Høgskulen på Vestlandet

BACHELOROPPGAVE

Utvikling av menyapplikasjon for CanCannotEat

Development of menu application for CanCannotEat

Dataingeniør

Institutt for data- og realfag

Fakultet for ingeniør- og naturvitenskap

Innleveringsdato: 03.06.2019

Antall ord: 7667

Karoline Skylstad, Jone Træet og Stian Finjord

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

Rapportens tittel:	*Dato:*
Utvikling av menyapplikasjon for CanCannotEat Development of menu application for CanCannotEat	*03.06.2019*
Forfatter(e):	*Antall sider u/vedlegg: 45*
Karoline Skylstad, Jone Træet, Stian Finjord	*Antall sider vedlegg: 1*
Studieretning:	*Antall disketter/CD-er:*
Dataingeniør	ingen
Kontaktperson ved studieretning:	*Gradering:*
Sven-Olai Høyland	ingen
Merknader:	

Oppdragsgiver:	*Oppdragsgivers referanse:*
CanCannotEat	Ingen
Oppdragsgivers kontaktperson:	*Telefon:*
Cecilie Hauge Ågotnes	913 67 619

Sammendrag:

Målet for prosjektet er å utvikle en menyapplikasjon for CanCannotEat som skal gjøre det enklere for personer med behov for tilrettelagt mat.

Stikkord:

Progressiv webapplikasjon (PWA)	Vue.js	Laravel Nova	Laravel

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

Forord

Denne rapporten dokumenterer arbeidet på prosjektet "Utvikling av menyapplikasjon for CanCannotEat" ved Høgskulen på Vestlandet (HVL). Prosjektet ble gjennomført våren 2019 av Karoline Skylstad, Jone Træet og Stian Finjord.

Vi vil først og fremst takke Cecilie Hauge Ågotnet, Renate Robberstad og Hildegunn Lid Meidell fra CanCannotEat, som har gitt oss muligheten til å gjennomføre denne Bacheloroppgaven. Vi vil også takke Nora Modesta Hvide for gode innspill gjennom perioden. Oppgaven har vært både spennende og utfordrende, som har resultert i at vi har tilegnet oss mye lærdom vi vil ta med oss videre etter endt studie.

Vi vil takke vår veileder fra Høgskulen på Vestlandet, Sven-Olai Høyland, for god hjelp og veiledning i arbeidet med skriving av Bacheloroppgaven.

Jone Træet

Karoline Skylstad

Stian Finjord

Innhold

Terminologi

API – Application Programming Interface, et sett metoder brukt til kommunikasjon mellom tjeneste og annen programvare.
Android – Operativsystem for mobil utviklet av Google.
Backend – Del av program som ikke er direkte tilgjengelig for brukeren av programmet.
Brukergrensesnitt – Middel som tillater bruker og datamaskin å interagere.
Database – Strukturert datamengde i datamaskin, ofte tilgjengelig på flere måter.
Databasespørring – Instruksjon til database.
Dokumentasjon – Brukerhåndbok som forklarer hvordan noe er ment til å brukes.
ER-diagram – Entity Relationship diagram, diagram som illustrerer forhold mellom entiteter.
Gantt-diagram – Fremdriftsplan brukt til prosjektadministrasjon.
GUI-verktøy – Graphical User Interface-verktøy, verktøy med grafisk brukergrensesnitt.
Git – Versjonskontrollverktøy som brukes til deling av kode.
GitHub – Webplattform for Git.
iOS – Operativsystem for mobil utviklet av Apple.
Kildekode – Tekstfil med kode som en datamaskin kan lese.
Kryssplattform programvare – Programvare tilpasset flere plattformer, som mobil og web.
Mange-til-mange-forhold – Relasjonsforhold i database hvor flere av elementene i en tabell peker på flere elementer i en annen tabell.
Mobilapplikasjon – Applikasjon som kjører på mobiloperativsystemer som Android og iOS.
MySQL – Databasehåndteringsystem.
Open source – Produktets kildekode, design og innhold er tillatt for alle å bruke.
REST-struktur - Står for Representational State Transfer. Arkitekturisk stil eller designmønster.
Rammeverk – Struktur som hjelper med å utvide en eksisterende løsning.

Server – Dataprogram som tilbyr tjenester for andre dataprogrammer.
SQL – Structured Query Language, spørrespråk som brukes til å kommunisere med en database.
Webapplikasjon/webapp – Applikasjon som kjøres på ekstern server, oftest tilgjengelig med nettleser.
EXPO - Utstilling for bachelor- og masteroppgaver for avgangsstudentene ved Høgskulen på Vestlandet (Høgskulen på Vestlandet, 2019).

1 Introduksjon

Dette kapittelet er en introduksjon til oppgaven, hvor det gis kontekst til hva prosjektet går ut på. For enkelhetsskyld vil allergier, intoleranser og andre matpreferanser, bli referert som matpreferanser gjennom rapporten.

1.1 Mål og motivasjon

Målet med prosjektet er å hjelpe personer som trenger tilrettelagt mat, ved å lettere spesialtilpasse menyer som samsvarer med deres matpreferanser.

CanCannotEat har allerede utviklet en applikasjon for arrangementer, hvor brukere kan registrere sine matpreferanser. Arrangementansvarlig vil få en sortert liste av matpreferanser på arrangementet (som vist i Figur 2). Et stort delmål vil være å implementere menyapplikasjonen i den eksisterende løsningen. Å integrere dette i et eksisterende system vil bety at det er mye kode å sette seg inn i og mange teknologier å lære. Dette vil blant annet omhandle Vue.js og Laravel (PHP) som er de mest sentrale teknologiene systemet er bygget med.

Motivasjonen for prosjektet er hovedsakelig å hjelpe mennesker med matpreferanser, samt å gjøre arrangementopplevelsen deres enklere og tryggere. Dette er hva CanCannotEat ønsker, og prosjektet vil forhåpentligvis hjelpe de å nå målet.

1.2 Kontekst

I Norge i dag er det ifølge CanCannotEat 22% av befolkningen som trenger tilpasset mat grunnet irritabel tarm, cøliaki, diabetes eller matvareallergi. Likevel finnes det få/ingen digitale løsninger for serveringsbransjen for å samle og sortere informasjon om matpreferanser. CanCannotEat vil opprette en kommunikasjonskanal mellom kjøkkenet og personer med matpreferanser. Med denne vil de strømlinjeforme kommunikasjonsprosessen ved å samle inn nødvendig informasjon om gjestenes matbehov.

1.3 Begrensninger

De største begrensningene for prosjektet vil være tid og kompetanse. To av deltakerne i prosjektgruppen har delvis kompetanse innen app-utvikling, og én har litt erfaring innen webutvikling. For å jobbe med dette prosjektet vil det være behov for prosjektgruppen å lære nye teknologier på kort tid. Siden selskapet betaler mye for tid med utviklere, vil det ikke være lett tilgjengelig for gruppen å få hjelp fra noen som kjenner systemet bra. Det vil da kreve mer av prosjektgruppen å sette seg inn i systemet på egen hånd.

CanCannotEat er en bedrift med mange store og små mål til funksjonalitet i applikasjonen. Prosjektgruppen skulle gjerne utviklet alle funksjonene oppdragsgiver trenger, men grunnet begrenset tid og dokumentasjonskravene som kommer med en bacheloroppgave vil vi fokusere på én større funksjon. Dersom vi har tid, vil vi også integrere mindre funksjoner som vil være nyttige for CanCannotEat.

1.4 Ressurser

De viktigste ressursene som trengs i prosjektet kommer fra Guilty og CanCannotEat. Guilty AS er en designbedrift bestående av fem utviklere, som holder til i Bergen. De har utviklet webapplikasjonen (forklart i 2.1.2) som blir brukt i dag. De har gjort all

kildekode tilgjengelig for prosjektgruppen, samt hjulpet med å sette opp systemet, laste ned nødvendige verktøy, og gitt oss litt innblikk i systemet. Uten denne hjelpen ville det tatt mye lenger tid å komme i gang.

CanCannotEat vil være en viktig ressurs fordi de har innsikt i markedet. De forstår hva som skaper verdi for deres kunder og brukere. CanCannotEat gir oss også tilgang på testere. Dette vil hjelpe oss med å få nyttige tilbakemeldinger. I tillegg har CanCannotEat bidratt med kontorplass, som bidrar til at vi enklere kan kommunisere sammen. Vi vil ha en kontinuerlig dialog med dem, slik at vi holder oss på riktig spor.

For å kjøre adminprogrammet er det behov for XAMPP på Windows eller Laravel Valet på macOS. Begge disse alternativene vil også installere MySQL. For å få bedre oversikt over databasen bruker vi HeidiSQL på Windows og TablePlus på macOS, som er GUI-verktøy for databasehåndtering. Koden vil bli skrevet i Visual Studio Code, hovedsakelig fordi det er lett å bruke, og fordi tredjeparts programvare tilbyr mange nyttige utvidelser. For å dele og samle dokumenter bruker vi Google Disk. Microsoft Word brukes for å skrive rapporten i, hvor flere kan redigere dokumentet samtidig. For kodedeling vil Git brukes. Disse programmene er forklart mer i detalj i kapittel 3.3.

1.5 Oppsett av rapporten

Kapittel 1 - Introduksjon til oppgaven, hvor det gis litt kontekst til hva prosjektet går ut på.
Kapittel 2 - Spesifiser hva oppgaven er.
Kapittel 3 - Diskuterer hvordan prosjektet skal gjennomføres, og hvilke løsninger som skal brukes.
Kapittel 4 - Beskriver designet og arkitekturen for produktet. Fremgangsmåten og løsningen av produktet blir her nærmere forklart.
Kapittel 5 - Beskriver hvordan arbeidet ble evaluert og resultatet av evalueringen.
Kapittel 6 – Diskusjon av prosjektets resultat og prosess

Kapittel 7 – Konklusjon av rapporten
Kapittel 8 – Referanseliste
Kapittel 9 – Vedlegg

2 Prosjektbeskrivelse

2.1 Praktisk bakgrunn

2.1.1 Oppdragsgiver: CanCannotEat

CanCannotEat er en gründerbedrift som ble grunnlagt i 2017. Bedriften jobber med å utvikle programvare som skal effektivisere og kvalitetssikre kommunikasjon mellom de som serverer, og de med behov for tilrettelagt mat.

Cecilie Ågotnes er gründeren som står bak den nyoppstartede bedriften. Hun har tidligere skrevet 4 bøker knyttet til tilrettelegging av kosthold for allergikere. Bøkene baserer seg på lavFODMAP-dietten som en forskningsbasert diett. Cecilie ble selv tvunget til å endre kostholdet sitt grunnet irritabel tarmsyndrom, som er hovedmotivasjonen bak både bøkene og selskapet (Ågotnes, 2015). CanCannotEat har en plan om å rulle ut flere nyttige verktøy for å hjelpe de med behov for tilrettelagt mat etterhvert som bedriften blir større.

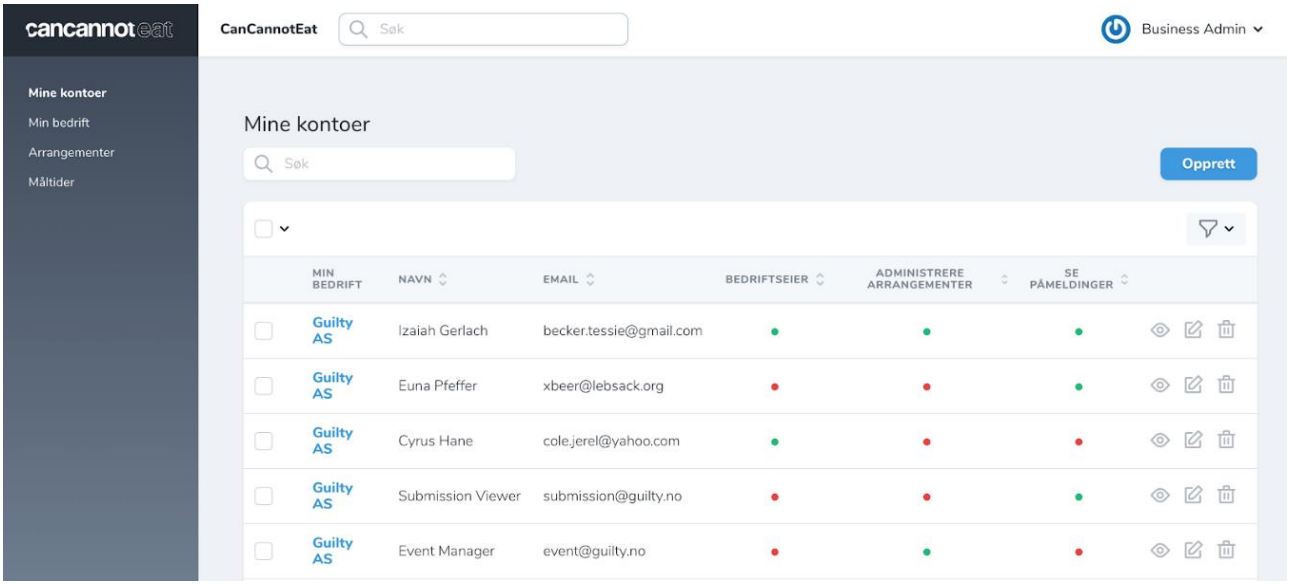
2.1.2 Tidligere arbeid

Webapplikasjonen er en kommunikasjonskanal mellom personer med behov for tilrettelagt mat og kokker. Målet med å utvikle kommunikasjonskanalen er å gjøre det lettere for kokkene og tryggere for brukeren. Kort fortalt legger arrangør ut et arrangement som brukeren kan melde seg på. Deretter vil brukeren kunne legge inn sine matpreferanser, og arrangøren vil så få en sortert liste over alle brukerne med tilhørende matpreferanser på arrangementet. Et eksempel av en slik liste er vist på

Figur 2. Webapplikasjonen består av to deler; et brukergrensesnitt for arrangøren, og et for brukeren/allergikeren.

Admin

Programmet er bygget med Laravel Nova, som er en del av Laravel-rammeverket, spesielt designet for adminstratasjonsdelen. Laravel Nova og Laravel forklares mer grundig senere i rapporten. I admin-delen er all koden skrevet i PHP og følger REST-struktur. Fra adminbrukeren kan du benytte CRUD-operasjonene (Create, read, update, delete) på objekter som kontoer, arrangementer og arrangementmåltider. I tillegg kan du definere hvilke roller forskjellige kontoer skal ha som vist i Figur 1. For eksempel kan det defineres at kjøkkenet kun kan se matpreferanselister og påmeldinger som vist i Figur 2.



Figur 1 - Kontovisning i adminpanel

Allergitabell

Last ned

Middag (24-05-2019)

ID Nummer	Peanøtter (Peanuts)	Egg (Egg)	Selleri (Celery)	Sennep (Mustard)	Sesamfrø (Sesam)	Soya (Soya)	Sitrus (Sitrus)	Gravid (Pregnant)	Sopp (Mushroom)	Løk (Onion)	Halal (Halal)	Hyper Comment	Comment
JT01	x	x	x	x				x			x		
KS03	x		x		x	x	x						
KS02	x	x							x	x			
	3	2	2	1	1	1	1	1	1	1	1		

Middag (26-05-2019)

ID Nummer	Peanøtter (Peanuts)	Egg (Egg)	Selleri (Celery)	Sennep (Mustard)	Sesamfrø (Sesam)	Soya (Soya)	Sitrus (Sitrus)	Gravid (Pregnant)	Sopp (Mushroom)	Løk (Onion)	Halal (Halal)	Hyper Comment	Comment
JT01	x	x	x	x				x			x		
KS03	x		x		x	x	x						
KS02	x	x							x	x			
	3	2	2	1	1	1	1	1	1	1	1		

Middag (25-05-2019)

ID Nummer	Egg (Egg)	Peanøtter (Peanuts)	Selleri (Celery)	Sennep (Mustard)	Gravid (Pregnant)	Sopp (Mushroom)	Løk (Onion)	Halal (Halal)	Sesamfrø (Sesam)	Soya (Soya)	Sitrus (Sitrus)	Hyper Comment	Comment
JT01	x	x	x	x	x			x					
KS02	x	x				x	x						
	2	2	1	1	1	1	1	1					

Figur 2 – Matpreferanseliste i adminpanel

På Figur 2 vises hvordan listene vil se ut for kokkene.

Brukersiden

Brukersiden er en progressiv webapplikasjon og Single Page Application, som er bygget med Vue 2. Disse begrepene blir nærmere forklart senere i rapporten. Dette er programmet brukeren vil benytte seg av. Her kan du opprette bruker, legge til matpreferanseliste og melde deg på et arrangement.

På Figur 3 ser man en skjermdump av en bruker som har meldt seg på et arrangement, og har delt sin matpreferanseliste.

Figur 3 - Brukersiden

2.1.3 Oppgavetekst

I utgangspunktet var oppgavebeskrivelsen at det skulle utvikles en mobilapplikasjon basert på webapplikasjonen oppdragsgiver allerede hadde. Etter hvert viste det seg at webapplikasjonen deres var bygd som en progressiv webapplikasjon. Dette vil si at

den er laget for mobil, oppfører seg som en mobilapp, og kan lastes opp på mobilappbutikker som Google Play Store og App Store.

Dette hadde blitt en meningsløs bacheloroppgave. Vi diskuterte derfor mye med oppdragsgiver for å finne en ny oppgave vi kunne gjennomføre. CanCannotEat har stort behov for videreutvikling av applikasjonen. Vi tenkte derfor dette ville være lærerikt og relevant for studiet vårt å implementere noen tilleggsfunksjoner. Nedenfor er de forskjellige tilleggsfunksjonene CanCannotEat trenger hjelp med å utvikle:

Tilleggsfunksjoner	
Meldingssystem	Funksjon hvor brukere kan sende meldinger til hverandre.
Tredjepartsinnlogging	Funksjon for å kunne trykke på knapp som gir mulighet for innlogging via Facebook-konto, og innlogging via Google-konto.
Pushvarsler	Opprette Varslingssystem for Android og iOS, samt web.
Utskriftsformat	Mulighet for å eksportere fil i diverse formater, som PDF og tilpasset utskrift.
Flere matpreferanselister per bruker	Foreløpig kan en bruker kun ha én matpreferanseliste tilknyttet sin konto. Oppdragsgiver ønsker at en bruker kan opprette flere lister, og velge hvilken av sine lister som blir brukt når en melder seg på et arrangement.
Popup-bokser/ infobokser	Hjepetekstbokser som kan komme opp der hvor ekstra beskrivelse er nødvendig.

Meldingssystem mellom bedrift og bruker	Mulighet for kommunikasjon mellom bruker og arrangementansvarlig.
Menymodul	Retter registreres av arrangør ved at det legges in allergener og inntoleranse retten inneholder. Etter arrangør har registrert retter, vil en bruker kunne velge mellom retter spesialtilpasset for deres matpreferanser
Endring av funksjoner som allerede finnes	Små endringer på programmet som må finpusses.

I diskusjon med veileder kom det klart frem at det ville bli vanskelig å skrive oppgave om mange forskjellige funksjoner. Vi valgte derfor å gå videre med en av forslagene altså meny-funksjonen. De resterende funksjonene ønsker CanCannotEat å utvikle, men det vil ikke bli fokusert på i denne bacheloroppgaven. Menymodulen som blir utviklet i denne oppgaven vil være et utgangspunkt for en løsning som skal videreutvikles av bedriften.

Krav til menymodul	Kommentar
Bedrift skal kunne opprette og publisere meny	Dette skal også bli lagret i CanCannotEat sin database.
Rettene i meny skal kobles opp mot matpreferanser (predefinert av CanCannotEat)	
Bruker skal kunne hente meny	

Brukerens matpreferanseliste påvirker hvilke retter som presenteres	Retter som passer bra for brukeren vil vise i menyen. Mye logikk på hva som skal vises her trenger å bli løst.

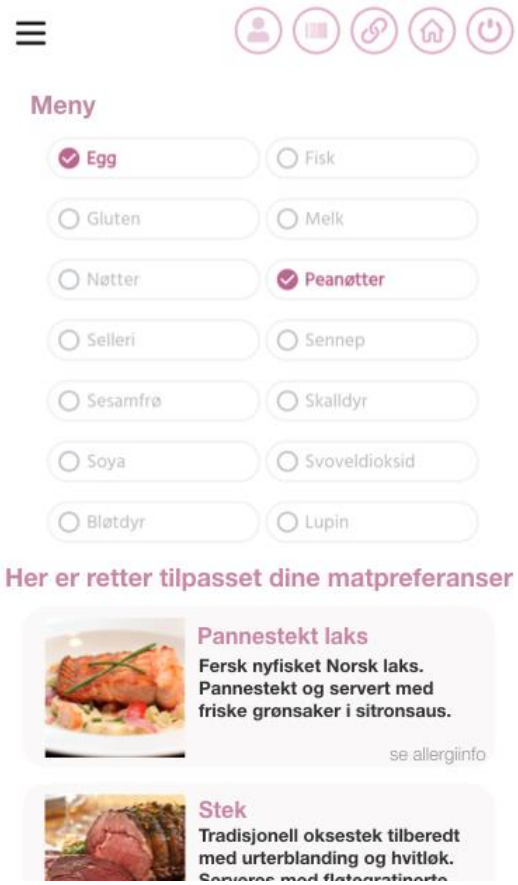
2.1.4 Løsningsforslag

Det aller viktigste i applikasjonen er at en meny kan kobles opp mot personlige matpreferanser, slik at brukeren kan se hvilke retter hen kan spise basert på disse matpreferansene.

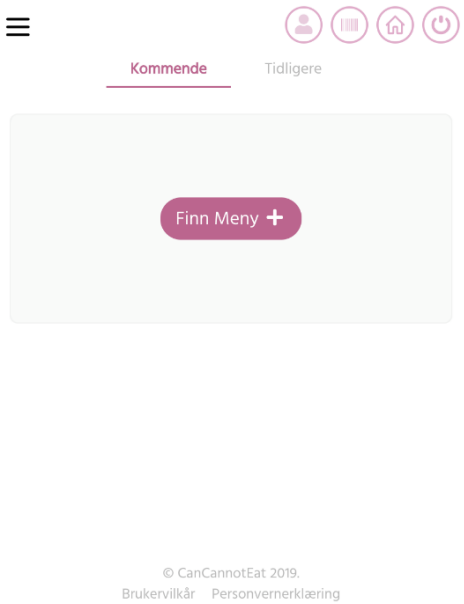
CanCannotEat ønsker å utvide programvaren sin i løpet av de neste årene slik at de dekker flere behov til personer med matpreferanser. Eksempler på dette er sosiale medier og meldingssystem mellom brukere. Av den grunn ønsker vi å beholde én brukerprofil, med tilhørende matpreferanser, som også vil fungere i menyapplikasjonen. I løsningen vil vi derfor benytte oss av brukerne, deres matpreferanselister og de definerte matpreferansene som allerede ligger i databasen. Det vil naturligvis være to ulike brukergrensesnitt for bedrift og bruker. Bedriften vil kunne opprette menyer bestående av matretter og muligens drikker. En matrett vil inneholde et bilde, og en matpreferanseliste, slik at hver matrett kan sammenlignes med brukerens matpreferanselister.

På brukersiden vil de oppgi en kode for å få tilgang til menyen, hvor de kan opprette ny bruker eller logge seg inn for å få tilgang til sin matpreferanseliste. Deretter vil de få opp en meny som er tilpasset deres preferanser. Det vil da også være mulighet til å se resten av menyen.

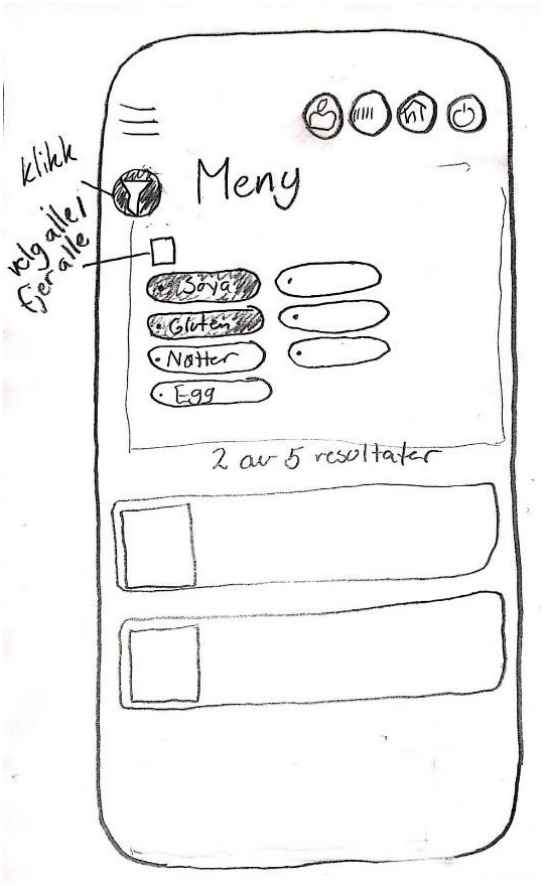
Førsteutkast Visualisering av brukersiden er vist på Figur 4, 5, 6 og 7.



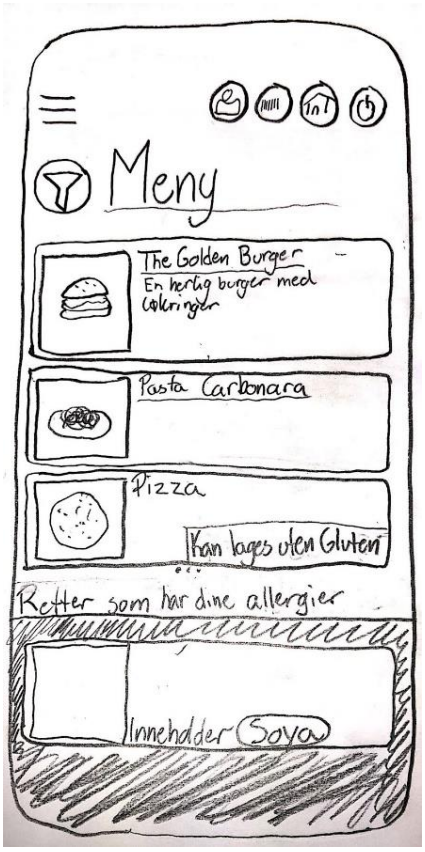
Figur 4



Figur 5



Figur 6



Figur 7

På Figur 4, 5, 6 og 7 ser vi førsteutkast av brukergrensesnittet til brukersiden av applikasjonen. Figur 4 og 5 er en grafisk prototype og Figur 6 og 7 er utkast tegnet for hånd.

3 Prosjektdesign

3.1 Mulige fremgangsmåter

3.1.1 Alternativ fremgangsmåte 1

En mulig løsning er å lage en såkalt “native applikasjon” eller en selvstendig app. Dette innebærer altså at vi velger egne rammeverk og utvikler en mobilapplikasjon som kan fungere uavhengig av det eksisterende brukergrensesnittet. Det blir altså utviklet en applikasjon som benytter det eksisterende API-et, slik at vi kan benytte

oss av brukerne, matpreferanselistene. Gitt at det er manglende kompetanse innenfor teknologiene i den eksisterende applikasjonen, kan det derfor være tidsbesparende å gjennomføre en “native” løsning.

3.1.2 Alternativ fremgangsmåte 2

En annen løsning for prosjektet vil være å utvikle en menymodul for webapplikasjonen. Ved å benytte de teknologiene og verktøyene som den eksisterende webapplikasjonen er bygget på, kan vi integrere vår modul med dette systemet. Det vil innebære å skrive brukersiden med Vue.js og adminsiden med Laravel Nova.

3.1.3 Diskusjon av mulige fremgangsmåter

Vi ønsker å skape mest mulig verdi for oppdragsgiver. Det tror vi vil oppnås ved å integrere menyapplikasjonen med det allerede eksisterende programmet. CanCannotEat har ingen på teamet som selv kan å utvikle. Dermed vil alle timene de bruker med utviklere være en direkte kostnad. Det er en av hovedargumentene for å gå for den integrerte løsningen, med tanke på at dette er en gründerbedrift og økonomi er en viktig faktor. Ved å velge den integrerte løsningen vil vedlikehold og videreutvikling være lettere for utviklerne, med tanke på at vi vil benytte oss av samme språk og teknologier som de utviklet webappen med. Vi ønsker også kunnskap innen Laravel og Vue.js, som ingen av oss har noe mye erfaring i. Ved å integrere dette i et program bygget på Laravel og Vue.js vil vi oppnå nettopp dette.

3.3 Verktøy og programmeringsspråk

Ved å implementere menymodulen i den eksisterende applikasjonen vil disse teknologiene bli benyttet:

Programmer:

Visual Studio Code	IDE for utvikling.
Laravel Valet (IOS)	Utviklingsmiljø som lar oss kjøre programmet lokalt på datamaskin, sammen med en lokal database.
PHP artisan	Kommandolinjegrensesnitt for å interagere med database.
XAMPP (Windows)	Package installer som installerer apache, MySQL, PHP og Perl på Windows.
TablePlus (IOS)	GUI-verktøy for database.
HeidiSQL (Windows)	GUI-verktøy for database.
Sourcetree	GUI-verktøy for Git.

Rammeverk:

Vue 2	Progressivt JavaScript-rammeverk bygget for å lage brukergrensesnitt.
Laravel (5.7)	Web-PHP-rammeverk som baserer seg på MVC-arkitekturen.

Språk:

PHP	PHP: Hypertext Preprocessor er et skriptspråk ofte brukt for å utvikle dynamiske og interaktive nettsider.

JavaScript (ES6) Høynivå programmeringsspråk. Sammen med HTML og CSS bygger de grunnmuren for webutvikling.

Dependencies:

```
25      "devDependencies": {
26          "axios": "^0.18",
27          "bootstrap": "^4.
28          "cross-env": "^5.
29          "jquery": "^3.2",
30          "laravel-mix": "^
31          "lodash": "^4.17.
32          "popper.js": "^1.
33          "vue": "^2.5.7"
34      }
```

Figur 8 – I adminprogram

```
11      "dependencies": {
12          "axios": "^0.18.0",
13          "laravel-vue-pagination": "^2.2.
14          "moment": "^2.22.2",
15          "pretty-checkbox-vue": "^1.1.9",
16          "vee-validate": "^2.1.0-beta.11"
17          "vue": "^2.5.17",
18          "vue-element-loading": "^1.0.4",
19          "vue-i18n": "^8.1.0",
20          "vue-js-modal": "^1.3.28",
21          "vue-notification": "^1.3.13",
22          "vue-pattern-input": "^2.0.3",
23          "vue-picture-input": "^2.1.6",
24          "vue-router": "^3.0.1",
25          "vue-tabs-component": "^1.5.0",
26          "vue-tippy": "^2.0.20",
27          "vuex": "^3.0.1",
28          "vuex-persist": "^1.5.3"
29      },
30      "devDependencies": {
31          "@fortawesome/fontawesome-pro":
32          "@vue/cli-plugin-babel": "^3.0.1
33          "@vue/cli-service": "^3.0.1",
34          "node-sass": "^4.9.0",
35          "sass-loader": "^7.0.1",
36          "tailwindcss": "^0.6.5",
37          "vue-template-compiler": "^2.5.1
38      }
```

Figur 9 – I brukerprogram

3.4 Utviklingsmetode

3.4.1 Scrum

Scrum er et agilt rammeverk som brukes til organisering av prosjekter og utvikling. Det brukes vanligvis for team bestående av 3 til 10 personer og bygger på at oppgaver skal kvantifiseres og utføres i et begrenset tidsrom kalt “sprint”. En sprint foregår normalt sett over 2 uker. Før hver sprint skal teamet møtes med oppdragsgiver og se tilbake på forrige sprint, finne ut hva som kan bli gjort annerledes, og definere hvilke arbeidsoppgaver som skal gjennomføres i neste sprint. Hver dag utføres et lite møte hvor personene i gruppen forteller om fremgangen opp til det punktet.

Scrum vil bli brukt i prosjektet i den grad det vil være praktisk å bruke det. Det vil bli daglige korte møter innad i teamet med refleksjon og oppdatering på fremgang. Det blir møte med oppdragsgiver mellom hver sprint for å få tilbakemelding slik at det eventuell kan utføres korrigeringer. Dette kommer i tillegg med andre møter vi har. I hver sprint bygges en modul, som sørger for at oppdragsgiver vil få verdi av arbeidet vi gjør, ikke bare av sluttproduktet.

3.4.2 Prosjektplan

Vi fant ut i uke 13 at vi ikke lenger kunne ha bacheloroppgaven vår, og fikk definert og godkjent en ny oppgave i uke 14. Dette gjorde at vi ble på etterskudd med innleveringene

Sprinter	Uke 13	14	15	16	17	18	19	20	21	22	23	24
Sprint 1				PÅSKE								
Forprosjektsrapport/presentasjon												
Sprint 2												
Statusrapport												
Sprint 3												
Sprint 4												
Plakat												
Bloggskriving												
Bachelorskriving												
EXPO												

Figur 10 – Gantt-diagram

Lys lilla: Oppgaver gitt av bedriften

Mørk lilla: Oppgaver gitt av HVL

Grå: Ferie

Sprint 1:

I sprint 1 vil det brukes mye tid på å sette seg inn i språkene og verktøyene som blir brukt i det eksisterende programmet, for å klare å implementere menymodulen. Kravene til oppgaven vil defineres, og designet vil skisseres. Det innebærer møter med oppdragsgiver for å forsikre oss at vi er enige om hvordan programmet skal se ut. I denne sprinten vil også forprosjektrapporten skrives.

Sprint 2:

I denne sprinten skal backend planlegges og implementeres. Det innebærer å sette opp databasen og utvikle administratorside av applikasjonen. På slutten av denne perioden vil vi begynne å tilrettelegge for spørringer mellom klient og server. I denne sprinten vil også statusrapport skrives.

Sprint 3:
I denne sprinten skal brukersiden av applikasjonen implementeres. Her vil det utforskes hvordan data skal hentes, filtreres og vises på best mulig måte, med designtegningene fra sprint 1 som utgangspunkt.

Sprint 4:
I den siste sprinten skal programmet brukertestes slik at justeringer kan bli gjort. Her ønsker vi også å bruke tiden på å implementere andre funksjoner som vil hjelpe helheten av applikasjonen.

3.4.3 Risikohåndtering

Planen er å unngå risikoene så tidlig som mulig. For å unngå en av de største risikoene; feiltolkning og/eller feilkommunikasjon, har vi kontor sammen med CanCannotEat. Vi kan da kommunisere med dem hver dag, samt holde jevnlige møter.

I tabellen under blir Sannsynligheten for at faktoren oppstår forkortet S, og Konsekvens for at den oppstår forkortet K. Sannsynligheten og Konsekvensen vurderes begge på en skala fra 1 til 10. Risikofaktor er produktet av Sannsynlighet og konsekvens og forkortes RK. Fullstendig oversikt over risikofaktorer ligger ved som Figur 18.

Suksessfaktor	S	K	RF	Tiltak
Manglende kompetanse innen fagområdet.	9	7	63	Dersom vi vil gå for å prøve å integrere løsningen vår i den allerede eksisterende webapplikasjonen, vil dette innebære at vi må bruke mye tid på å sette oss inn i de teknologiene som har vært benyttet. Dersom vi vil minske sannsynligheten for denne faktoren kan vi satse på å ikke integrere vår modul med det eksisterende programmet, men heller lage en slags prototype.
Dårlig kommunikasjon med - eller feil tolkning av krav fra oppdragsgiver.	4	8	32	Kontor er gjort tilgjengelig, som er svært nær oppdragsgiver. Gjennom å holde hyppige møter, sikres det at kommunikasjonen er god og at det er mindre rom for feiltolkning.

Figur 11 – Risikoanalyse

Basert på vår liste over risikoer er det vår faglige kompetanse som kan være den største risikoen med å ikke få prosjektet gjennomført. Dersom vi bestemmer oss for at vi vil integrere vår modul med den eksisterende webapplikasjonen, vil dette innebære at vi må bruke svært mye av tiden vår på å sette oss inn i de teknologiene og verktøyene som blir benyttet i applikasjonen. Det er selvsagt uvisst nøyaktig hvor lang tid dette vil ta, men om det skulle vise seg at dette ikke er oppnåelig innenfor tidsrammen vil vi heller fokusere på å begrense hvilke funksjoner som skal implementeres.

3.5 Evalueringsmetode

3.5.1 Brukertesting

Det er viktig at applikasjonen er enkel og intuitiv å bruke. Det aller viktigste er at kundene drar nytte av å bruke den, og at brukeren får følelsen av at ting blir lettere ved å bruke appen. CanCannotEat har tett samarbeid med Haukeland Universitetssykehus, og har gjennom dem tilgang på testgruppe. I den siste sprinten av prosjektet vil vi bruke denne testgruppen og få tilbakemelding på det vi har laget. Det vil da være mulighet for justering før vi leverer det endelige produktet. Hvis det viser seg at vi ikke får tid til å endre på ting, vil fortsatt testgruppen gi verdifull informasjon som CanCannotEat kan bruke for å utvikle applikasjonen sin videre.

Vi vil jobbe i samme lokale som oppdragsgiver, og ha statusmøter cirka en gang i uken. Som sluttevaluering vil vi også intervjue CanCannotEat angående om vi har nådd målene våre.

3.5.2 Kodetesting

CanCannotEat har planer om å fortsette å utvide applikasjonen. Det vil derfor være til fordel for dem at kodekvaliteten er bra, da andre skal bygge videre på koden senere. Laravel-rammeverket tilbyr et integrert testmiljø, som vil bli brukt i adminprogrammet. Ved å teste koden vil vi forsikre oss at alle metodene faktisk gjør det de skal gjøre, og vi forhindrer fremtidige bugs.

4 Produktdesign

4.1 Startfasen – uklar oppgave

Oppgaven vi først fikk utdelt i begynnelsen av året omhandlet å lage en kryssplattform mobilapp som tok utgangspunkt i en eksisterende nettside. I den første fasen av prosjektet ble ukene brukt til obligatoriske innleveringer knyttet til bacheloroppgaven. Noe av tiden gikk også i møter med oppdragsgiver og veileder,

kravspesifisering og planlegging av hvordan arbeidet skulle bli utført. Denne perioden foregikk samtidig som vi fortsatt hadde andre fag på skolen, og etter dette begynte bachelorperioden for fullt.

Den første uken fikk vi tildelt kontor på Nyskapningsparken på VilVite sammen med oppdragsgiver. Vi hadde ingen erfaring med å utvikle en kryssplattform mobilapp, så vi brukte den kommende uken på å lære oss de ulike språkene og verktøyene vi skulle bruke. Siden planen var å utvikle en applikasjon som skulle integreres med den eksiterende webappen, krevde dette å sette opp systemet lokalt på våre private PCer. Vi forsto at dette allerede var en progressiv webapplikasjon, og som forklart i 2.1.3 trengte vi å definere en ny bacheloroppgave. Tidligere innleveringer, problemstilling og mål måtte deretter defineres på nytt, to uker etter prosjektstart.

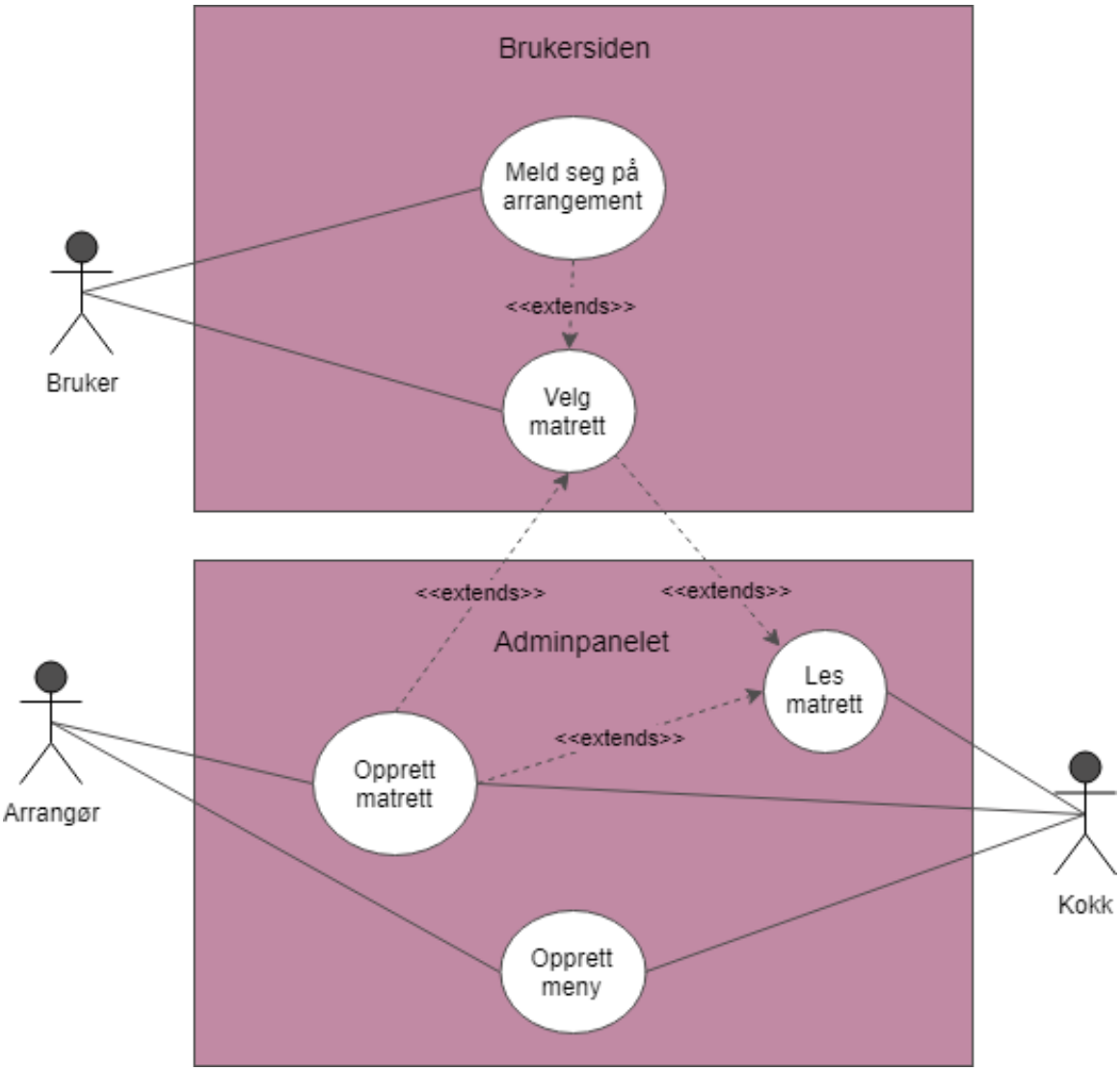
4.1.1 Definere ny oppgave

For å definere ny oppgave var det mye intern idémyldring og flere møter med CanCannotEat og veileder. Vi kommuniserte mye med CanCannotEat for å finne ut hvilke behov de hadde som bedrift, og hvilke fremtidsplaner de hadde for selskapet. Vi fikk flere forslag til diverse funksjoner og endringer, og tok dette videre til veileder, for å diskutere hvordan vi kunne definere en bacheloroppgave ut fra den nye informasjonen.

Vi kom etter mye tid fram til at vi skulle utvikle en menyfunksjon. Denne funksjonen hadde CanCannotEat tenkt å utvikle etter de får mer kapital. Vi kom fram til at det ville være verdifullt for dem å lage dette allerede nå. Vi ville da jobbe mye med design og planleggingsdelen for dem. Hvis vi utforsker ideen først, trenger de ikke å bruke like mye tid på dette senere. Dette er mye av motivasjonen til at vi har det som prosjekt.

4.2 Brukstilfeller

På Figur 12 ser vi brukstilfellediagram til adminpanelet og brukersiden. Strek mellom handling og aktør representerer at aktøren kan gjøre handlingen. Prikkete strek betyr at handlingen som blir pekt på er avhengig av handlingen som blir pekt fra.

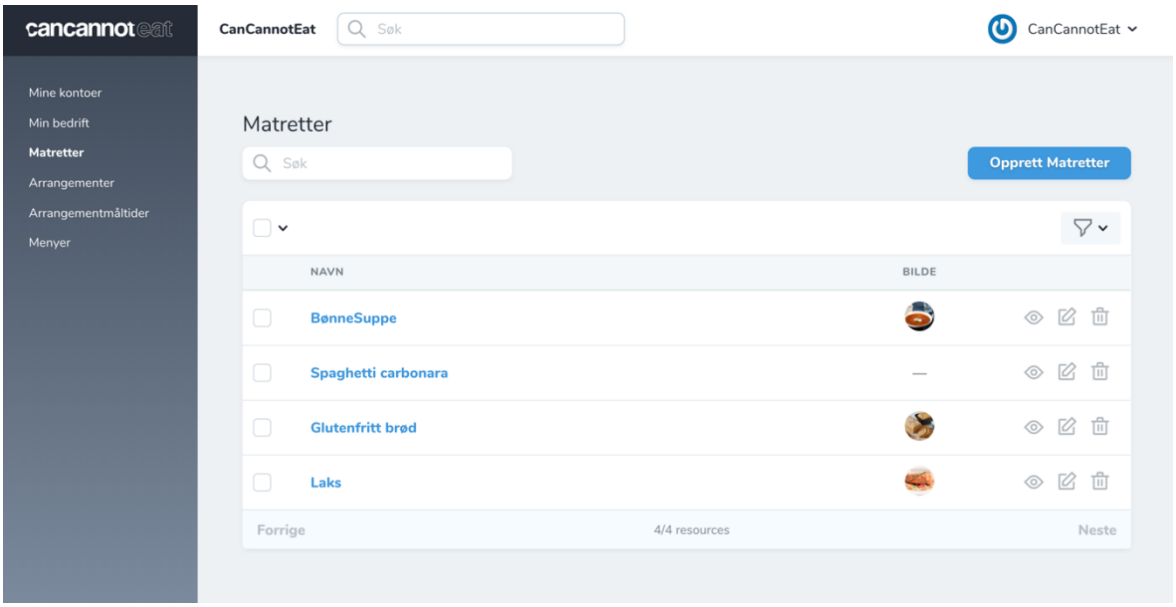


Figur 12 – Brukstilfellediagram

Som vist på Figur 12 kan en bruker melde seg på et arrangement, og vil så kunne velge matrett basert på sin personlige matpreferanseliste. Arrangøren og kokken kan

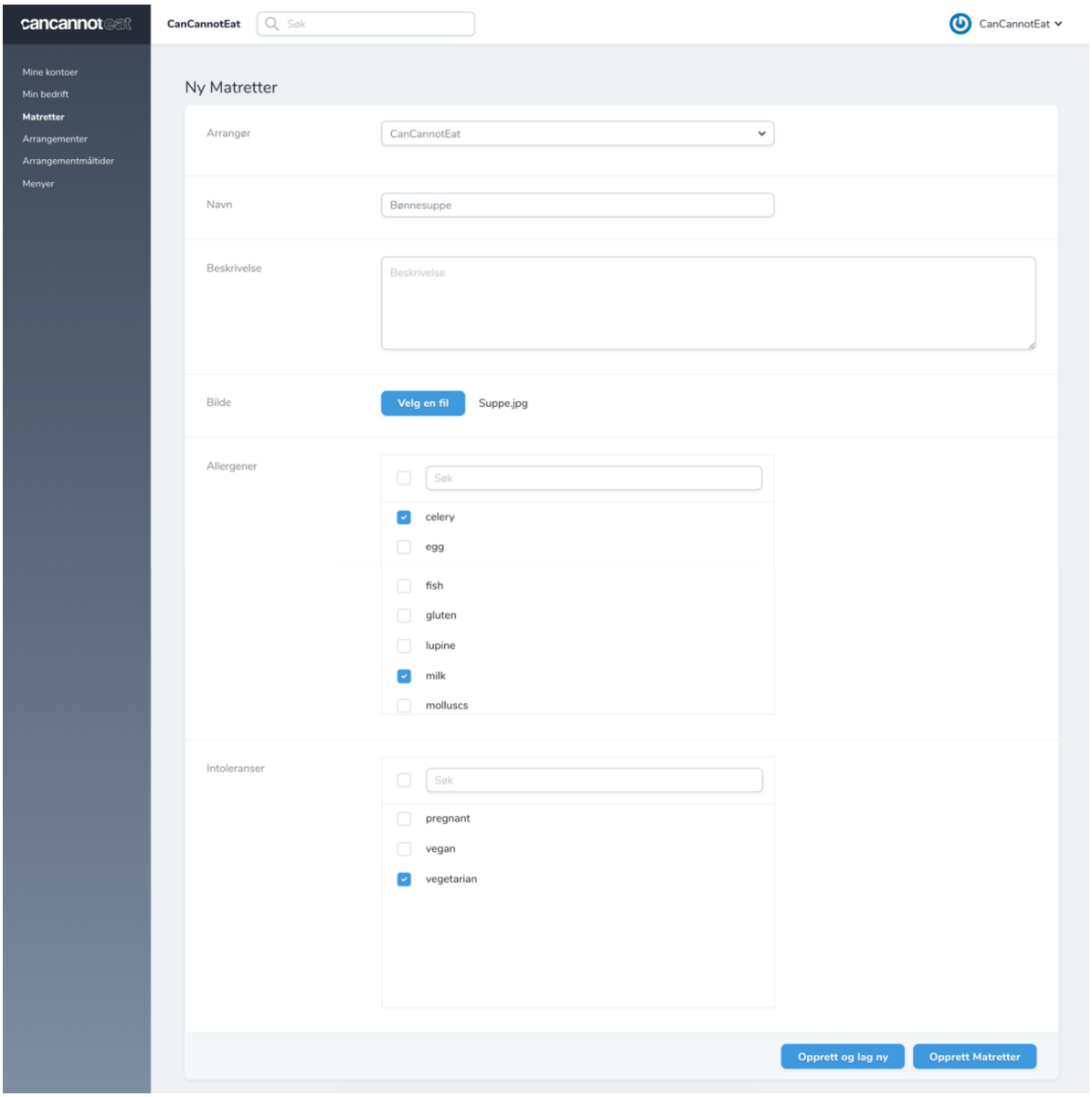
opprette menyer bestående av matretter. Kokken vil kunne se hvilke matretter som er valgt av brukere. Under blir de forskjellige brukstilfellene nærmere beskrevet.

Et av brukstilfellene er at arrangør oppretter en matrett i adminpanelet. Dette gjøres ved å klikke på «Opprett matretter», vist på Figur 13.



Figur 13 – Oversiktssiden for matretter. https://api.cancannoteat.com/courses

Man blir så tatt til siden for å opprette matrett som vist på Figur 14.



Figur 14 – Siden for oppretting av matrett.

https://api.cancannoteat.com/courses/new

Her kan brukeren legge inn navn, beskrivelse og bilde av matretten. Hver matrett kan ha tilhørende matpreferanser som kan spesifiseres ved å krysse av boksene under allergener og intoleranser. Ved å trykke “Opprett matretter” blir matretten lagret og du blir sendt til oversiktsiden for matretten som nettopp ble opprettet. Ved å trykke

“Opprett og lag ny” blir matretten lagret og du vil få et tomt skjema slik at en ny matrett kan opprettes.
Matretten vil tilhøre bedriften brukeren er logget inn med.

4.3 Grafisk prototyping

CanCannotEat hadde lite tanker om hvordan de ønsket at brukergrensesnittet på brukersiden skulle se ut. Det ble gitt veldig mye frihet når det kom til design. Dette innebar også at det var prosjektgruppens oppgave å utforme designet på brukergrensesnittet. Rollen innebar også å sette begrensninger, dersom dette er en gründerbedrift med store planer og mange idéer.

Etter vi hadde laget tegninger til hvordan vi tenkte oss at designet kunne se ut, hadde vi møte med CanCannotEat og presenterte idéene våre.

4.4 Database

Da databasen skulle opprettes var det naturlig å bruke samme teknologier som allerede ble brukt i den eksisterende databasen. Dette innebar å bruke en MySQL-database, opprette tabeller med migrasjoner (Migrations), og bruke Eloquent ORM (Object-relational Mapping) som kommunikasjon. Disse teknologiene vil bli forklart i punktene under.

4.4.1 Migrasjon

For oss var det viktig å bruke et verktøy som gjorde det lett å opprette, forandre og slette tabeller i databasen mens vi utviklet. Det var også viktig at forandringene kunne deles slik at alle i gruppen hadde like lokale databaser. Gjennom Laravels komandolinjegrensesnitt artisan, kan man benytte seg av Laravels databasemodul «Migration».

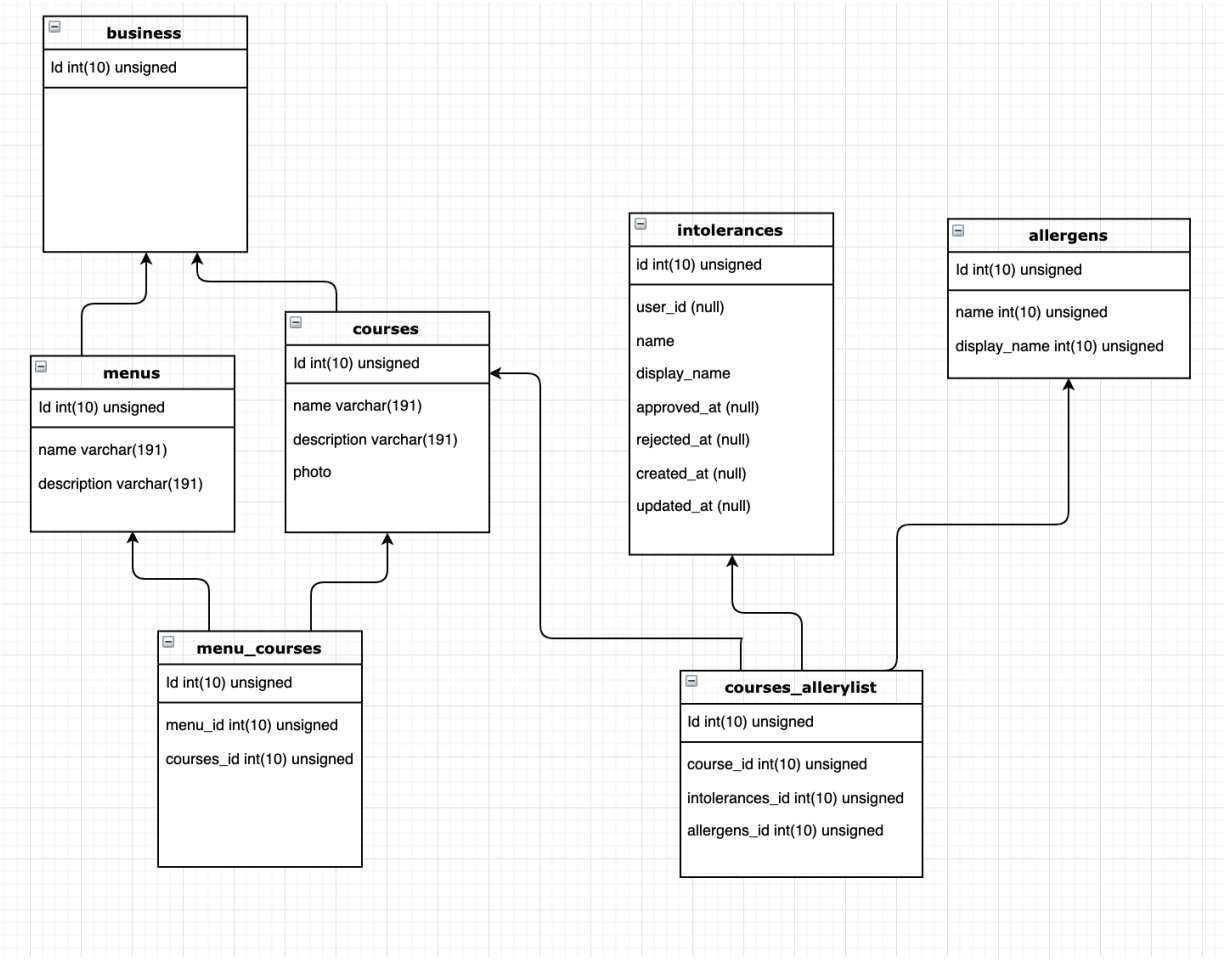
En migrasjon kan opprette tabeller i databasen og knytte de opp mot modellene i programmet. Migrasjonsfilen ble lastet opp på Git hver gang det ble utført en endring. På denne måten sikrer en at alle utviklerne har samme migrasjonsfil, som igjen fører til at alle jobber med identisk database.

```php
class CreateCoursesTable extends Migration
{
    public function up()
    {
        Schema::create('courses', function (Blueprint $table) {
            $table->increments('id');
            $table->String('name');
            $table->String('description')->nullable();
            $table->string('photo')->nullable();
            $table->unsignedInteger("business_id");
            $table->foreign("business_id")->references("id")->on("businesses")
                ->onUpdate("cascade")
                ->onDelete("cascade");

            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('courses');
    }
}
```

Figur 15 – Migrasjonsfil til matrett

Laravels Seeder ble brukt for å generere testdata, slik at programmet allerede hadde noen brukere, arrangementer, matretter osv. Dette gjorde det lett å justere databasen uten å miste alt innhold.

4.4.2 Databasearkitektur



Figur 16 – ER-diagram

Menus og Courses tabellene må ha navn og tilhøre en bedrift. For å koble allergener og intoleranser til en matrett ble det laget en relasjonstabell for å håndtere mange-til-mange-forhold, kalt courses_allergylist. Denne må inneholde en matrett-ID, og kan inneholde allergen- og intoleranse-ID. Det samme har blitt gjort for menyer og matretter med menu_courses-tabellen.

4.4.3 Kommunikasjon med database

For å kommunisere med databasen ble “Eloquent” brukt, som er Laravavels implementasjon av ORM. Object-Relational mapping (ORM) er en teknikk som lar en utføre spørringer til en database når en benytter et objekt-orientert programmeringsspråk. Hver tabell i databasen har en korresponderende “Model” som blir brukt til å kommunisere med denne tabellen. Et ORM-bibliotek gjør det enklere å utføre spørringer til databasen ved at den innkapsler koden en trenger for å manipulere data i databasen. På denne måten unngår en å skrive SQL-spørringer, og kan forholde seg til samme programmeringsspråk som resten av koden.

4.5 Admin

4.5.1 Arkitektur

Admindelen av programvaren er som tidligere nevnt bygget på Laravel Nova sitt rammeverk. Programvaren baserer seg på det arkitekturiske mønsteret Model-View-Controller (MVC). MVC blir brukt for å separere ulike problemer innenfor en applikasjon i 3 logiske komponenter. Modellen er komponenten som håndterer data og hvordan det skal behandles. «View» blir brukt for presentasjon av data for brukeren og kontrolleren er grensesnittet mellom Model og View [14]. Kontrolleren skal ta hånd om innkommende forespørsler, manipulere dataen gjennom modellen. For at menymodulen enkelt skal integreres med den eksisterende applikasjonen, skal denne også følge MVC-mønsteret.

Som nevnt i beskrivelsen av brukstilfellene (4.1.3) skal en kunne opprette, redigere, og slette menyer. De samme operasjonene skal gjelde for matretter. Videre skal én eller flere matretter knyttes til en meny i databasen, og én eller flere menyer skal kunne knyttes til et arrangement. Det er altså denne funksjonaliteten som må implementeres i adminpanelet, ved hjelp av Laravel Nova.

4.5.2 SPA

Applikasjonen er bygget som en “Single Page Application” (SPA). En SPA blir kjørt i en nettleser, men er unik i den forstand at den ikke krever å utføre flere spørringer til en server for å laste inn sider. Det som kreves av HTML, CSS og JavaScript for å bruke applikasjonen blir lastet inn fra én enkelt spørring til serveren. Etter dette blir alt annet innhold lastet inn asynkront ved hjelp av JavaScript. Dette fører til at applikasjonen er svært rask, og oppleves som svært responsiv for brukeren. Applikasjonen vi utvikler benytter Vue.js som rammeverk for SPA-en.

4.5.3 Resource

Den mest fundamentale egenskapen til Laravel Nova er muligheten til å administrere den underliggende databasen ved hjelp av Eloquent ORM som forklart i 4.4.1. Nova har definert det slik at hver tabell i databasen kan ha en samsvarende “Model” og “Resource” (i relasjonstabellene er dette ikke nødvendig). En Resource kan gjennom modellen gjøres tilgjengelig for adminpanelet. Alle Resource-filene inneholder en

Fields-metode. Denne metoden tillater oss å binde et inputfelt mot en kolonneverdi i databasen. Man kan sette ulike spesifikasjoner for hver field, som vist i Figur 17.

```
47      public function fields(Request $request)
48      {
49          return [
50              ID::make()->sortable()
51                  ->hideFromIndex()
52                  ->hideWhenUpdating()
53                  ->hideWhenCreating()
54                  ->hideFromDetail(),
55
56              BelongsTo::make(__('Customer'), 'business', Business::class)
57                  ->withMeta([
58                      'belongsToId' => $this->business_id ?? auth()->user()->business_id,
59                  ])
60                  ->hideFromIndex()
61              ,
62
63              Text::make(__("Name"), "name")
64                  ->rules("required")
65                  ->sortable()
66                  ->hideFromIndex(),
67
68              Text::make(__('Name'), function () {
69                  return "<a href='/admin/resources/courses/{$this->id}'
70                  class='no-underline dim text-primary font-bold'>{$this->name}</a>";
71              })
72                  ->asHtml()
73                  ->hideWhenUpdating()
74                  ->hideWhenCreating()
75                  ->hideFromDetail(),
76              Text::make(__("Description"), "description")
77                  ->sortable(),
78              Image::make(__("Photo"), "photo"),
79              AttachMany::make(__("Allergens"), "allergens", Allergen::class),
80
81              AttachMany::make(__("Intolerances"), "intolerances", Intolerance::class),
82
83          ];
84      }
```

Figur 17 – Fields-metode i CourseResource

4.5.4 Policy

Laravel legger opp til at brukere kan ha ulik tilgang til det man kan se og endre på inne i programmet, med å gi brukere ulike roller. En rolle kan for eksempel være en

bedriftsansvarlig, som har tilgang til å opprette nye brukere for sin bedrift. En annen rolle kan være å få lov til å opprette menyer og matretter, men ikke opprette nye brukere. Policy gir mulighet til å definere hvem som kan utføre handlinger knyttet opp mot en resource. En handling kan for eksempel være å opprette, endre, eller slette en matrett. De ulike rollene i programmet ble illustrert i Figur 1.

4.5.5 Problemer/vanskeligheter

Laravel Nova er designet for å gjøre det enkelt å utvikle et administrasjonspanel, ved at en kan definere de forskjellige attributtene som er blitt presentert i punktene ovenfor. Dette rammeverket ble gitt ut i august 2018 og er utviklet av Laravel-teamet (Redmond, 2018). Siden det er så nytt er det naturlig at det ikke er tilpasset akkurat slik vi kunne trengt å bruke det. Det er open source, og lagt opp til å kunne opprette spesialtilpassede attributter. Dermed ligger det ulike private løsninger offentlig på GitHub.

En av de funksjonene som ikke eksisterte, men som vi trengte, omhandlet mange-til-mange-relasjon. Mer spesifikt at en kan velge flere objekter samtidig, og knytte de opp mot et annet objekt, i det man oppretter et objekt. Altså å velge flere allergener og intoleranser når man oppretter en matrett, og samtidig lagre dette i databasen. Vi brukte mye tid på å prøve å manipulere de forskjellige fieldsene til ønsket funksjonalitet. Det fantes for eksempel en field som genererte avkrysningsbokser(checkboxes), som lagret en tabell av verdiene som ble krysset av. Da var tanken å iterere igjennom tabellen og lagre hver verdi i relasjonstabellen Course_allergylist. Dette fungerte ikke. Vi klarte å løse det samme problemet i Menu_course, relasjontabellen som definerer hvilke matretter som finnes i en meny. Problemet med den løsningen var at vi bare kunne knytte til en og en matrett, og måtte lagre mellom hver matrett som ble lagt til i menyen. Dette er en altfor tungvint måte å gjøre det på for arrangøren. Etter mye «prøving og feiling» samt undersøking fant vi ut at denne funksjonaliteten vi var ute etter ikke var tilstede i Laravel Novas ferdigdefinerte biblioteker. Vi begynte å utvikle vår egen custom field, da vi kom over

en annen custom field på Github (Dillingham, 2019). Dette løste mange problemer, men var heller ikke helt ideel for produktet.

5 Evaluering

5.1 Evalueringsmetode

For å evaluere brukervennlighet var det i utgangspunktet tenkt at testgruppen på Haukeland universitetssykehus skulle benyttes. Dette ble aldri gjennomført på grunn av at utviklingen av adminprogrammet var mye mer krevende enn forventet. Dette resulterte altså i at utviklingen på brukersiden ikke var kommet langt nok til at det var hensiktsmessig å teste.

Som en alternativ evaluering av applikasjonen formulerte vi oppgaver basert på noen brukstilfeller, som oppdragsgiver skulle utføre i applikasjonen. Den ene oppgaven som skulle utføres var å opprette en matrett med tilhørende navn, beskrivelse, bilde og matpreferanser. Etterpå ble de bedt om å opprette en ny meny, for så å legge til en eller flere matretter i menyen.

Underveis i utviklingen ble det gjennomført funksjonelle tester på applikasjonen av prosjektgruppen. Det ble testet kontinuerlig etter hvert som nye funksjoner ble lagt til og etter hver iterasjon.

5.2 Evalueringsresultat

5.2.1 Evaluering av oppdragsgiver

Oppdragsgiver fikk testet applikasjonen som beskrevet i evalueringsmetoden over, før de ble stilt en rekke spørsmål. Det ble stilt spørsmål både angående selve produktet, og hva de synets om arbeidet som var blitt gjort. Første spørsmål var om det fungerte som det skulle. De mente det manglet funksjonalitet som kan koble

matrettene til et arrangement. Videre uttrykte de at uten denne funksjonaliteten er produktet ikke brukbart. Bortsett fra det fungerte det som ønsket.

Vi spurte også om hvor intuitivt programmet var å bruke. Her mente de det var uklart hva «gravid», «vegansk» og «vegetariansk» på intoleranseseksjonen betydde, som vist på Figur 14. I det oppdaterte programmet vises flere intoleranser å kunne velge mellom, men siden vi har jobbet med en litt eldre versjon av adminprogrammet vises bare disse 3 alternativene.

Personen som oppretter matretten huker av på intoleransene. Dette skal være ting som løk, laktose, gravid, og halal. Her var det uklart hva det betyr å huke av. Å huke av på «gravid» kan både bety at det er egnet for gravide, og at det ikke er egnet. Dette er logikk CanCannotEat har hatt problemer med før, og mangler fortsatt en løsning på. Problemet kom tydelig fram i demoen vår.

Det var lett for dem å forstå oppbygningen ettersom de er vant med det eksisterende systemet og vår modul følger generelt sett samme oppbygning som denne. De mente derimot at systemet kunne være mindre intuitivt for noen som ikke har kjennskap til det fra før. De foreslo derfor å ha beskrivende tekstbokser flere steder i programmet. Hvis det heller hadde blitt brukt en testgruppe uten forståelse på systemet ville det nok gitt bedre tilbakemeldinger om brukervennligheten.

Programmet var veldig likt slik de hadde sett for seg, bortsett fra at det manglet funksjonaliteten med å koble matrett til arrangement. Tingene de ønsket å forbedre var å legge til denne funksjonaliteten og å gjøre intoleransedelen mer intuitiv for brukeren av applikasjonen. Selv om brukersiden ikke hadde blitt utviklet var CanCannotEat fornøyd med det som hadde blitt laget.

6 Diskusjon

Det ble tidlig valgt at menyapplikasjonen skulle integreres i det eksisterende programmet, som betydde at det var mye å sette seg inn i. Dette gjelder ikke bare det å sette seg inn i det eksisterende programmet, men også det å lære programmeringspråket PHP som prosjektgruppen ikke hadde noe erfaring med fra tidligere. Det ble svært undervurdert hvor mye tid dette kom til å ta. Gruppen var ivrige å begynne å produsere kode, og begynte med dette før de egentlig hadde lært mye av «grunnmuren» til rammeverket. Dette førte til unødvendige feil, som kunne ha blitt unngått hadde dokumentasjonen til rammeverket (Laravel Nova) blitt lest ordentlig.

De som utviklet det eksisterende programmet var konsulenter, som tok timebetaling hos CanCannotEat. Dette førte til at det var høy terskel for å spørre om hjelp, da gruppen stod fast. Det oppsto noen problemer, som også andre hadde støtt borti. Hvor gruppen lærte at mange flere hadde samme problem, men ingen hadde funnet(postet) en løsning på det. Dette er eksempler hvor mye tid ble brukt til undersøking. Noe som enkelt kunne bli løst hadde gruppen spurt om hjelp.

Hadde gruppen utført prosjektet på nytt er det mye som hadde blitt gjort annerledes. Det ville blitt lagt mer vekt på oppgavefordeling innad i gruppen, slik at hvert gruppemedlem hadde en konkret oppgave å jobbe med til enhver tid. Når problemer oppsto, kunne gruppen ha satt en tidsramme for hvor lenge de skulle prøve å løse problemet. Dersom en overskred tidsrammen, måtte en begynne å jobbe med en annen del av programmet. Innimellom kan et problem løse seg bare ved å vente noen dager, og ta for seg problemet med «nye øyne»

7 Konklusjon og videre arbeid

7.1 Oppsummering

Sammendrag av mål:

- å implementere menyapplikasjonen i den eksisterende løsningen.
- sette seg inn i tekonologiene som ble brukt i det eksisterende programmet, som blant annet Vue.js og Laravel (PHP) som systemet er bygget med.
- å hjelpe personer som trenger tilrettelagt mat, ved å lettere spesialtilpasse menyer som samsvarer med deres matpreferanser.

Et av målene var et personlig mål om å få kunnskap om de teknologiene som allerede var brukt i applikasjonen, samt å lære Laravel og Vue. Gruppen mener dette målet ble nådd. Kunnskapen om Laravel Nova er spessielt høy, da dette var hva mesteparten av tiden gikk til. Det ble samlet kunnskap om Vue.js, men dette ble ikke brukt like mye i praksis, da mesteparten av javascriptprogrammeringen lå i brukerside-delen av programmet.

Å implementere menyapplikasjonen i den eksisterende løsningen ble gjort i admindelen av programmet, dette var som nevnt tidligere mer omfattende enn forventet. Dette gikk trolig utover tiden som kunne blitt brukt til utvikling av brukersiden, likevel var verdien i å implementere deler av løsningen større enn å bygge en selvstendig plattform. Dette målet ble egentlig ikke nådd, da brukersiden ikke ble klar innen innleveringsfristen. Det som ble produsert følger likevel målet.

Det siste målet som omhandler å hjelpe personer som trenger tilrettelagt mat, har ikke gruppen direkte nådd. Dette er fordi produktet ikke er klart for publisering. Om CanCannotEat velger å bruke ressurser på å ferdigstille produktet, kan det hjelpe personer som trenger tilrettelagt mat. Om produktet faktisk ville ha hjulpet disse brukerne, står ubesvart som følge av at gruppen ikke fikk gjennomført brukertesting

av et ferdigstilt produkt. Nytteverdien av et ferdigstilt produkt ble dermed ikke evaluert av en reell brukergruppe.

Gjennom hele prosjektet har gruppen lært mye om hvordan det er å jobbe for en oppstartsbedrift. Det har vært lærerikt å ha kommunikasjonsrollen gruppen har hatt for CanCannotEat. Gruppen har trengt å kommunisere kravspesifikasjoner med noen som er veldig lite tekniske. Dette har lært dem mye om å finne et felles språk, og å se målene fra oppdragsgivers perspektiv. Det har også vært lærerikt å måtte sette seg inn i et allerede eksisterende system og lære nye teknologier for å videreutvikle dette systemet. Gruppen sitter også igjen med kunnskap om selve språkene, rammeverkene og verktøyene som ble brukt.

Det eksisterende systemet som prosjektgruppen har videreutviklet, er mye større enn andre system gruppen har utviklet gjennom utdanningsløpet. Dette gir god innsikt i arkitekturen på et større system, og viser hvor viktig struktur er. Gruppen lærte også mye om det å løse problemer på egenhånd, og har blitt flinkere til å feilsøke kode.

7.2 Videre arbeid

Videre arbeid som kan gjøres er naturligvis å ferdigstille produktet. Dette innebærer å koble matretter opp mot arrangementer, og implementere brukersiden av applikasjonen. I tillegg til å utvide oversikten hvor arrangørbedriften får en sortert liste over matpreferanser, som vist i Figur 2.

Programmet burde også bli testet hos en testgruppe, slik at CanCannotEat får tilbakemeldingen de har behov for. Dersom det hadde vært mer tid hadde testklasser blitt laget, for å forsikre at koden funker som den skal. API-endepunkter hadde også blitt laget, samt kontrollere.

Andre ting som kan implementeres har blitt diskutert i møte med CanCannotEat. I startfasen av prosjektet hadde oppdragsgiver mange funksjoner de ønsket at

programmet sitt skulle få. Disse funksjonene vil bli utviklet videre. I løpet av utviklingsperioden har oppdragsgiver kommet med enda flere idéer til hvilke funksjoner programmet trenger.

Én funksjon er å kunne laste opp flere bilder samtidig, slik at en kan lage mange matretter på én side, og kunne legge til bilder på en annen side. Muligheten om å legge til bilde ved å dra filen inn i nettleseren er også en ønsket tilleggsfunksjon. En annen idé var at å lage varslingsystem, slik at brukeren hadde fått pushvarsel om det ble opprettet en meny på påmeldt arrangement.

Videre fram i tid ønsker CanCannotEat at programmet skal kunne foreslå matretter til arrangøren basert på hvilke matpreferanser som er sendt inn. Arrangøren vil da få hjelp av programmet til å sette opp menyen. Mulighet for at hotellkjeder kan dele menyer og matretter mellom sine hoteller er også ønskelig på sikt.

8 Referanser

[1] Nova.laravel.com. (2019). *Laravel Nova Docs*. [online] Available at: https://nova.laravel.com/docs/ [Accessed 14 Mar. 2019].

[2] Cook, S. (2018). *Creating a CRM with Laravel Nova | VOLTAGE*. [online] VOLTAGE. Available at: https://voltagead.com/creating-a-crm-with-laravel-nova/ [Accessed 24 April 2019].

[3] Cook, S. (2018). *Creating a CMS with Laravel Nova | VOLTAGE*. [online] VOLTAGE. Available at: https://voltagead.com/creating-a-cms-with-laravel-nova/ [Accessed 24 April 2019].

[4] Ågotnes, C. (2014). *Om Cecilie - lavFODMAP (Low FODMAP)*. [online] lavFODMAP (Low FODMAP). Available at: https://www.lavfodmap.no/om-cecilie/ [Accessed 2 May 2019].

[5] Skólski, P. (2016). *Single-page application vs. multiple-page application*. [online] Medium. Available at: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58 [Accessed 11 Apr. 2019].

[6] Sherman, P. (2018). *blog.pshrmn*. [online] Blog.pshrmn.com. Available at: https://blog.pshrmn.com/how-single-page-applications-work/ [Accessed 12 Apr. 2019].

[7] Redmond, P. (2018). *Laravel Nova is Now Available*. [online] Laravel News. Available at: https://laravel-news.com/laravel-nova-release [Accessed 15 Apr. 2019].

[8] Dillingham, B. (2019). *dillingham/nova-attach-many*. [online] GitHub. Available at: https://github.com/dillingham/nova-attach-many [Accessed 31 Apr. 2019].

[9] Kandil, A. (2019). *ahmedkandel/nova-attach-many*. [online] GitHub. Available at: https://github.com/ahmedkandel/nova-attach-many [Accessed 2 May 2019].

[10] Cli.vuejs.org. (2018). *Overview | Vue CLI*. [online] Available at: https://cli.vuejs.org/guide/ [Accessed 5 Apr. 2019].

[11] Router.vuejs.org. (n.d.). *Introduction | Vue Router*. [online] Available at: https://router.vuejs.org/ [Accessed 4 Apr. 2019].

[12] Basile, N. (2018). *Deep Diving Laravel Nova*. [online] Nick-basile.com. Available at: https://nick-basile.com/blog/post/deep-diving-laravel-nova [Accessed 20 Mar. 2019].

[13] Basile, N. (2018). *Deep Diving Laravel Nova*. [online] Nick-basile.com. Available at: https://nick-basile.com/blog/post/getting-started-with-laravel-nova [Accessed 20 Mar. 2019].

[14] www.tutorialspoint.com. (n.d.). *MVC Framework Introduction*. [online] Available at: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm [Accessed 2 May 2019].

[15] Sandberg, T. (2019). *PANNESTEKT LAKS MED GRØNNSAKER I SITRONSAUS | TRINES MATBLOGG*. [online] TRINES MATBLOGG. Available at: https://trinesmatblogg.no/recipe/pannestekt-laks-med-gronnsaker-i-sitronsaus/ [Accessed 28 May 2019].

[16] Hvl.no. (2019). *EXPO 2019*. [online] Available at: https://www.hvl.no/kalender/expo-2019/ [Accessed 30 May 2019].

9 Vedlegg

9.1 Risikoliste

Suksessfaktor	S	K	RF	Tiltak
Manglende kompetanse innen fagområdet.	9	7	63	Dersom vi vil gå for å prøve å integrere løsningen vår i den allerede eksisterende webapplikasjonen, vil dette innebære at vi må bruke mye tid på å sette oss inn i de teknologiene som har vært benyttet. Dersom vi vil minske sannsynligheten for denne faktoren kan vi satse på å ikke integrere vår modul med det eksisterende programmet, men heller lage en slags prototype.
Dårlig kommunikasjon med - eller feil tolkning av krav fra oppdragsgiver.	4	8	32	Kontor er gjort tilgjengelig, som er svært nær oppdragsgiver. Gjennom å holde hyppige møter, sikres det at kommunikasjonen er god og at det er mindre rom for feiltolkning.
Feilberekning av tid/ Sette for store mål i forhold til tid.	8	3	24	Som tiltak vil vi prøve å begrense omfanget på oppgaven, og heller prøve å fokusere på de viktigste delene av applikasjonen.
Bedriften endrer behov.	4	6	24	Gitt at CanCannotEat er en oppstartsbedrift, er det ikke helt definert i hvilken retning selskapet vil gå. Enten kan kravene for applikasjonen endre seg, eller bedriften kan se en endring i behovet for vår modul.
Sykdom innad i gruppen.	2	9	18	Selv om sykdom ikke er en stor sannsynlighet, vil det ha en stor innvirkning på gruppen siden gruppen vår bare består av 3 medlemmer. Det kan åpnes opp for at medlemmer kan jobbe hjemmefra dersom dette skulle inntreffe.
Bedriften går konkurs/manglende investorer.	2	4	8	Dersom det er behov for å snakke med utviklere, vil dette være en kostnad for CanCannotEat. Av den grunn bør det kun være møter med utviklere ved behov, og møtene bør være så effektive som mulig for holde kostnadene til et minimum.

Figur 18 – Fullstendig risikoanalyse