsett for test suite ble brukt hvor det er laget en

mappe ved rotnivå som inneholder klasser med navn som referer til hvilken kontroller som testes.

5.4.2 Enhetstestene

Navngivning av testene følger et mønster for hva forventet utfall skal være. Figur 49 under viser en total oversikt av test suiten og derav testene den inneholder. Testene har blitt kjørt i sin helhet under utvikling, for eksempel ved endring eller tilføying av logikk i gitt controller. Under følger en oversikt i bilder av en enhetstest.



Figur 49 Oversikt over test suite

Videre vises eksempel på en test (Figur 50), hvor funksjonen som testes kalles (Figur 51), og utfallet i grensesnittet av metoden (Figur 52).

```
[Fact]
| 0 references | Sondre Bjarkum, 5 days ago | 2 authors, 3 changes
public async Task GetQuizes_with_categories_returns_correct_quiz()
{
    //Arrange

    var mock = new Mock<IQuizsRepository>();


    var aquiz = new Quiz()
    {
        Id = 1,
        QuizCategory = new QuizCategory
        {
            Id = 1,
            Name = "Test Category"
        },
        Title = "Quiz Title 1",
        TotalScore = 0
    };

    mock.Setup(k => k.GetQuizesWithCategory(1)).ReturnsAsync(aquiz);
    var controller = new QuizsController(mock.Object);

    //Act
    var response = await controller.GetQuizesWithCategory(1);

    //Assert
    Assert.Equal<Quiz>(aquiz, response.Value);
}
```

Figur 50 Eksempel på test

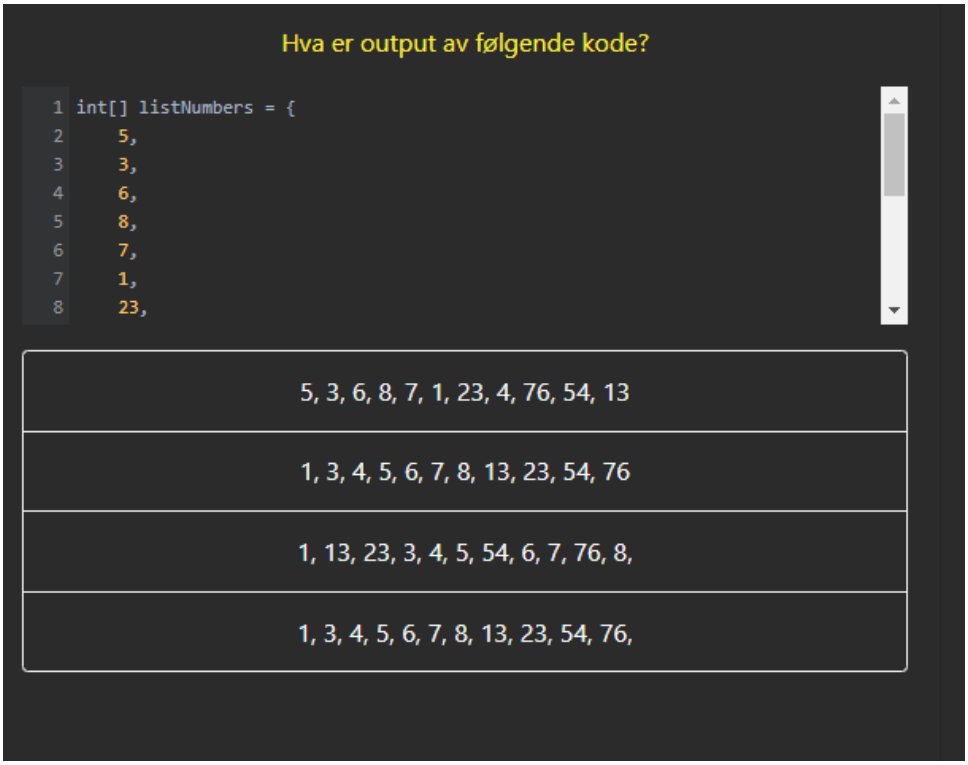
```
async function getQuiz(category) {
    /*
    -> setter i gang codemirror for snippets
    -> henter data fra api, return verdier er JSON
    -> setter denne dataen til variabel data
    -> initierer resten av quizen
    */

    let response = await fetch('http://localhost:40062/api/quizs/GetQuizesWithCategory/'+category);
    let datas = await response.json();
    data = datas;

    manageCodemirror();

    init();
}
```

Figur 51 Eksempel på metode kall



Figur 52 Eksempel på resultat av metode

5.4.2 Autorisering

Det var ønskelig å integrasjonsteste autoriseringen. Derimot viste det seg at dette var svært komplekst og beveget seg utenfor vårt scope. For å teste autorisering ble PostMan brukt fremfor å lage tester i Visual Studio 2019.

Riktignok ble det gjort mange forsøk på å enhetsteste metoder som var lukket bak autorisering.

Oversikt over noen fremgangsmåter som ble prøvd:

1. opprette User, UserRoles og UserClaims for bruk i Identity og injisere falsk data
2. forsøke å bypasse [Authroize] i sin helhet
3. definere en falsk Startup.cs som ikke injisjerer autoriseringsmiddleware

Uheldigvis var ikke noen av fremgangsmåtene vellykket grunnet Identity-middlewaren sin kompleksitet. Det ble besluttet å kommentere ut metoden som definerer at autorisering skal brukes. Dette er dokumentert i test suiten under filen “readme.txt”.

5.4.3 Code Coverage

Bransjestandarden for code coverage er 70%-80% (Cornett, 2021). Ser man på applikasjonens code coverage i sin helhet er den på kun 21%. Dette er fordi metoder gruppen ikke har brukt, eller som er skrevet for videreutvikling, ikke har blitt enhetstestet da dette ikke var kritisk for vår utvikling. Derimot om en kun ser på metodene gruppen har enhetstestet, er det en code coverage på nærmere 100%, som mer enn tilfredsstiller krav.

5.5 Gjenstående aktiviteter

Metoder som bruker autorisering, er ikke blitt testet gjennomgående. Ettersom autorisasjon må integrasjonstestes og var utenfor vårt scope, er dette noe som bør implementeres før videreutvikling. Videre bør det også implementeres resterende enhetstester for Http-metodene tiltenkt videreutvikling.

Ikke alle utfall fra metoder er blitt testet og en mer helhetlig gjennomgang av dette burde vurderes for videreutvikling.

6. Kravspesifikasjoner

Gamification av ansettelser

Oslomet – Gruppe 31 – vår 2021

Skrevet av

Christian Dyrli

Jørgen Borander

Maximilian Stiegler Sharoyan

Sondre Næbb Bjarkum

6.1 Forord

Kravspesifikasjonene er laget for å fungere som et kart for utviklingen og prosjektet, slik at både produkteier og utviklerne sammen skal danne et bilde om hvordan sluttproduktet skal se ut.

6.2 Bakgrunn

Oppgaven som ble tildelt fra OXX handlet om å utvikle et supplement til deres ansettelsesprosess, som kunne skape et godt inntrykk av OXX når de skal hente nye kandidater for ansettelse

6.3 Utviklingen av Kravspesifikasjonene

Kravspesifikasjonene ble utviklet i samarbeid med OXX. Gruppen hadde stor frihet i denne prosessen da bedriften ikke ønsket å begrense mulighetene gruppen hadde i utvikling av applikasjonen. OXX ønsket å bruke oppgaven som en mulighet til å få eksterne innspill og ideer. Det ble derfor gruppens ansvar å bestemme hvilken retning gruppen ønsket å ta oppgaven, og deretter utvikle kravspesifikasjonen i samarbeid med OXX basert på retningen vi ønsket å gå. Dette ble gjort over flere samtaler med bedriften, før gruppen endte på en liste med krav begge parter var fornøyd med som et utgangspunkt.

6.4 Rammekrav

Enkelte kravspesifikasjoner er markert med rødt eller oransje for å indikere at de enten ikke er fullført, eller om de er delvis fullført. Overordnet er årsaken til dette mangel på tid, og at det dermed ikke er blitt prioritert. Punktene markert med oransje er derimot underveis.

Rød = ikke fullført

Oransje = delvis fullført

Ikke markert = fullført

6.4.1 Funksjonelle krav

1. Applikasjonen skal ha en administratorside.

2. Administratorsiden skal ha en oversikt over alle deltakere/kandidater.

3. Teststatistikk for hver enkelt kandidat skal lagres, og gjøres tilgjengelig for administratorer.

4. Administrator skal kunne sortere statistikken.

5. Flere nivåer av vanskelighetsgrad på tester, skal vise forskjellen på 1. og 3. års studenter.

6. Applikasjonen skal være engasjerende for å motivere kandidater.

7. Evnetestene skal teste kandidaten sin kunnskap om forskjellige programmeringsspråk og ferdigheter innen disse.

8. Det skal lages en form for statistikk ut av testresultatene som skal kunne brukes til å rangere deltakere.

9. Skal fungere som en plattform for å promotere OXX.

10. Skal kunne tas i bruk på stand eller presentasjoner.

11. Skal skape engasjement for å lede til intervjuer.

12. Ha et utvalg av evnetester som er uttrykt på forskjellige måter.

13. Mulig å oppdatere og legge til nye tester.

14. Bruker skal kunne velge mellom flere karakterer å spille som.

15. Skal kunne laste opp kontaktinformasjon.

16. Bør kunne fungere på både mobil og desktop.

17. Bruker skal enkelt klare å navigere seg rundt i webapplikasjonen og forstå hvordan han/hun får gjort testene.

18. Applikasjonen skal kunne være lett tilgjengelig.

6.4.2 Ikke funksjonelle krav

1. Webapplikasjonen må kunne tjenes på en webserver.

2. Applikasjonen skal benytte Microsoft sin teknologistack, da dette er hva OXX bruker fra før.

3. Webapplikasjonen må kunne funke på tvers av nettlesere og burde være kompatibel med nettlesere som Safari, Chromium, Firefox og Opera.

4. Løsningen bør kunne fungere på eldre nettlesere, så langt det lar seg gjøre.

5. Skal kunne hostes og kjøres på nettet med minimalt med nedetid.

6. Webapplikasjonen skal testes ved å bruke følgende testmetoder: enhetstest.

7. Etter beste evne følge retningslinjer til WCAG og til minimum 80% via tredjeparts WCAG-checkers.

8. Siden skal kunne lastes inn på <10 sekunder.

9. Applikasjonen skal være skalerbar for mobil og desktop.

Funksjonelle krav

5. Punktet om vanskelighetsgrad er satt til oransje fordi spørsmålene i databasen er markert med vanskelighetsgrad, og er hentet fra forskjellige fag og semester fra studiet. Den er derimot ikke svart, da dette ikke vises i resultatene på administratorsiden.

6. Punktet om engasjement og motivasjon er markert med oransje, da brukertesten gruppen forberedte for å få tilbakemelding rundt dette, ikke ble gjennomført. Derimot har muntlig tilbakemeldinger fra testkandidater underveis uttrykt fasinasjon.

7. Punktet er markert da spørsmålene som nå ligger i databasen dekker noe kunnskap om programmeringsferdigheter og systemutvikling, men det ikke dekker flere språk, slik det står i spesifikasjonen.

13. Denne er markert da frontend er klargjort, men funksjonaliteten mangler i backend.

Ikke-funksjonelle krav

6. Punktet er markert da applikasjonen ikke er testet i den grad gruppen hadde ønsket, men testingen er underveis.

7. Er markert da vi har jobbet mot at applikasjonen skal bestå kravene WCAG dekker, men applikasjonen det var tenkt å teste løsningen med, fungerte ikke på gruppens webapplikasjon, noe gruppen tror kan være forårsaket av en av teknologiene vi anvender.

6.4.3 Krav til kode

1. Det skal være lett å implementere ny kode.

2. Kode skal kommenteres.

6.5 Aktører

I prosjektet finnes det to type aktører, primæraktørene og sekunderaktørene. Primæraktørene er kandidatene som skal ta testene, og administratorene som er ansatt hos OXX. Sekundæraktøren er Azure Appservice.

6.6 Brukerhistorier

Brukerhistoriene for kandidat og administrator kan en se i Tabell 3 og Tabell 4

6.6.1 Kandidat

En kandidat vil være en person som enten har mottatt spill-lenke på epost, eller en person som mottar lenken fra OXX på stand.

Tabell 3: Brukerhistorier kandidat.

Brukerhistorie	Prioritet
Som kandidat ønsker jeg meg informasjon om hvordan jeg beveger meg	Høy
Som kandidat ønsker jeg å kunne gå inn på evnetestene	Høy
Som kandidat ønsker jeg at testene skal teste mitt nivå	Middels
Som kandidat ønsker jeg informasjon om hvordan jeg gjennomfører en evnetest	Middels
Som Kandidat ønsker jeg å kunne registrere min informasjon for å bli kontaktet	Høy

6.6.2 Administrator

En administrator vil være en registrert person med tilknytning OXX. En administrator skal kunne se alle registrerte kandidater og tilhørende resultater.

Tabell 4: Viser Brukerhistorier Administrator side.

Brukerhistorie	Prioritet
Som administrator ønsker jeg å kunne logge inn på administrator siden	Høy
Som administrator ønsker jeg å kunne se scoren til de forskjellige kandidatene	Høy
Som administrator ønsker jeg å kunne filtrere kandidater basert på tid	Lav
Som administrator ønsker jeg å se kontaktinformasjonen til hver enkelt kandidat	Høy
Som administrator ønsker jeg å kunne administrere testene	Middels
Som administrator ønsker jeg å kunne opprette nye administratorer	Middels

7. Avslutning

Gamification av ansettelser

Oslomet – Gruppe 31 – vår 2021

Skrevet av

Christian Dyrli

Jørgen Borander

Maximilian Stiegler Sharoyan

Sondre Næbb Bjarkum

I prosjektperioden har gruppen laget og utviklet en HiFi-prototype av en webapplikasjon. Prosjektet oppfyller mesteparten av krav og retningslinjer som ble utarbeid i samarbeid med OXX. Løsningen kan enkelt implementeres i deres eksisterende prosess for ansettelse og vi tror den kan hjelpe OXX med å skape oppmerksomhet rundt selskapet.

Vi mener, basert på den endelige løsningen samt tilbakemeldinger fra OXX i sluttfasen, at det leverte produktet treffer godt. Løsningen tester kandidater og er noe OXX kan benytte på stands som gjør at de skiller seg ut. Ved gjennomføring av evnetester settes spillet på pause og bruker sendes til en ekstern side. På bakgrunn av dette mener gruppen at det er tilrettelagt for enkel justering, eller utskiftning av evnetester.

Vårt fokus har hele veien vært å ha jevnlige møter med OXX for å få deres tilbakemeldinger, og ønsker rundt videre arbeid.

Vi håper at OXX, ved bruk av denne løsningen på stands, vil kunne spre oppmerksomhet og bygge et godt omdømme rundt bedriften blant studenter. Dette med hensikt om å styrke sin søkerbase.

Rapportens hovedfokus er å gi lesere et bilde av hvordan gruppen jobbet og utførte valg underveis. Derimot mener vi at rapporten også kan brukes som dokumentasjon for videreutvikling eller for øvrige aktører som jobber med lignende problemstillinger.

Oslo ble i løpet av høsten 2020 åpnet mer og mer, men måtte i november stenges så godt som helt ned igjen. Denne svært usikre situasjonen rundt pandemien medførte store utfordringer for oss i forhold til samarbeid med produkteier OXX, så vel som innad i gruppen. Originalt planla vi å jobbe ved kontorene til OXX, men det måtte dessverre skrinlegges. Kommunikasjon måtte derfor tas hovedsakelig digitalt fra hjemmekontor, og i likhet med mange andre, føler vi at dette har ført til delvis kreativ tørke og hindret produktivitet.

Til tross for dette har vi klart å forholde oss til tidsfristene vi satte oss i løpet av prosjektet. Samtidig har vi klart å utvikle webapplikasjonen selvstendig noe som er en bekreftelse på vår læring fra årene på OsloMet.

Avslutningsvis ønsker vi å trekke frem at prosjektet har vært svært lærerikt og interessant. Vi har tilegnet oss ny kunnskap og veldig relevant erfaring igjennom arbeidet med prosjektet.

8. Brukerveiledning

Gamification av ansettelser

Oslomet – Gruppe 31 – vår 2021

Skrevet av

Christian Dyrli

Jørgen Borander

Maximilian Stiegler Sharoyan

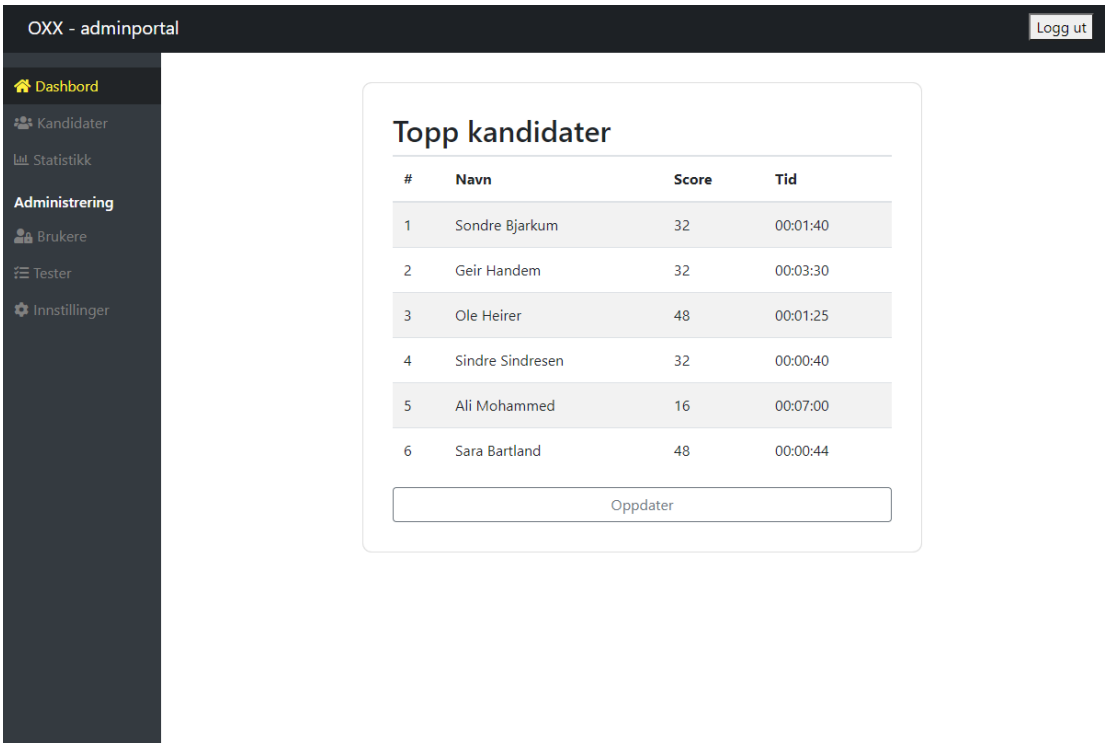
Sondre Næbb Bjarkum

8.1 Forord

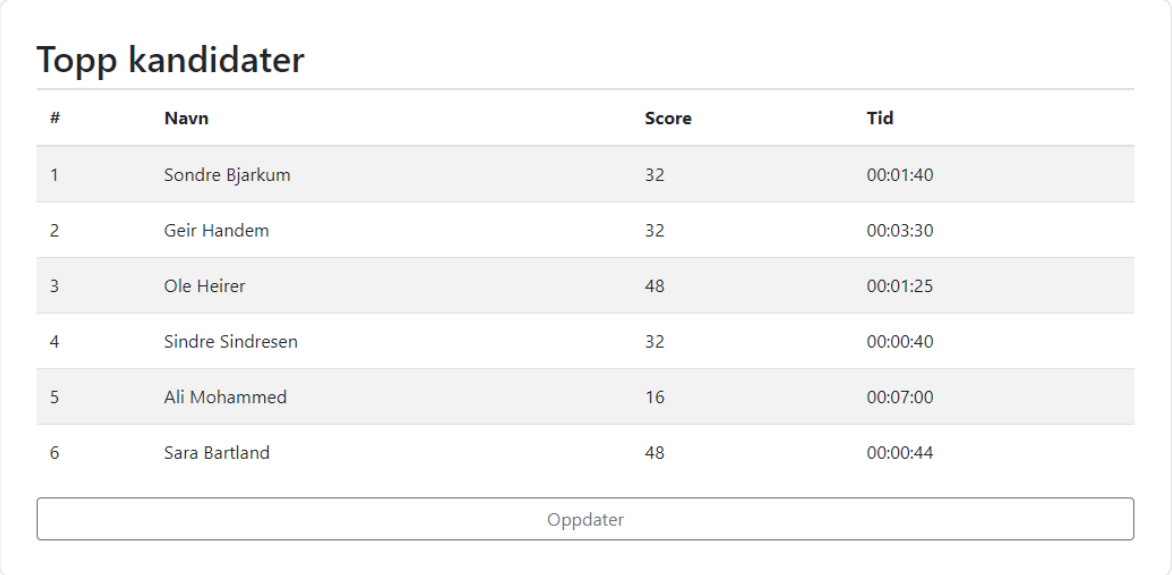
Denne seksjonen inneholder en dokumentert veiledning for personer som skal benytte seg av løsningen. Brukerveiledningen er todelt. En del om administratorsiden og en del om spillsiden. Veiledningen inneholder illustrasjoner som supplement til teksten.

8.2 Administratordel

Administratorer ankommer først applikasjonens innloggingsside. Der må brukernavn og passord fylles inn, og "Logg inn"-knappen må trykkes. Når dette er gjort sendes bruker videre til dashbordsiden (Figur 53).

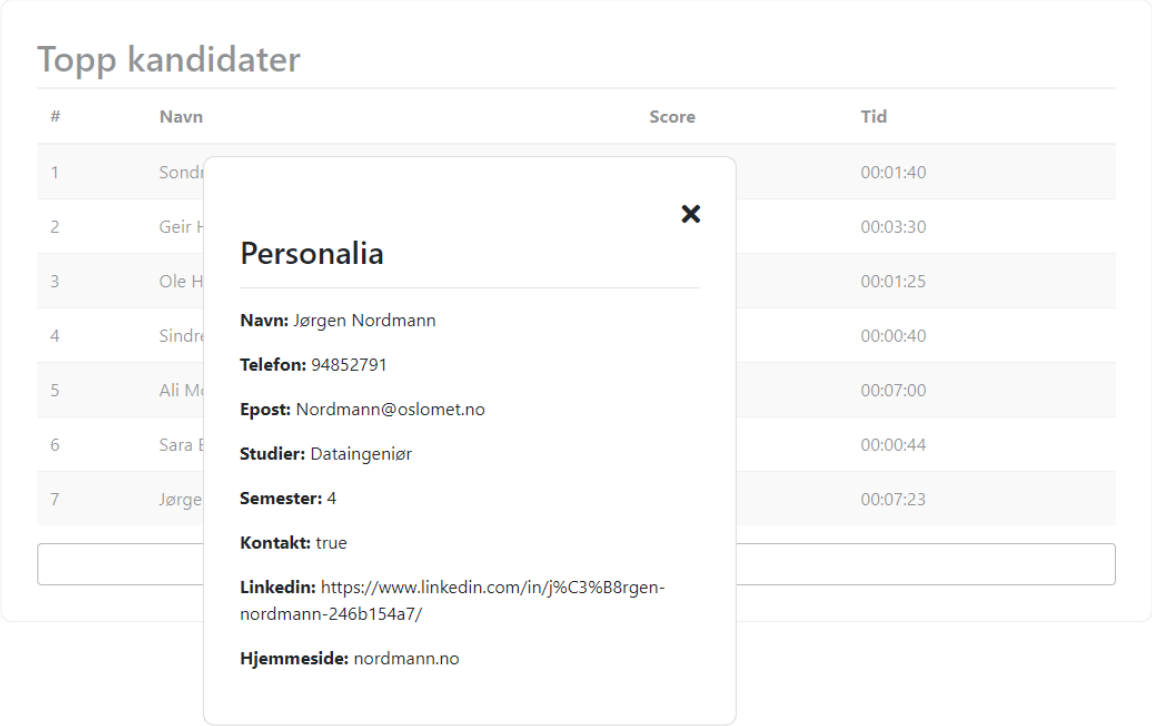


Figur 53 Dashbordsiden

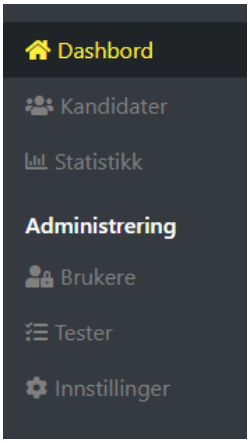


Figur 54: Toppkandidater fra adminsiden.

På dashbordsiden presenteres statistikk om de 10 beste kandidatene siste måned (Figur 54) den siste måneden bli listet opp. For å få opp detaljert informasjon om hver kandidat, kan bruker klikke på kandidatens rad i tabellen. (Figur 55).



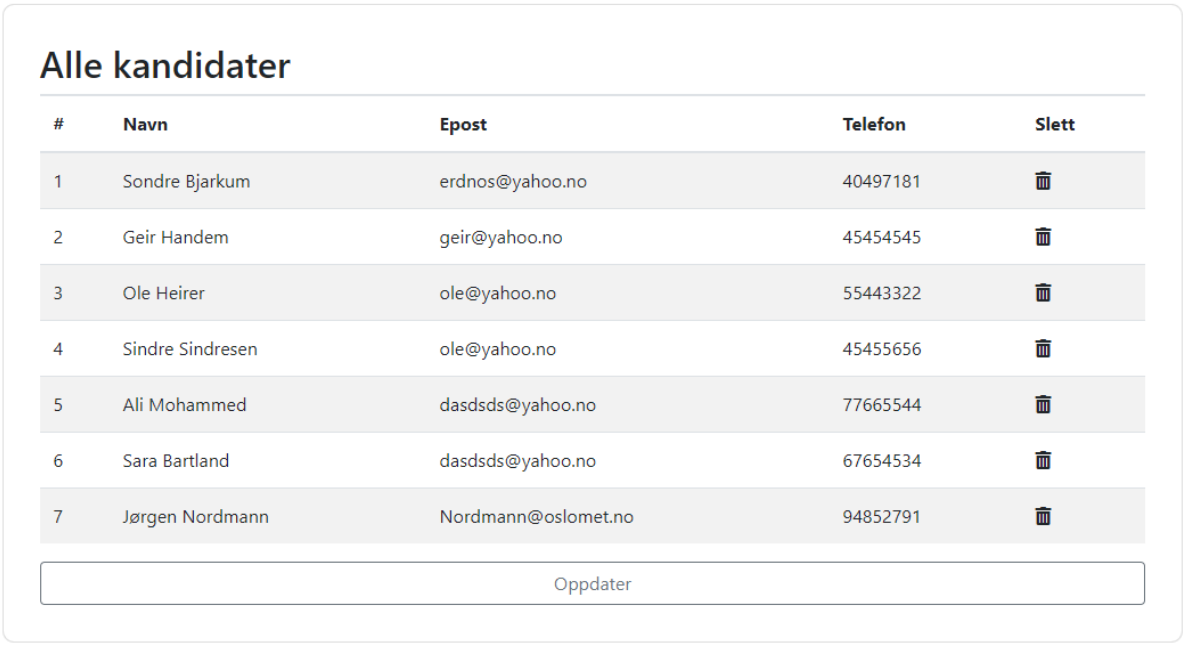
Figur 55: Informasjon om en kandidat.



Figur 56 Navigeringsbar administrator side

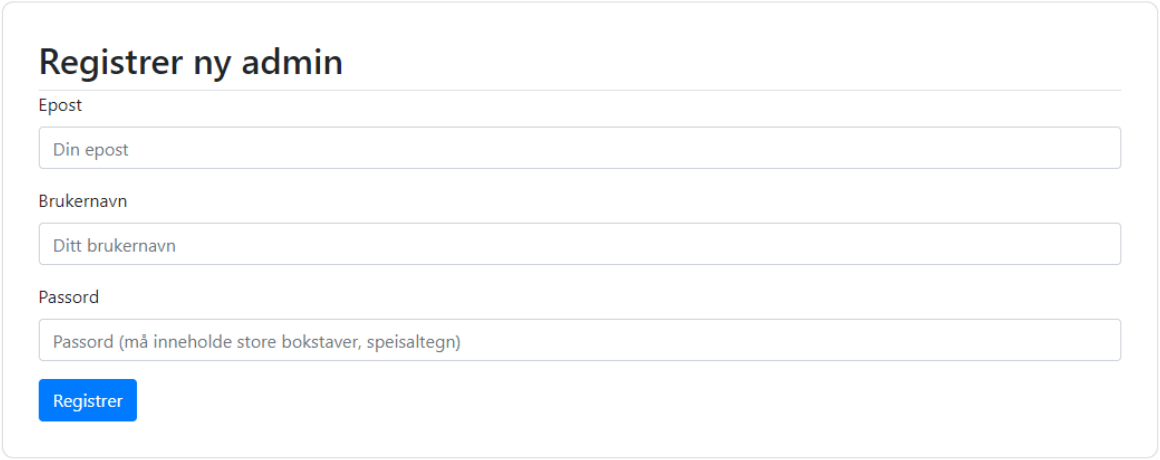
På venstre side er menyen for administratordelen plassert. Derfra kan det navigeres til alle sidene illustrert i Figur 56

Under «kandidater» (Figur 57) - fanen vil alle kandidatene som har utført testen bli listet opp, med mulighet til å slette kandidaten ved å trykke på søppelkasse ikonet til høyre i hver rad.



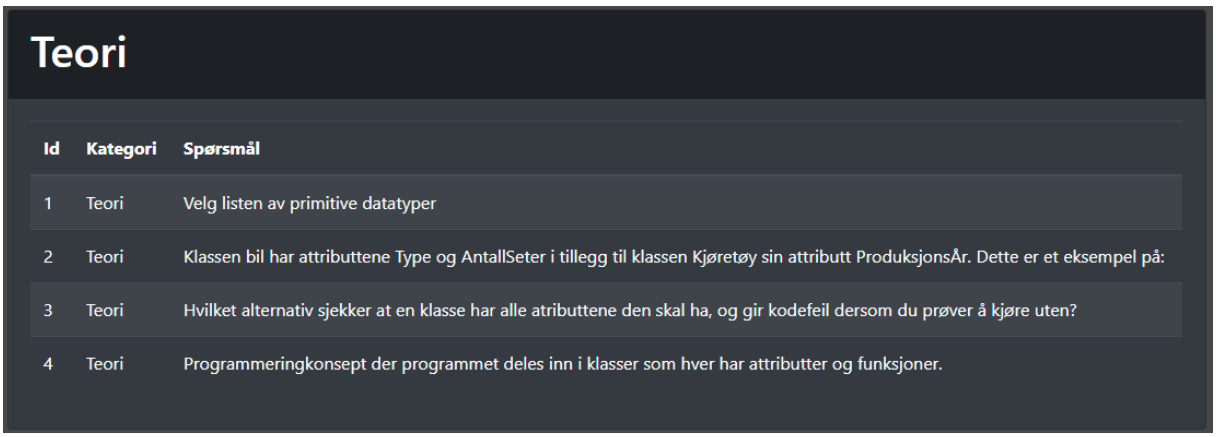
Figur 57 Viser alle kandidater i administrator siden

Under «brukere» - fanen kan nye administratorbrukere opprettes (Figur 58). Denne funksjonen er forbeholdt “super”-admin.

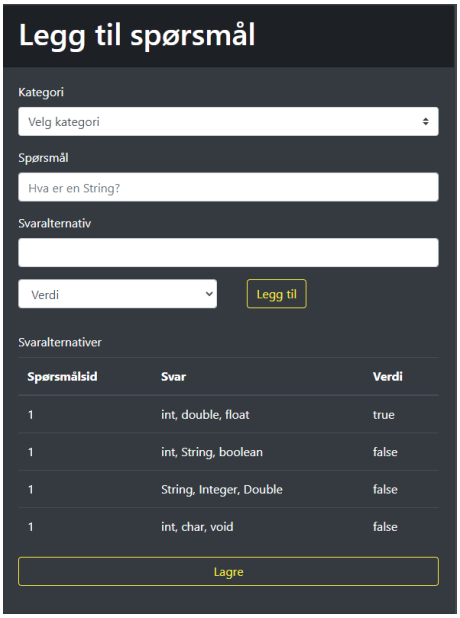


Figur 58: Oppretning av en ny admin.

I "Tester"- fanen (Figur 59) blir alle quizspørsmålene presentert. Spørsmålene som allerede er lagret blir presentert i tre separate lister, basert på kategori.



Figur 59: Tester fanen i adminsiden.



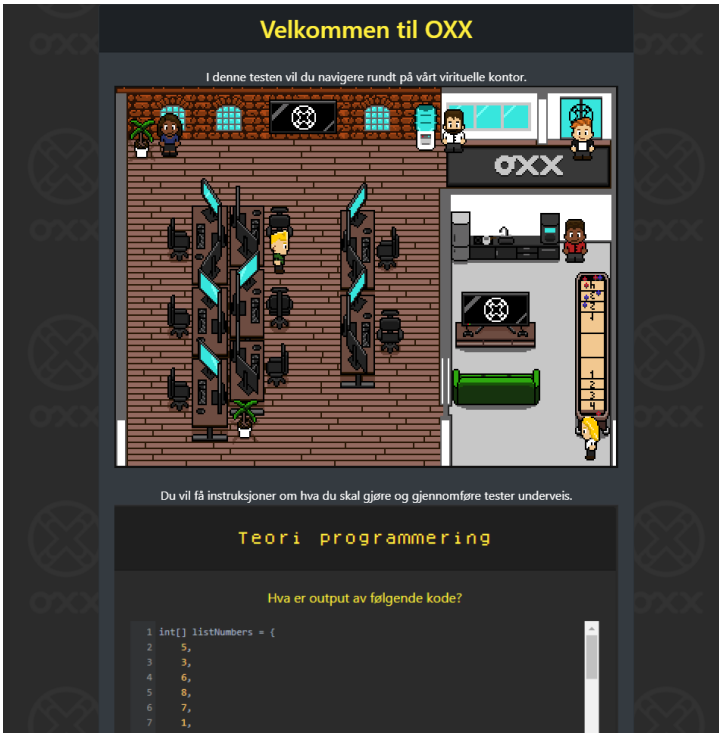
Figur 60: Legge til nye spørsmål.

På høyre del av siden vil det være mulig å opprette nye spørsmål (Figur 60). Her må Administratoren velge kategori, Fylle inn spørsmål, og legge til svaralternativer. Per alternativ er det mulig å registrere hvorvidt det svaret vil gi poeng til kandidater.

Fanene «Statistikk» og «Innstillinger» er det for øyeblikket ingenting under. Disse er ment for videre utvikling som kan leses om i Kapittel 9.

8.3 Kandidatdel

Ved bruk av pc for gjennomføring av spillet brukes piltastene på tastaturet for å bevege seg rundt og knappen Enter for å interagere med andre karakterer.
Ved gjennomføring på mobile enheter brukes joysticken på skjermen for å bevege deg rundt, og knappen ved siden av til å interagere med andre karakterer.
Det første du som kandidat vil møte er at du enten blir bedt om å gå til en spesifikk nettadresse eller får tilsendt en lenke på epost. Når kandidaten trykker på lenken eller går til den aktuelle nettadressen havner du på en landingside, Kandidaten ankommer først en landingsside (Figur 61). Her presenteres informasjon om hva som skjer videre. Hvordan de kan navigere seg rundt på kontoret til OXX, og hvordan evnetestene vil foregå forklares kort her. For å gå videre må kandidaten trykke på “start”-knappen nederst på siden.



Figur 61 Landingsside

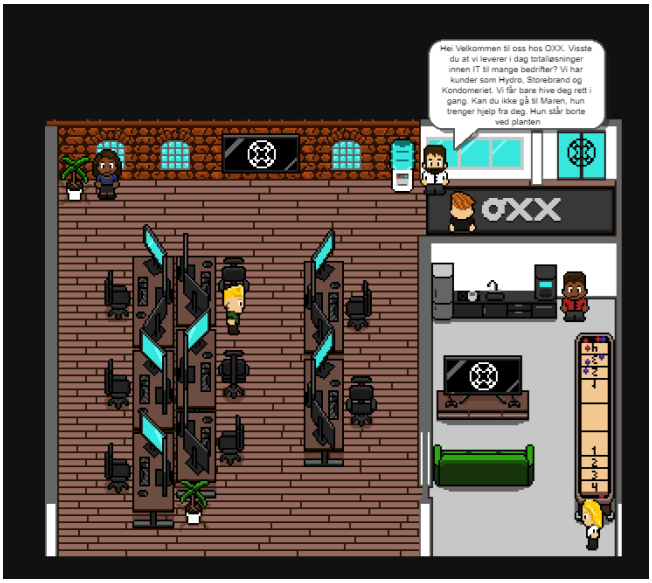
Da ankommer kandidaten en svart side, med OXX-logo. For å gå videre herfra må kandidaten trykke på “play”-knappen. Når musen plasseres på “play”-knappen, så vises spillkarakteren.(Figur 62)

Figur 62 Landing side spill

Etter å ha klikket på “play”-knappen starter spillet, og kandidaten blir plassert på kontoret til OXX. (Figur 63)

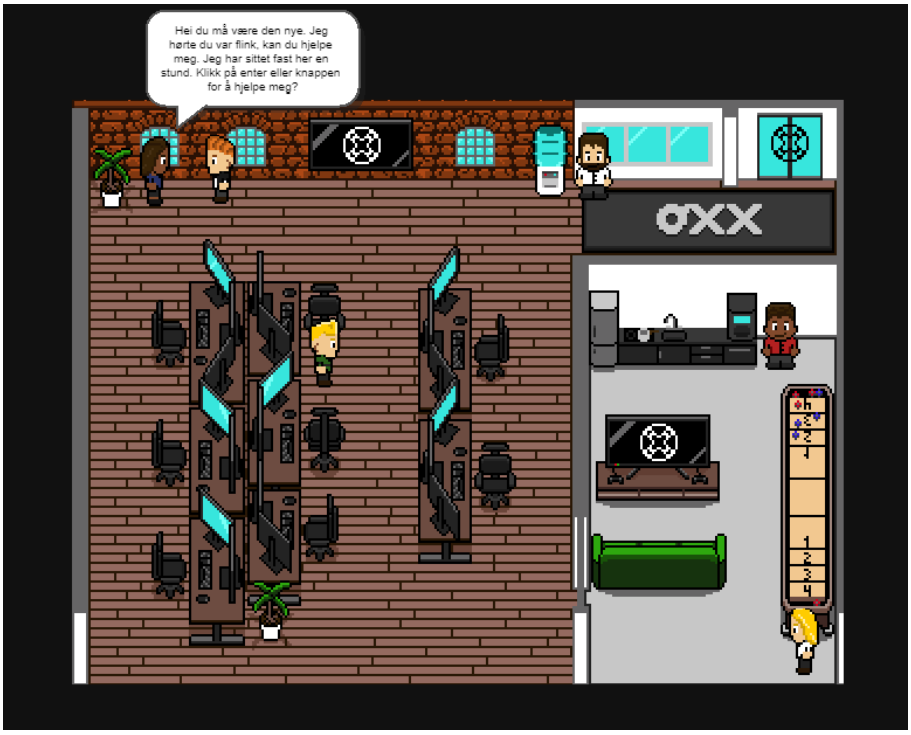
Figur 63 Spillkartet

Når kandidaten beveger deg til første karakter vises en snakkeboble med tekst, som introduserer OXX og sender kandidaten videre til karakteren opp i venstre hjørne (Figur 64)



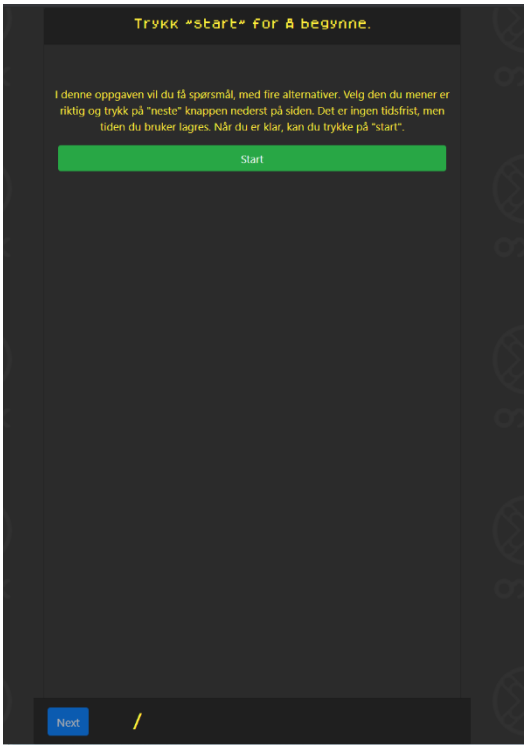
Figur 64 Introduksjon OXX

Her møter kandidaten karakteren Maren, hun har et problem og trenger hjelp for å løse dette (Figur 65). Kandidaten blir her bedt om å enten klikke på den faste knappen på skjermen, eller klikke på Enter-knappen på tastaturet. Dette sender kandidaten videre til første evnetest.



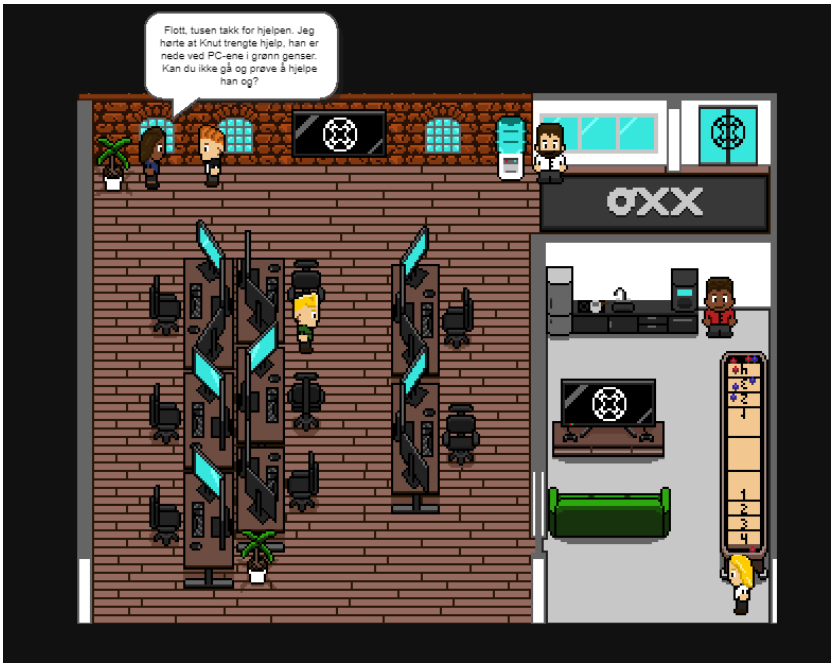
Figur 65 Dialog med Maren

Deretter ankommer kandidaten første evnetest, som handler om kodeforståelse. Her forklares testens gjennomføring (Figur 66). Hvert spørsmål har 3 eller 4 svaralternativer, der kun ett svar er riktig. Etter alle spørsmål er besvart lukkes quizen og kandidaten blir sendt tilbake til spillet.



Figur 66 Evnetest informasjon

Tilbake i spillet vil Marens dialog ha endret seg (Figur 67). Hun takker for hjelpen og sender kandidaten videre til Knut, som står nede ved pc'ene.



Figur 67 Marens dialog fase 2

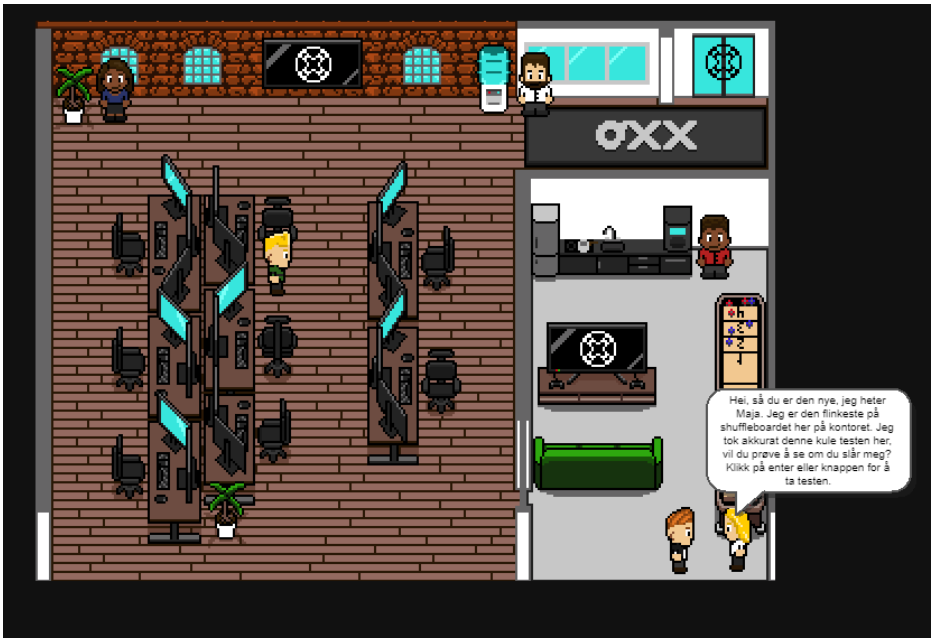
Når kandidaten ankommer Knut, blir vedkommende på nytt bedt om å trykke på den faste knappen på skjermen, eller klikke på Enter-knappen på tastaturet for å hjelpe han. (Figur 68)

Figur 68 Dialog Knut

Deretter gjennomføres en ny evnetest i samme stil som den første. Denne testen omhandler programmeringsteori. Kandidaten blir så informert om å bevege deg til karakteren Maja, nede til høyre, da hun ønsker å snakke med deg. (Figur 69)

Figur 69 Dialog Knut fase 3

Maja snakker kort om seg selv og ber kandidaten ta en test. (Figur 70) Denne er i samme stil som foregående tester, og omhandler systemutviklingsteori.

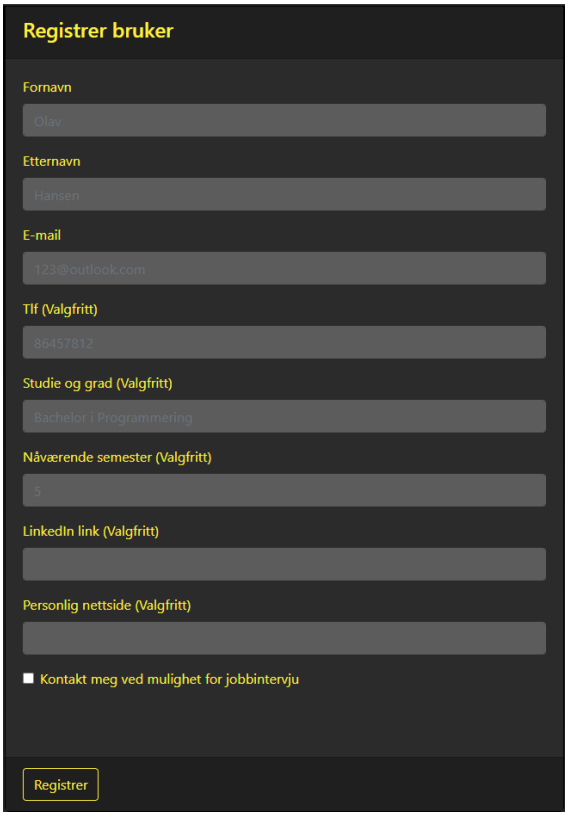


Figur 70 Dialog Maja

Etter testen sendes kandidaten tilbake til Morten ved inngangsdøren. Morten forteller at han trenger kandidatens personalia så han kan bestille ID-kort. Videre ber han kandidaten trykke på knappen eller enter, for å registrere seg. (Figur 71)

Figur 71 Avsluttning med Morten

Deretter ankommer kandidaten et registreringsskjema for utfylling av nødvendig informasjon, i tillegg til noen valgfrie felter. Siste spørsmål er en sjekkboks som kandidaten kan huke av hvis det er av interesse å bli kontakt ved mulighet for jobbintervju. (Figur 72)



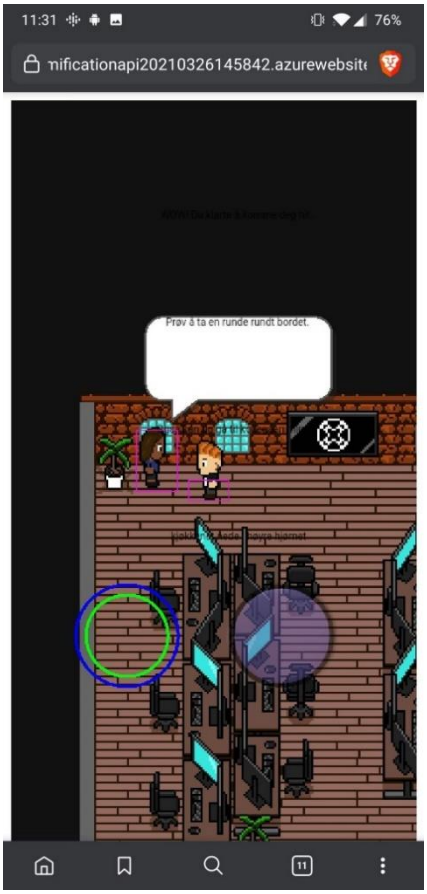
Figur 72 Registreringsskjema

Når kandidaten klikker på "registrer"-knappen. lastes en ny side (Figur 73) som takker for kandidatens søknad, og informerer om at OXX sin hjemmeside inneholder mer informasjon.

Figur 73: Takk for søknad på en ny fane.

8.4 Bugs og kjente feil

Det ble under brukertestingene funnet noen bugs og feil som ikke er prioritert fikset ved innlevering. Disse er dokumentert med bilde og tekst under.



Figur 74: Problem tekst som forsvinner.

Bruk av nettlesere på mobil som benytter reklameblokker, medførte i noen tilfeller at deler av javascriptkoden ikke ble kjørt. Dette medførte at teksten i snakkeboblene ikke alltid lastet inn riktig. (Figur 74).

Figur 75: Problem streker fra hjørnet.

På noen mobiler, oppsto streker fra snakkeboblene som strakk seg opp til venstre hjørne Figur 75 Årsaken til dette er ikke kjent, men det oppsto i hovedsak bare på noen få Android enheter.

9. Videreutvikling

Gamification av ansettelser

Oslomet – Gruppe 31 – vår 2021

Skrevet av

Christian Dyrli

Jørgen Borander

Maximilian Stiegler Sharoyan

Sondre Næbb Bjarkum

9.1 Forord

Denne delen tar for seg forslag om hvordan prosjektet kan videreutvikles.

9.2 Kartet

Kartet, slik det er i dag, representerer ikke kontorene til OXX på en god måte med tanke på planløsning. Dersom gruppen hadde hatt mulighet til å jobbe på kontorene til OXX og fått bedre forståelse av planløsningen, kunne kartet matchet kontoret bedre. Dette vil gi kandidater større areal å spille på og et mer representativt inntrykk av bedriften.

9.3 Spillermodeller

Kandidater kan ikke for øyeblikket velge spillermodell under gjennomføring av spillet. Dette er noe gruppen ønsker å endre på i fremtiden. Spillermodellen og karakterene kandidaten interagerer med underveis er designet i et forsøk på å representere mangfoldet av etnisiteter i Norge i dag. Dette er noe gruppen også ønsker å kunne representere i spillermodellene i fremtiden.

9.4 Språk

Gruppen ønsker også at kandidater skal ha mulighet til å endre språket i spillet, da norsk er det eneste språket som er implementert for øyeblikket.

9.5 Dashbord

Dashbordsiden er for øyeblikket begrenset til å presentere de 10 beste kandidatene basert på testresultater. Dette er noe vi ønsker å utvide med for eksempel grafer og lister. Disse kan bl.a. presentere resultater sammenlignet mellom hvilket semester kandidater har kommet til, eller hvilke tester kandidater presterer gjennomsnittlig best på.

9.6 Kandidater

Kandidatsiden er tenkt å inneholde mer detaljert informasjon om hver kandidat. Siden vil gi tilgang til kontaktinformasjon, øvrig personalia og ikke minst detaljert informasjon om testresultater.

9.7 Statistikk

Denne siden vil inneholde informasjon om hvilke tester, og spesifikke spørsmål, alle kandidater presterer enten bra eller dårlig på. Ulik statistikk på denne siden kan presenteres i forhold til øvrige detaljer om kandidater. For eksempel hvilket semester de er kommet til, kandidatens alder, tid brukt med mer. Denne siden er tiltenkt å gi OXX et bedre inntrykk av de forskjellige testene og om de burde justeres.

9.8 Brukere

På denne siden bør funksjonalitet implementeres slik at administratorer kan fjernes eller redigeres ved behov.

9.9 Tester

Testsiden i dag gir ikke mulighet til å legge til spørsmål i quizene. For fremtidig utvikling ønsker gruppen at denne funksjonaliteten skal implementeres. I tillegg bør mulighet til å redigere quizer så vel som enkeltspørsmål, samt slette, aktivere eller deaktivere spørsmål midlertidig implementeres.

9.10 Innstillinger

I fremtiden ønsker gruppen at siden skal kunne tilby flere fargepalletter, for eksempel dark/lightmode. I tillegg ønsker gruppen at brukere skal ha mulighet til å velge språk slik at ansatte som ikke snakker norsk skal ha et alternativ når de bruker siden.

9.11 Evnetester

I applikasjonsversjonen ved innlevering av oppgaven er alle evnetestene en form for quiz, med variasjon i hvilket tema som dekkes. Originalt var det tenkt å ha større variasjon i måten testing blir gjennomført på, men dette ble som nevnt, utelukket på bakgrunn av på tidsrestriksjoner. Dette er noe gruppen hadde ønsket å utbedre i videre utvikling.

9.12 Lydspor

Vi diskuterte kort hvorvidt vi ønsket å ha et lydspor i spillet. Vi kom frem til at et lydspor ville være mer forstyrrende enn noe annet. Dette begrunnes med at applikasjonens planlagte

bruk er på mobiltelefonen i områder der det hovedsakelig kun vil medføre støy, og vil være vanskelig å høre. Det er allikevel noe gruppen kunne ønsket å ha med i senere iterasjoner, med muligheten til å skru av og på lyden i brukergrensesnittet. Årsaken til dette er at spillet også er mulig å gjennomføre på pc, og/eller med øreproppper, og vi tror dette kan være med å gjøre opplevelsen mer underholdene. Dette er i utgangspunktet tenkt å være et lydspor i 8bit-stil, for å stå i stil med spillets øvrige design.

9.13 Ganghastighet

En annen idé gruppen har hatt, er å skru opp ganghastigheten til spilleren etter et par gjennomførte tester. Eksempelvis kan en av karakterene påpeke at det er tillatt å løpe på kontoret, eller gi kandidaten «løpesko» etterfulgt av økning av bevegelseshastighet.

10. Ordliste

Gamification av ansettelser

Oslomet – Gruppe 31 – vår 2021

Skrevet av

Christian Dyrli

Jørgen Borander

Maximilian Stiegler Sharoyan

Sondre Næbb Bjarkum

Adobe Illustrator

- Adobe Illustrator er et verktøy for redigering og vektorgrafing av bilder

API

- Programmeringsgrensesnitt

App Service

- Azure sin App Service er en webbasert hosting service hvor man kan bygge opp web applikasjoner, det er opprettet og drevet av Microsoft (Microsoft Azure, 2021).

Authorize

- Autorisering

Backlog

- Gjenstående oppgaver i løpet av et prosjekt

Branch, Brancher

- Aspekt av versjonshåndteringsverktøy som gjør det mulig å jobbe videre med et prosjekt, uten risiko for å ødelegge deler som allerede fungerer. Endringer gjort i en branch blir ikke overført til andre brancher før en eventuell merge.

Brukergrensesnitt

- Det visuelle uttrykket en bruker interagerer med

C#

- C# er et objektorientert programmeringsspråk utviklet av Microsoft.

Code coverage

- Hvor stor andel av koden som er testet i enhetstesting

CRUD-modellen (Create, Read, Update and Delete)

- HTTP modeller

CSS

- Cascading Style Sheet brukes til design av nettsider. Det kan blant annet spesifisere farger, rammer, skriftstørrelse og plassering av et objekt på en nettside (Wikipedia, 2021B).

Eager loading

- En metode å hente relatert data

Facebook Messenger

- Facebook Messenger er en chattetjeneste levert av Facebook, som lar brukere skrive og sende tekstmeldinger til hverandre.

Firewalls and virtual networks, Azure

- Brannmurer og virituelle nettverk

Gamifisert

- Gamifisert vil si at det noe gjøres om slik at det ser ut som et spill, eller at noe blir mer likt som et spill, selv om det gjerne egentlig ikke er et spill.

Hashing

- Måte å sikre passord på

HTML

- HyperText Markup Language er et "markeringsspråk", og brukes til formatering av nettsider. språket inneholder bl.a. spesifikasjoner for hypertekst, titler og lister (Wikipedia, 2021C).

HiFi-prototype

- Hi-Fi står for High-Fidelity. En HiFi-prototype er en prototype som skal være så nærme det endelige produktet som mulig, med mest mulig funksjonalitet implementert. For eksempel bevegelse, interaksjoner i applikasjonen.

Hitbox

- En usynlig barriere rundt en figur

Interface

- Definerer at en metode må ha gitt metoder

Javascript

- Javascript er et programmeringsspråk designet for frontend-utvikling. Gjennom biblioteker som ReactJS og Node.js kan det brukes til å designe høyst dynamiske og reaktive nettsider (Wikipedia, 2020 A)

Json

- Json er en forkortelse for javascript object notation. Det er standard måten man formatere tekstbaserte dokumenter som skal brukes til utveksling av data. Json støtter følgene type data: tall, tekst, boolske verdier, tabeller, objekter og tomme verdier (W3schools, 2021).

JWT Bearer token

- Representerer en sikkerhetsnøkkel for identifisering

Kanban

- En lean metodikk

Key

- En «key» er brukt til å lagre digital informasjon og er nødvendig for å få tilgang til en spesifikk data.

Klasse

- Klasser er en av de grunnleggende bygge blokkene i objekt orientert programmering. En klasse er arbeidstegning for opprettelse av objekter som har spesifikke verdier og metoder.

Lazy loading

- En måte å laste relaterte data

LinkedIn

- LinkedIn er et sosialt nettverk tilrettelagt for deling av brukeres utdanning, arbeidserfaring, sertifiseringer etc.

Merge

- Sammenslåing av kode fra separate brancher.

Metode

- En metode er en blokk kode som gjør en spesifikk oppgave.

Microsoft Teams

- Microsoft Teams er et gruppe-/samarbeidsverktøy levert av Microsoft. Verktøyet er designet for å forenkle arbeid der gruppemedlemmer ikke har mulighet til å jobbe fra samme lokasjon. Applikasjonen inkluderer funksjonalitet som muligheten til å skrive i samme dokument samtidig med liveoppdatering, videosamtaler, tekstchat, samt lagring og deling av filer. I tillegg gir Teams muligheten for å opprette, og være medlem av, flere grupper (Microsoft, 2021).

Microsoft SQL Server

- En SQL database

Microsoft SQL Server Management

- Program for behandling av databaser

Middleware

- Mellomprgramvare

Mocking

- Å isolere og forfalske et programs funksjoner for testing

MVC (Model View Controller)

- En programvare design mønster

Object-relational mapping (O/RM)

- En teknikk som konverterer data mellom ikke kompatible systemer

Plugin

- Plug-in er en utvidelse av programmet som legger til nye funksjoner uten store endring av programmet i seg selv.

Pokemon

- Spillserie i fulgeperspektiv der spilleren fanger monstre, og kjemper med disse.

Publish

- Publishing eller publisering på norsk er når man laster opp programmet sitt på internett.

Regulære uttrykk

-

Release

- Utgivelsesversjonen av programvare

Resource group

- En gruppe i Azure

REST

- En arkitekturstil innen programmering

Scenene

- Scenes er en måte å samle komponenter av applikasjonen som brukes i en spesifikk del av applikasjonen. F.eks. i vår applikasjon ligger alt som kun brukes i top down spillet i PlayScene, og alt som må lastes på forhånd i LoadScene.

Scope

- Planlagt omfang på prosjekt

Scriptet

- Et datamaskinscript er en rekke kommandoer som blir brukt til å automatisere en prosess på en lokal maskin eller websider på nettet.

SCRUM

- En smidig utviklingsprosess

Session

- Mellomlagring av tokens for senere bruk

Sprint

- Er en kort tidsperiode på opptil en måned hvor utviklingsteamet skal gjøre ferdig et fastsatt oppgaver.

Sprites

- En sprite er en piksel grafikk som er designet til å være del av en større scene. Det kan være et statisk bilde eller en animert grafikk.

Teknologistack

- Teknologistack er en betegnelse på en samling av teknologier og programmeringsspråk, som blir benyttet av en bedrift.

Test suite

- Der testene ligger lagret i programmet.

Token

- Representerer en sikkerhetsnøkkel for identifisering.

Tredje normalform (3NF)

- En måte å utforme databaser

Visual Studio 2019

- Fullverdig kodeeditor fra Microsoft med funksjonalitet for koding, feilsøking, testing, kodefullføring med mer.

Visual Studio Code

- Kodeeditor med støtte for liveoppdatering av forhåndsvisning. Editoren gjør det enkelt å teste endringer av både design og funksjonalitet underveis.

WCAG (Web Content Accessibility Guidelines)

- WCAG er en forskrift med regelverk og krav om universell utforming av nettløsninger og automater. Regelverket har til hensikt å skape en felles standard for nettbaserte løsninger. Denne skal gjøre innhold og funksjonalitet tilgjengelig for flest mulig, uavhengig av et individs eventuelle funksjonsnedsettelser, samt møte behovene til individer, organisasjoner og myndigheter på internasjonal basis (Henry, 2021).

Webapplikasjon

- En webapplikasjon er et program som kan kjøres igjennom en nettleser, applikasjonen er vanligvis sammensatt av forskjellige kodespråk.

xUnit

- Et verktøy for testing.

Zelda

- Spillserie i fugleperspektiv der spilleren utforsker og samler utstyr nødvendig for å redde prinsessen sin.

11. Appendiks

Gamification av ansettelser

Oslomet – Gruppe 31 – vår 2021

Skrevet av

Christian Dyrli

Jørgen Borander

Maximilian Stiegler Sharoyan

Sondre Næbb Bjarkum

11.1 Forord

I dette dokumentet finner vi brukertestene vedlagt.

11.2 Brukertester

Nedenfor finner du alle brukertestene som var utført, hva tekst som ble gitt til brukerne. Hvordan tenktfremgangsmåten skulle være og spørsmålene brukerne ble stilt etter fullført bruker testing. For å se resultatene, endringer og tilbakemeldinger av bruker testen må du bevege deg opp til prosessdokumentasjonen.

11.2.1 Første brukertest

Tenkt fremgangsmåte på første brukertest

Intro tekst

Velkommen til testing av spillet vi har laget som er en del av vårt bachelorprosjekt med OXX. Når du starter spillet ønsker vi at du beveger deg til nærmeste person og følge instruksene denne karakteren gir deg.

Under testen ønsker vi at du legger spesielt godt merke til hvor brukervennlig du opplever programmet og noterer deg dine inntrykk underveis (dette kommer til gode på spørreskjemaet).

Framgangsmåte i testen

- Når bruker klikker på linken, havner bruker i på en sort skjerm hvor det står begynn spill.
- Etter klikket på start spill havner brukeren inn i spillet. Her blir bruker fortalt at de skal bevege seg fra punkt A til punkt B.
- Etter ankommet punkt B blir brukeren bedt om å gå til punkt C, der det er en figur.
- Figuren sender ut en snakkeboble som ber deg klikke på en knapp som sender deg videre til et google dokument.

- I dokumentet vil det være et spørreskjema med seks spørsmål som skal prøve å få klarhet i hvor intuitivt spillet var og hvordan opplevelsen og den visuelle delen rundt opplevdes.

Brukerne ble stilt følgende spørsmål:

- Fra en skala fra 1-10. Hvor lett eller vanskelig opplevde du det å bevege deg rundt i spillet?
- Fra en skala fra 1-5 hvor oversiktlig følte du spillet var?
- Opplevde du noen problemer med brukergrensesnittet? Ja/Nei
- Hvis Ja, hvilke problemer opplevde du?
- På en skala fra 1-10, hvor lett eller vanskelig opplevde du det var å gjennomføre testen?

11.2.2 Andre bruker test

Intro tekst

Velkommen til testing av spillet vi har laget som er en del av vårt bachelorprosjekt med OXX. Når du starter spillet ønsker vi at du beveger deg til nærmeste person og følge instruksene denne karakteren gir deg.

Du vil gjennom denne testen bli sendt til flere sorte skjermer med en knapp. På disse skjermene trykker du på knapper for å fortsette.

Under testen ønsker vi at du legger spesielt godt merke til hvor brukervennlig du opplever programmet, hvordan du opplever hva karakterene sier, og noterer deg dine inntrykk underveis (dette kommer til gode på spørreskjemaet).

Framgangsmåte i testen

- Bruker får tilsendt en lenke hvor de blir spurt om de ønsker å ta en bruker test.
- Bruker klikker på lenken og får en introtekst hvor det står om brukertesten

- Bruker klikker videre og havner inn i spillet
- Bruker blir bedt om å bevege seg til en karakter for å så klikke på Space eller benytte seg av knappen på skjermen.
- Bruker blir etter det bedt å bevege seg til ny karakter
- Bruker blir bedt om å bevege seg til en karakter for å så klikke på Space eller benytte seg av knappen på skjermen.
- Bruker blir etter det bedt å bevege seg til ny karakter
- Bruker blir bedt om å bevege seg til en karakter for å så klikke på Space eller benytte seg av knappen på skjermen.
- Bruker blir etter det bedt å bevege seg til ny karakter
- Figuren sender ut en snakkeboble som ber deg klikke på en knapp som sender deg videre til et google dokument.
- I dokumentet vil det være et spørreskjema med spørsmål som skal prøve å få klarhet i hvor intuitivt spillet var og hvordan opplevelsen og den visuelle delen rundt opplevdes. Og hvilken enhet som blir benyttet for å løse brukertesten

Brukerne ble stilt følgende spørsmål:

1. Fra en skala fra 1-10 hvor 1 er "veldig vanskelig" og 10 er "ingen problemer", hvordan opplevde du å bevege deg rundt i spillet?
2. Fra en skala fra 1-5, hvor 1 er "Veldig uoversiktlig" og 5 er "Veldig oversiktlig", hvor oversiktlig opplevde du spillet var?
3. Opplevde du noen problemer med brukergrensesnittet Ja/Nei
4. Hvis ja. Hvilke problemer opplevde du?

5. Fra en skala fra 1-10. Hvor lett eller vanskelig opplevde du det var å gjennomføre testen?
6. Benyttet du PC eller mobil for å gjøre denne brukertesten?
7. Hvis mobil, hvilken mobil telefon benyttet du deg av? Merk og modell. Eks Iphone 11 pro
8. Hvilken nettleser benyttet du til å utføre denne brukertesten? eks Chrome, Firefox, Edge, Safari
9. Benyttet du deg av en Adblock/annonseblokker i nettleseren som var brukt?
10. Har du noen andre tilbakemeldinger?

11.2.3 Tredje bruker test

Intro tekst

Velkommen til testing av spillet vi har laget som er en del av vårt bachelorprosjekt med OXX. Benytt mobil ved gjennomføring av denne brukertesten. Når du starter spillet ønsker vi at du beveger deg til nærmeste person og følge instruksene karakterene gir deg.

Du vil gjennom denne testen bli sendt til flere sorte skjermer med en knapp. På disse skjermene trykker du på knapper for å fortsette.

Under testen ønsker vi at du legger spesielt godt merke til hvor brukervennlig du opplever programmet, hvordan du opplever hva karakterene sier, og noterer deg dine inntrykk underveis (dette kommer til gode på spørreskjemaet).

Framgangsmåte i testen

- Bruker får tilsendt en lenke hvor de blir spurt om de ønsker å ta en bruker test.
- Bruker klikker på lenken og får en introtekst hvor det står om brukertesten

- Bruker klikker videre og havner inn i spillet
- Bruker blir bedt om å bevege seg til en karakter for å så klikke på Space eller benytte seg av knappen på skjermen.
- Bruker blir etter det bedt å bevege seg til ny karakter
- Bruker blir bedt om å bevege seg til en karakter for å så klikke på Space eller benytte seg av knappen på skjermen.
- Bruker blir etter det bedt å bevege seg til ny karakter
- Bruker blir bedt om å bevege seg til en karakter for å så klikke på Space eller benytte seg av knappen på skjermen.
- Bruker blir etter det bedt å bevege seg til ny karakter
- Figuren sender ut en snakkeboble som ber deg klikke på en knapp som sender deg videre til et google dokument.
- I dokumentet vil det være et spørreskjema med spørsmål som skal prøve å få klarhet i hvor intuitivt spillet var og hvordan opplevelsen og den visuelle delen rundt opplevdes. Og hvilken enhet som blir benyttet for å løse brukertesten

Brukerne ble stilt følgende spørsmål:

- Fra en skala fra 1-10 hvor 1 er "veldig vanskelig" og 10 er "ingen problemer", hvordan opplevde du å bevege deg rundt i spillet?
- Fra en skala fra 1-10, hvor 1 er "Veldig uoversiktlig" og 10 er "Veldig oversiktlig", hvor oversiktlig opplevde du spillet var?
- Opplevde du noen problemer med brukergrensesnittet Ja/Nei
- Hvis ja. Hvilke problemer opplevde du?

- Fra en skala fra 1-10. Hvor lett eller vanskelig opplevde du det var å gjennomføre testen?
- Hvilken mobil telefon benyttet du deg av? Merke og modell. Eks Apple iPhone 11
- Hvilken nettleser på mobilen benyttet du til å utføre denne brukertesten? eks Chrome, Firefox, Edge, Safari
- Benyttet du deg av en Adblock/annonseblokker i nettleseren som var brukt?
- Har du noen andre tilbakemeldinger?

11.2.4 Fjerde bruker test, Hypotesetesting. Spillifiserte evnetester.

Intro tekst

Velkommen til testing av spillet vi har laget som er en del av vårt bachelorprosjekt med OXX. Når du starter spillet ønsker vi at du beveger deg til nærmeste person og følge instruksene denne karakteren gir deg.

Under testen ønsker vi at du legger spesielt godt merke til hvor Engasjerende du opplever programmet og noterer deg dine inntrykk underveis (dette kommer til gode på spørreskjemaet).

Framgangsmåte i testen

- Bruker får tilsendt en lenke hvor de blir spurt om de ønsker å ta en bruker test.
- Bruker klikker på lenken og får en introtekst hvor det står om brukertesten
- Bruker klikker videre og havner inn i spillet

- Bruker blir bedt om å bevege seg til en karakter for å så klikke på Space eller benytte seg av knappen på skjermen.
- Bruker utfører evnetest kategori programmerings teori
- Bruker blir etter det bedt å bevege seg til ny karakter
- Bruker blir bedt om å bevege seg til en karakter for å så klikke på Space eller benytte seg av knappen på skjermen.
- Bruker utfører evnetest kategori Kode forståelse
- Bruker blir etter det bedt å bevege seg til ny karakter
- Bruker blir bedt om å bevege seg til en karakter for å så klikke på Space eller benytte seg av knappen på skjermen.
- Bruker utfører evnetest kategori Systemutviklings teori
- Bruker blir etter det bedt å bevege seg til ny karakter
- Figuren sender ut en snakkeboble som ber deg klikke på en knapp som sender deg videre til et google dokument.
- Bruker blir så sendt videre til et spørreskjema med spørsmål som skal prøve å finne ut hvor engasjerende spillet var og hvor vanskelig evnetestene opplevdes.

Brukerne ble stilt følgende spørsmål:

- Fra en skala fra 1-10 hvor 1 er "veldig engasjerende" og 10 er "lkke engasjerende", opplevde du spillet?
- Fra en skala fra 1-10, hvor 1 er "Veldig vanskelig" og 10 er "Veldig lett", hvor vanskelig opplevde evnetestene/quizen?
- Opplevde du noen problemer med spillet Ja/Nei

- Hvis ja. Hvilke problemer opplevde du?
- Fra en skala fra 1-10. Hvor lett eller vanskelig opplevde du det var å gjennomføre testen?
- Har du noen andre tilbakemeldinger?

11.2.5 Fjerde bruker test, Hypotesetesting. Evnetester.

Intro tekst

Velkommen til testing av evnetester vi har laget som er en del av vårt bachelorprosjekt med OXX. Nedenfor finner du tre linker til tre forskjellige evnetester som skal gjennomføres.

Under testen ønsker vi at du legger spesielt godt merke til hvor Engasjerende du opplever testene og noterer deg dine inntrykk underveis (dette kommer til gode på spørreskjemaet).

Framgangsmåte i testen

- Bruker får tilsendt en lenke hvor de blir spurt om de ønsker å ta en bruker test.
- Bruker klikker på lenken og får en introtekst hvor det står om brukertesten
- Bruker klikker på en lenke for å starte evnetestene
- Brukeren gjør og fullfører evnetest programmerings teori
- Bruker går ut og velger neste evnetest
- Bruker fullfører evnes testen kode forståelse
- Bruker går ut og velger neste evnetest
- Bruker fullfører evne testen systemutviklings teori

- Bruker blir så sendt videre til et spørreskjema med spørsmål som skal prøve å finne ut hvor engasjerende og vanskelig evnetestene var

Brukerne ble stilt følgende spørsmål:

1. Fra en skala fra 1-10 hvor 1 er "veldig engasjerende" og 10 er "lkke engasjerende", opplevde du evnetestene?
2. Fra en skala fra 1-10, hvor 1 er "Veldig vanskelig" og 10 er "Veldig lett", hvor vanskelig opplevde evnetestene/quizen?
3. Opplevde du noen problemer med utføring av brukertestingen Ja/Nei
4. Hvis ja. Hvilke problemer opplevde du?
5. Har du noen andre tilbakemeldinger?

12. Referanser

Gamification av ansettelser

Oslomet – Gruppe 31 – vår 2021

Skrevet av

Christian Dyrli

Jørgen Borander

Maximilian Stiegler Sharoyan

Sondre Næbb Bjarkum

§1, F. o.-l. (2019, 10. oktober). *Forskrift om universell utforming av informasjons- og kommunikasjonsteknologiske (IKT)-løsninger (FOR-2020-10-16-2063).* Hentet fra https://lovdata.no/dokument/SF/forskrift/2013-06-21-732?q=wcag

Automation Panda. (2021, 20. Mai). *ARRANGE-ACT-ASSERT: A PATTERN FOR WRITING GOOD TESTS*. Hentet fra automationpanda: https://automationpanda.com/2020/07/07/arrange-act-assert-a-pattern-for-writing-good-tests/

Copes, F. (2018, 11. oktober). *JWT authentication: When and how to use it*. Hentet 10. mai, 2021 fra LogRocket: https://blog.logrocket.com/jwt-authentication-best-practices/

Cornett, S. (2021, 15. Mai). *Minimum Acceptable Code Coverage*. Hentet fra bullseye: https://www.bullseye.com/minimum.html#:~:text=Code%20coverage%20of%2070%2D80,higher%20than%20for%20system%20testing.

Degges, R. (2017, 17. august). *Why JWTs Suck as Session Tokens*. Hentet 3. februar, 2021 fra okta Developer.

EntityFrameworkTutorial. (2020). *Context Class in Entity Framework*. Hentet 9. april, 2021 fra EF Basic: https://www.entityframeworktutorial.net/basics/context-class-in-entity-framework.aspx

Hasan, F., Anderson, R., & Smith, S. (2019, 12. mai). *Prevent Cross-Site Request Forgery (XSRF/CSRF) attacks in ASP.NET Core*. Hentet 10. mai, 2021 fra Microsoft Documentation: https://docs.microsoft.com/en-us/aspnet/core/security/anti-request-forgery?view=aspnetcore-5.0

Henry, S. L. (2021, 29. april). *Web Content Accessibility Guidelines (WCAG) Introduction.* Hentet 19. mai, 2021 fra Web Accessibility Initiative: https://www.w3.org/WAI/standards-guidelines/wcag/#intro

Lerman, J. (2020, 15. desember). *Entity Framework Core: Getting Started*. Hentet 4. mars, 2021 fra Pluralsight: Entity Framework Core: Getting Started

Microsoft. (2021). *Microsoft Teams*. Hentet 23. mars, 2021 fra https://www.microsoft.com/nb-no/microsoft-teams/group-chat-software

Microsoft Azure. (2021). Hentet 12. mai, 2021 fra Nettsted for Microsoft Azure: https://azure.microsoft.com/nb-no/

Microsoft Azure. (2021). *App Service*. Hentet 9. mai, 2021 fra https://azure.microsoft.com/en-in/services/app-service/

Molchan, A. (2021, 15. April). *Stroke rounded rect visual artifacts*. Hentet fra Github: https://github.com/photonstorm/phaser/issues/3955?fbclid=IwAR3FynwZ92shkIv2pAJ1ZWAOnrP67ovQNwg7w23dTmFbpjSyH3xFG1355PY

Open Trivia. (2021). *Trivia API*. Hentet 7. mars, 2021 fra Open Trivia Database: https://opentdb.com/api_config.php

OXX. (2021). *Forside*. Hentet 20. mai, 2021 fra Oxx: https://www.oxx.no/

Pal, D. (2013, 8. november). *Understanding Three Layer Architecture and its Implementation in C# .NET*. Hentet 2. mars, 2021 fra Code Project: https://www.codeproject.com/Articles/679185/Understanding-Three-Layer-Architecture-and-its

Petrova, S. (2019, 18. januar). *Adopting Agile: The Latest Reports About The Popular Mindset*. Hentet 28. april, 2021 fra Adeva: https://adevait.com/blog/remote-work/adopting-agile-the-latest-reports-about-the-popular-mindset#:~:text=Scrum%20Statistics%20You%20Should%20Know%20About&text=The%20reason%20for%20Scrum's%20popularity,used%20by%2083%25%20of%20companies

Plow Games. (2021). *Plow Games*. Hentet 4. februar, 2021 fra www.plowgames.com

rexrainbow. (2021). *Virtual Joystick*. Hentet 13. mars, 2021 fra Notes of Phaser 3: https://rexrainbow.github.io/phaser3-rex-notes/docs/site/virtualjoystick/

Saseendran, S. (2020, 19. september). *Authentication And Authorization In ASP.NET Core Web API With JSON Web Tokens*. Hentet 9. mai, 2021 fra C-Sharpcorner: https://www.c-

sharpcorner.com/article/authentication-and-authorization-in-asp-net-core-web-api-with-json-web-tokens/

Tiled. (2021). *Introduction*. Hentet 2. mai, 2021 fra Documentation: https://doc.mapeditor.org/en/stable/manual/introduction/

Tilsynet for universell utforming av ikt. (2021). *Gjeldene regelverk og krav*. Hentet 19. mai, 2021 fra Uutilsynet: https://www.uutilsynet.no/regelverk/gjeldende-regelverk-og-krav/746

W3schools. (2021). *JSON - Introduction*. Hentet 4. mai, 2021 fra W3schools: https://www.w3schools.com/js/js_json_intro.asp

Walpita, P. (2019, 9. juli). *Software Architecture Patterns — Layered Architecture*. Hentet 14. april, 2021 fra Priyal Walpita: https://priyalwalpita.medium.com/software-architecture-patterns-layered-architecture-a3b89b71a057

White, C. (2015, 3. Desember). *5 Companies That Are Successfully Using Gamification for Recruiting*. Hentet Februar 11, 2021 fra Linkedin: https://www.linkedin.com/business/talent/blog/talent-strategy/companies-successfully-using-gamification-for-recruiting?fbclid=IwAR2iwiyf-XltXMyHm0a_2jSOHUM7tXk9_JB1UkHvufeFM_UvvKCLHZlKba0

Wikipedia A. (2020, 15. april). *JavaScript*. Hentet 4. april, 2021 fra Wikipedia: https://no.wikipedia.org/wiki/JavaScript

Wikipedia B. (2021, 3. mars). *Cascading Style Sheets*. Hentet 20. mars, 2021 fra Wikipedia: https://no.wikipedia.org/wiki/Cascading_Style_Sheets

Wikipedia C. (2021, 30. mars). *HTML*. Hentet 28. januar, 2021 fra Wikipedia: https://no.wikipedia.org/wiki/HTML

Wikipedia D. (2021, 23. mai). *Model–view–controller*. Hentet 23. mai, 2021 fra Wikipedia: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

Wildermuth, S. (2018, 28. juli). *Avoid Lazy Loading in ASP.NET*. Hentet 19. april, 2021 fra Wildermuth: https://wildermuth.com/2018/07/28/Avoid-Lazy-Loading-in-ASP-NET).