CHALMERS

GÖTEBORGS UNIVERSITET

MeadLoggr

En Webbapplikation för Mjödbryggeri

Examensarbete inom data- och informationsteknik

Oscar Larsson
Sebastian Sela

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA

Göteborg, Sverige 2023
www.chalmers.se

Examensarbete 2023

MeadLoggr

En webbapplikation för mjödbryggeri

Oscar Larsson
Sebastian Sela

CHALMERS

Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg 2023

MeadLoggr
En webbapplikation för mjödbryggeri
Oscar Larsson
Sebastian Sela

Handledare: David Lidell, CSE
Examinator: Jonas Duregård, CSE

Examensarbete 2023
Institutionen för data- och informationsteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Institutionen för data- och informationsteknik
Göteborg 2023

MeadLoggr
En webbapplikation för mjödbryggeri
Oscar Larsson
Sebastian Sela
Institutionen för data- och informationsteknik
Chalmers Tekniska Högskola

Sammanfattning

MeadLoggr är en webbapplikation vilken har konstruerats med det primära syftet att förbättra bryggningsprocessen för entusiaster inom mjödbryggning. Målet har varit att tillhandahålla en innovativ lösning som möjliggör enkel dokumentation och organisering av bryggeriprojekt. Genom att definiera tydliga mål har MeadLoggr framgångsrikt uppnått detta primära syfte. En väsentlig aspekt i MeadLoggrs utvecklingsprocess har varit att skapa en interaktiv och användarvänlig plattform. För att uppnå detta har gestaltlagar integrerats tillsammans med principer inom användarupplevelsedesign (UX-design) i applikationens utformning. Dessutom har användartester utförts för att systematiskt utvärdera gränssnittets intuitivitet med avseende på en fördefinierad uppsättning av kriterier. Genom att effektivt tillämpa UX-designprinciper i kombination med gestaltlagarna har MeadLoggr resulterat i en applikation vilken var lätt att använda, hantera och navigera för användare.

Abstract

MeadLoggr is a web application developed with the aim of enhancing the brewing process for individuals interested in brewing mead. The objective was to provide an innovative solution that facilitates the documentation and organization of brewing projects. By defining clear objectives, MeadLoggr successfully achieved this goal. Furthermore, an essential aspect of MeadLoggr was to create an interactive and user-friendly platform. To accomplish this, gestalt laws along with principles from user experience design (UX design) were integrated into the application's development and user testing was conducted to evaluate the intuitiveness of the interface against a collection of predefined criteria. The application of UX-design principles along with the gestalt laws ensured that MeadLoggr was easy to use, manage and navigate for its users.

Nyckelord: React, TypeScript, webbapplikation, full-stack, front-end, back-end, mjöd

Innehåll

1. Inledning

1.1 Bakgrund

Mjödbryggeri är en komplicerad process där små ändringar av parametrar och tillsättning av extra ingredienser kan ha stor påverkan på slutresultatet. Därför ska en webbapplikation för en mjödloggbok skapas. Användarna kommer att ha möjlighet att skapa personliga konton och logga in för att både få tillgång till systemet och skapa projekt. Dessutom kommer de att kunna lägga till anteckningar och observationer för varje enskilt projekt. Webbapplikationen kommer att byggas med hjälp av React, Node.js och PostgreSQL. En fördjupad beskrivning av detta finns i kapitel 4, vilket utgör en avstickare från rapportens huvudsakliga fokus – att skapa en intuitiv gränssnittsdesign genom tillämpning av gestaltlagar och design principer.

1.2 Syfte

Syftet med projektet är att skapa en webbapplikation som underlättar bryggeriprocessen för både företag och privata bryggare, oavsett deras erfarenhetsnivå. Applikationen ska möjliggöra en användarvänlig loggbok som dokumenterar enskilda bryggeriprojekt med ingredienser, jäsning och näringsvärden, vilket ger användaren en enkel och praktisk plattform att organisera och spåra sina projekt med. Denna loggbok kommer att kallas MeadLoggr, och ska tas fram tillsammans med Consat Engineering AB, samt en extern handledare med erfarenheter inom mjödbryggning.

Ytterligare syften med projektet är att utforska och utveckla tekniska färdigheter inom front- och back-end utveckling. Detta inkluderar att undersöka olika webbutvecklingsverktyg.

1.3 Frågeställning

Genom att skapa webbapplikationen och utforska olika tekniska verktyg och metoder, är målet med projektet att förbättra bryggeriprocessen och erbjuda en innovativ lösning för att dokumentera och organisera bryggeriprojekt. Detta kommer att göra det lättare för bryggare att spåra och replikera sina framgångsrika projekt samt lära sig av sina mindre lyckade bryggerisatser.

För att undersöka och utvärdera webbapplikationen berör rapporten följande frågeställning:

- På vilket sätt kan en webbapplikation utvecklas med hjälp av designlagar och kriterier på användbarhet för att skapa ett intuitivt gränssnitt?

1.4 Mål och Omfattning

Projektet kommer att enbart fokusera på en mjödprocess och skulle kunna vidareutvecklas för att täcka in andra processer såsom ölbryggeri. Gällande teknisk implementation kommer programmeringsramverken React och Node.js användas för att hantera front-end och back-end, medan PostgreSQL kommer användas som databashanterare. Ytterligare en detalj är att MeadLoggrs design är tänkt att användas på en skärm med bildförhållande 16:9.

För att uppnå ”minimum viable product” (MVP) för MeadLoggr kommer projektet delas in i två stadier: MVPA och MVPB. Förstnämnda MVPA kommer inkludera grundläggande funktioner för att programmet ska anses som godtagbart. Dessa

inkluderar:

- Skapa ett konto; användaren ska kunna skapa ett konto för att logga in på MeadLoggr och börja logga sin mjödprocess.

- Programmet ska hantera loggning av händelser vilket kommer benämnas events senare i rapporten. Dessa inkluderar mätvärden, tillsatser och kommentarer.

- Planera ett projekt; Vid projektplaneringen tilldelas först ett projektnamn och ett batchnamn tillsammans med en angiven starttid, önskad batchvolym, alcohol by volume (ABV) och söthetsgrad. Därefter genomförs valet av jäst från en fördefinierad lista. Efter detta presenteras en summering av de angivna parametrarna, inklusive den nödvändiga mängden honung i kilogram för projektet samt behovet av näringstillsatser.

- Ändra parametrar i planeringsstadiet. Dessa inkluderar önskad batchvolym, ABV, söthet och val av jäst och därmed uppdateras även mängd honung för projektet och näringstillsatser**.**

- Efter planeringsfasen följer fermenteringsfasen då mjödet jäser. Under denna fas är det viktigt att användaren tillsätter näring i fyra doser inom 24 timmar från det att den tidigare dosen tillsattes, samt en fjärde dos innan 1/3 av att sockret har jäst ut till alkohol. Programmet ska bevaka mängden näring som ska tillsättas samt vilken dos användaren är på och en nedräknande timer för tillsatsfönstret.

- Efter fermenteringen kommer lagerfasen och programmet ska hantera loggning av så kallade ”events” och ändring av batchnamn.

- Projektet avslutas efter lagerfasen och användaren har möjlighet att betygsätta tillsatserna eller avsluta projektet. Under avslutningsfasen presenteras en sammanfattning av projektets olika faser och parametrar, samt en knapp som ger möjligheten till att starta om projektet. Om användaren väljer att starta om hamnar projektet tillbaka i planeringsstadiet.

Sistnämnda MVPB ges prioritet baserat på återstående tid i projektet och inkluderar följande:

- Graf på hur den relativa densiteten jämfört med vatten förändras över tid i fermenteringsfasen.

- En tidslinje för användaren, vilken ger en överskådlig blick på projektets framfart samt när man startade en ny fas och exempelvis loggade ett mätvärde.

2. Teknisk Bakgrund

I den tekniska bakgrunden förklaras hur en mjödprocess ser ut för att underlätta förståelsen för MeadLoggrs implementation samt webbutvecklings-verktygen involverade i utvecklandet av projektet. Till sist redovisas några teorier kring vad som utgör ett användarvänligt gränssnitt.

2.1 Att Brygga Mjöd och TOSNA Calculator

2.1.1 Bryggningsprocess

Mjödbryggnings-processen är en process där huvudsakligen honung, jäst och vatten fermenteras tills att ett önskat resultat uppnåtts, i form av en alkoholhaltig dryck. För att nå resultatet krävs olika tekniker och steg i processen.

Första steget i processen är att ta fram en must, det vill säga en blandning av honung, vatten och eventuella tillsatser av ex. frukt och bär som ingredienser som ska smaksätta mjödet. Musten framtas genom att koka honungen och vattnet under en kort tid på ca. 10–30 minuter - även kallat pastörisering - för att uppnå bästa resultat innan näringsämnen och smaker i honungen bryts ned och för att ta död på bakterier och andra jästar som kan finnas i honungen.

Därefter är det andra steget att tillsätta specifik jäst som ska bryta ned sockermolekyler i musten till alkohol, alltså fermentering. Fermenteringsprocessen kan pågå under några dagar upp till veckor, beroende på hur stark alkoholprocent den slutliga produkten ska ha. Vid fermenteringen kan även extra tillsatser av näring behövas läggas till för att jästen ska fortsätta processen.

Efter fermenteringen startas en lagringsprocess för att mjödet ska utveckla mer smak och få en klar och genomskinlig färg, innan den buteljeras i flaskor. [1][2]

2.1.2 TOSNA 3.0 Calculator

TOSNA är en förkortning för ”Tailored Organic Staggered Nutrient Additions” och är en kalkylator för mjödbryggeri som tillhör webbsidan ”Mead Made Right” [3]. Kalkylatorn är skapad för att ta fram exakta mängder av ingredienser och näringsämnen för olika typer av recept på mjöd, då variation av dessa kan göra stor skillnad för slutproduktens smak och egenskaper. Beräkningarna för TOSNA används i det här projektet för att beräkna mängden av de olika ingredienserna i mjöd.

2.2 Webbapplikationer och Full-Stack Utveckling

Webbapplikationer är en central del när det kommer till internets funktion. I modern tid har en närvaro på internet stor betydelse för företag och organisationer. Handel, bankärenden och informationssökning är tre exempel som numera kan skötas online på respektive hemsidor för att lösa ett problem som tidigare endast hade gått att lösa fysiskt. Webb-applikationer och webbutveckling är det fält som möjliggjort att det kan ske över internet i stället, och att bygga en bra webbsida kräver olika tekniker. En av dessa tekniker kallas full-stackutveckling. [4]

Full-stackutveckling är ett begrepp som används för att beskriva en webbapplikation som är uppbyggd med en front-end, även kallad ”client”, och back-end, eller server, tillsammans med en databas. [6]

2.2.1 TypeScript, React och CSS som Verktyg I Front-End

TypeScript är ett av de vanligaste programmeringsspråken som används vid webbutveckling, framtaget av Microsoft. TypeScript är ett så kallat ”typed superset” av språket JavaScript, vilket innebär att det förstår syntax från JavaScript och lägger till extra funktionalitet. Syftet med TypeScript var att få fram ett språk som kan skala upp ett JavaScript program, och dessutom erbjuder fler verktyg som ex. statisk type-checking. [7]

React är ett OSS, alltså ett ”Open Source Software”, eller open-sourcebibliotek, för JavaScript, framtaget av Facebook, vilket innebär att källkoden finns öppen och tillgänglig för allmänheten. React används för att rendera HTML på en webbapplikation, som sedan tillsammans med front-end kod i antingen Java- eller TypeScript kan knytas ihop för ett funktionellt syfte. [5]

2.2.2 NodeJS och ExpressJS som Verktyg I Back-End

NodeJS är ett ”run-time environment” för JavaScript, som tillåter sammanslagning av flera JavaScriptfiler genom att använda egna moduler. NodeJS använder sig av en ”eventloop”, en händelseslinga, som arbetar som en kö för olika händelser under arbetsgången av koden så att den körs synkront. Dock kan NodeJS vara krångligt när det gäller utveckling på serversidan, vilket blir lättare genom att använda befintliga ramverk som ExpressJS. ExpressJS tillåter att definiera rutter på back-end sidan som gör det enklare att specificera hur http (hypertext transfer protocol) - protokollet ska verka mellan de olika filerna, mellan front-end och back-end och även databasen. [5]

För att exemplifiera hur NodeJS och ExpressJS kan användas skall en enkel bloggplattform utvecklas. Under utvecklingsprocessen används ”run-time environment” för att strukturera koden i separata moduler, såsom hantering av inlägg och användare. Genom att strukturera koden i moduler resulterar det i en ökad möjlighet till återanvändning och enklare underhåll. För att hantera samtidiga anrop för att exempelvis hämta inlägg till bloggplattformen används NodeJS ”eventloop”, vilket möjliggör synkron körning av anropen för att undvika flaskhalsar och blockeringar. Vidare, förenklas kommunikationen mellan front-end och back-end genom användningen av ExpressJS i bloggplattformen. Detta ramverk möjliggör definiering av http-rutter, där varje rutt triggar en specifik åtgärd i respons på klientens begäran. I kontexten av bloggplattformen kan dessa åtgärder exempelvis inkludera att hämta befintliga inlägg eller att skapa nya inlägg. Genom att implementera http-rutter etableras en tydlig struktur som effektivt hanterar kommunikationen mellan front-end och back-end.

Sammanfattningsvis är NodeJS en miljö där JavaScript kan köras utanför webbläsaren. Det möjliggör sammanföring av JavaScript-filer genom egna moduler. NodeJS använder en händelseloop för att hantera olika händelser och möjliggör körandet av kod synkront. Utveckling på serversidan med NodeJS kan vara komplicerat, men detta underlättas med hjälp av befintliga ramverk som ExpressJS. Genom att använda ExpressJS underlättas hanteringen av rutter på serversidan, vilket definierar hur http-protokollet ska fungera mellan olika filer.

2.2.3 PostgreSQL som Databas
Även PostgreSQL är ett OSS, som har ett gott rykte och en hög användargrad bland utvecklare. PostgreSQL har en stor intressegrupp som samarbetar och bidrar till att bra dokumentation finns tillgänglig. Det finns också många fördelar med att använda PostgreSQL inom företag då det som tidigare nämnt är gratis, lätthanterligt och lättläst och skalbart för större applikationer. [8]

2.2.4 APIer och Kommunikation mellan Front-End och Back-End
Axios är ett ”application programming interface” (API) som använder en ”promise based” http-klient för NodeJS och webbläsaren. Begreppet promise based kommer ifrån JavaScript och beskriver hur JavaScript skapar objekt, så kallade promises, som i generella drag hanterar och reducerar problem som tillkommer med callback-funktioner. [10][7]

CORS är ett annat API som står för ”cross origin resource sharing”, som används för att skicka http-requests mellan domäner för front-end till back-end. JavaScript förhindrar dessa typer av requests då JavaScript har ett ”same origin policy”, vilket är en säkerhetsåtgärd för webbapplikationer, som begränsar kommunikationen till en enda domän. Med hjälp av CORS kan ett http-request skickas mellan olika domäner, vilket är en fördel om back-end finns på en server och ska kommunicera med front-end på en annan domän. [9]

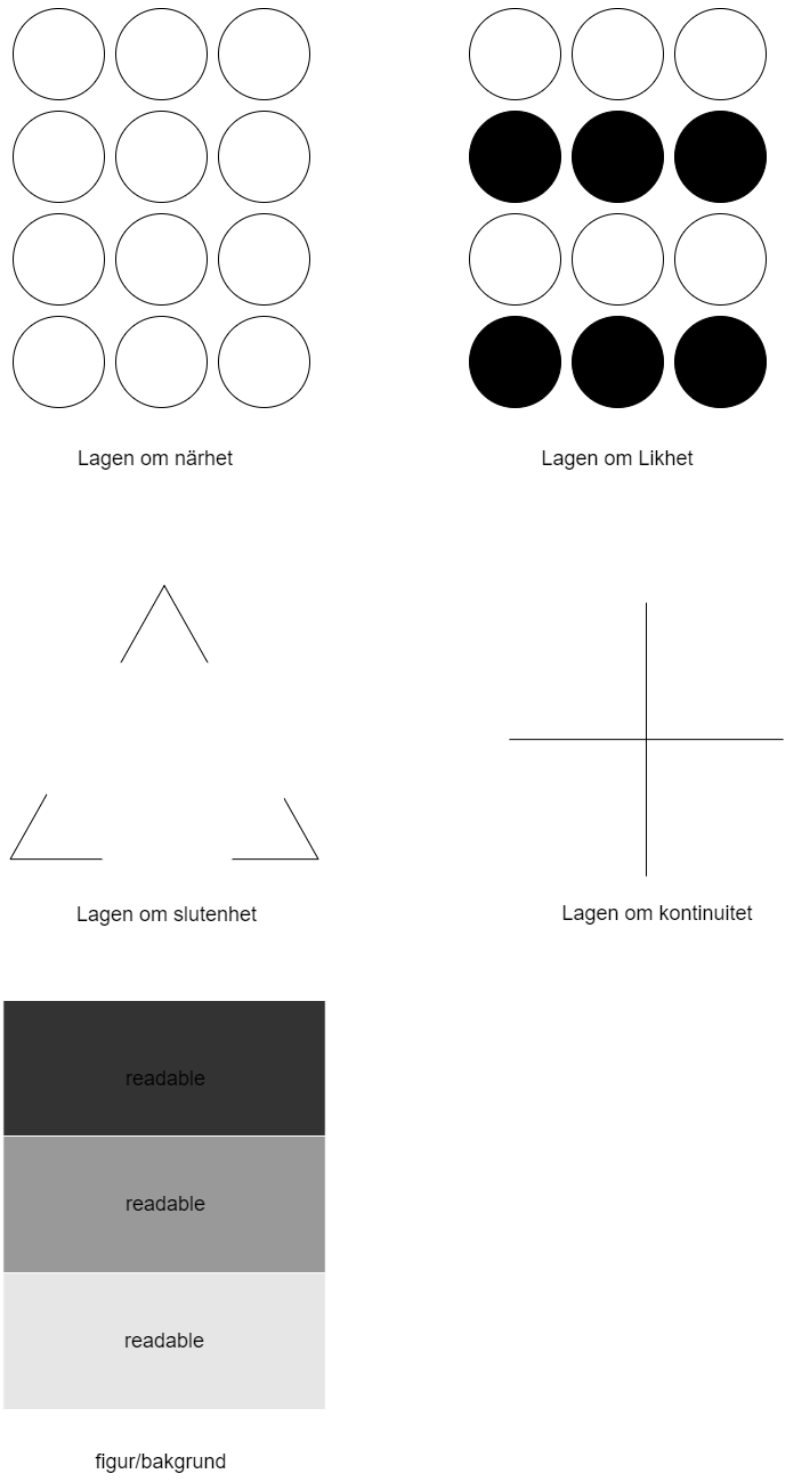
2.3 Teoretiska Grundstenar för ett Användarvänligt Gränssnitt
Skapandet av ett tydligt och lättnavigerat gränssnitt är nyckeln till en framgångsrik webbapplikation. Om detta inte uppnås kan det bidra till en hög inlärningskurva till webbapplikationen och en ökad frustration hos användaren. För att motverka en hög inlärningskurva till gränssnittet samt systematiskt skapa en intuitiv design kan gestaltlagar och principer för lättnavigerat gränssnitt tillämpas. [11]

2.3.1 Gestaltlagarna
Genom att tillämpa gestaltlagarna kan man skapa en visuell struktur som underlättar användarens interaktion med gränssnittet. Inom design finns en mängd olika gestaltlagar att beakta, men för denna rapport kommer vi att fokusera på ett utvalt antal som anses vara fundamentala för att skapa ett intuitivt gränssnitt. Dessa inkluderar lagarna om närhet, likhet, slutenhet, kontinuitet och figur/bakgrund. Närhetslagen innebär att former som är placerade nära varandra uppfattas som en grupp. Likhetslagen återspeglar att objekt som delar liknande egenskaper, såsom form och färg, tenderar att grupperas visuellt tillsammans. Kontinuitetslagen innebär att objekt som placeras i närheten av varandra i en kontinuerlig linje eller rörelse upplevs som en sammanhängande enhet eller mönster. Slutenhetslagen innebär att former uppfattas som helheter baserat på arrangemanget eller placeringen av deras element i förhållande till varandra. [11]

Den sistnämnda principen figur/bakgrund, syftar till att separera en tydlig figur från dess omgivande bakgrund för att förbättra användbarheten och användarupplevelsen. Kontrast spelar en avgörande roll inom denna princip då synliga bilder och texter är kritiska för läsbarheten och förståelsen av innehållet på en webbsida. Ett praktiskt exempel på användningen av figur/bakgrund kan observeras i textlänkar på diverse webbapplikationer där färgförändringar, vid muspekarens rörelse över länkarna, ger visuell återkoppling till användaren. För att framgångsrikt tillämpa principen inom webbapplikationer är det av

största vikt att skapa tydliga kontraster och använda en väldefinierad hierarki i texten för att underlätta läsbarheten och användarinteraktionen. Skapandet av en tydlig hierarki inkluderar också användningen av olika färger samt figurer med varierande storlek för att tydliggöra vad som förväntas av användaren. [11]

Sammantaget syftar gestaltprinciperna till att organisera visuell information på ett sätt som gör det enkelt att tolka och förstå för användaren. En fiktiv visuell representation av lagarna ges av figur 1.



Figur 1: Visuell representation av gestaltlagarna.

2.3.2 Principer för Användarvänligt Gränssnitt

För att utveckla ett användarvänligt gränssnitt finns en uppsättning designprinciper att förhålla sig till. Nyckeln är att identifiera och förstå efterfrågan hos typiska användare av programmet vilket i sin tur medför en förståelse för hur gränssnittet kan struktureras. En viktig aspekt att ta hänsyn till är att alla användare på internet inte kommer ha nytta av webbapplikationen [12]. Genom att förstå den typiska användargruppen blir utvärderingen lättare under användartester samtidigt som användarens mentala bild om hur applikationen bör fungera överensstämmer med hur den faktiskt fungerar. [12]

Efter identifikation av typiska användare till programmet går fokus över till nästa princip: att konstruera ett lättnavigerat gränssnitt med tydliga namnsättningar på exempelvis knappar och övriga element i gränssnittet som användaren integrerar med. Genom användningen av tydliga namnsättningar leds användaren vidare till nästa avsnitt på ett smidigt sätt. För att säkerställa en smidig navigation bör hinder som distraherar användarens uppmärksamhet undvikas, såsom överflödig eller osammanhängande information. I stället bör en fokuserad design eftersträvas, som riktar användarens uppmärksamhet mot relevanta uppgifter och funktioner. Denna design konstrueras med hjälp av de tidigare beskrivna gestaltlagarna. En fokuserad design resulterar också i att webbapplikationens delar har en enhetlig och logisk struktur för att underlätta användning. Liknande funktioner skall eftersträva en konsistent struktur, vilket gör det intuitivt för användaren att navigera och utföra uppgifter i applikationen. [12]

Nästa designprincip är en annan viktig aspekt av att förstå användargruppen och innefattar tidigare kunskaper en användare kan tänkas ha gällande webbapplikationer. Den tidigare erfarenheten kan utnyttjas på ett sådant sätt att möjligheten att upprepa en liknande uppgift förenklas och minskar inlärningskurvan till programmet. För att exemplifiera: vid besök på en webbshop förväntas kundvagnssymbolen vara placerad uppe i högra hörnet och vid klick på symbolen visa tillagda objekt. Med hjälp av tidigare erfarenhet från andra webapplikationer och lärdom från användarens första försök ska det vara lätt att reproducera förväntat resultat av en funktion vid interaktion med webbapplikationen. [12]

Sammanfattningsvis, för att utveckla ett användarvänligt gränssnitt kan följande design principer användas:

- Användaridentifiering och målgruppsförståelse: en lyckad design utgår från att förstå och adressera användarnas specifika behov och förväntningar.

- Tydliga namnsättningar och navigationsvägar: klara namnsättningar och intuitiva navigationsvägar skapar en smidig användarupplevelse genom applikationen.

- Fokuserad design: genom att undvika distraherande element och fokusera på relevant information och funktioner förbättras användbarheten.

- Tillämpning av gestaltlagar: Genom användningen av gestaltlagarna vid konstruktion av ett gränssnitt skapas en logisk design som underlättar interaktion och förståelse.

- Enhetlig struktur: en konsekvent struktur för liknande funktioner gör det enkelt för användare att orientera sig och utföra uppgifter.

- Användarnas tidigare erfarenheter: tillämpa kunskaper en användare kan tänkas ha gällande webbapplikationens funktioner i konstruktion av gränssnittet för att minska inlärningskurvan och skapa en bekant och användarvänlig miljö.

3. Metod

3.1 Planeringsfasen

3.1.1 Förundersökning och Tillvägagångssätt för att Brygga Mjöd

Innan projektets start behövdes en förundersökning göras om hur hela bryggeriprocessen går till. Med hjälp av en extern handledare med erfarenheter inom mjöd-bryggeri kunde ett tillvägagångssätt fastställas. Handledaren bidrog även med information angående formler och samband mellan de olika ingredienserna i mjöd.

3.1.2 Planering

När projektet skulle startas sattes en tidsplan upp i form av ett Gantt-schema (se bilaga 8.1). Detta schema sattes upp så att det skulle finnas en rimlig mängd arbetsuppgifter fördelade under en viss tid, och hur många arbetsdagar som det rimligtvis skulle krävas för att slutföra alla arbetsuppgifter.

3.2 Verktyg

3.2.1 Inlärning av Programspråk

Valet av programspråk föll på bland annat TypeScript, React och CSS då de är vanliga inom webbutveckling. Trots att det finns flera andra programspråk, valdes dessa på grund av en rekommendation av arbetsgivarna på Consat. Då programspråken inte var kända sedan tidigare avsattes en period innan projektets start för att söka information och kunskap om dessa, samt starta upp en Git.

3.2.2 Git

Git användes till att dela filer på ett smidigt sätt i ett gemensamt förvar (Engelska, repository) där alla gruppmedlemmar har tillgång till koden genom att ladda ner den lokalt till egen dator, samt ladda upp filerna igen så andra kan ladda ner en nyare version. På så vis underlättade det processen att inte behöva mötas, antingen på arbetsplats eller digitalt, för att arbeta och utveckla programmen. Det går även att bryta upp koden i s.k. ”branches” eller grenar, där utveckling kunde ske parallellt utan att det störde någon annans arbete tills det var dags att sammanfoga koden.

3.2.3 Scrum och Agilt Arbetssätt

Det agila arbetssättet är en metod för att organisera och genomföra projekt genom att dela upp arbetet i korta perioder kallade sprintar. Varje sprint pågår i en viss bestämd tid där ett antal uppgifter, kallade tasks och user stories, ska utföras för att nå ett mål. Tasks är mindre deluppgifter som hjälper till att nå en större user story, som beskriver ett problem eller ett mål som utvecklaren har. Efter varje sprint hölls ett möte för att diskutera vad som har uppnåtts och, i vissa fall, vad som behöver förskjutas till nästa sprint om målen inte uppfylldes. Ett vanligt verktyg för agilt utvecklande är Jira, tillverkat av Atlassian.com, som används för att hålla reda på uppgifter, tilldela dem till olika medlemmar i projektet och bevaka deras status. På Jira finns också en digital tavla där det går att lägga till, ta bort eller flytta uppgifter. Projektet delades upp i ”minimal viable product A” (MVPA) och ”minimal viable product B” (MVPB), för att bryta ner

huvudfunktionerna och en gemensam grund som minst krävs för att applikationen ska anses som färdig och funktionell.

3.2.4 Utvecklingsmiljöer

För att utveckla webbapplikationen används intelliJ som utvecklingsmiljö. intelliJ pro stöder TypeScript som standard för både utveckling och felsökning. Ytterligare funktionaliteter är att man i utvecklingsmiljön kan ansluta och programmera databasen via query console. Fördelen med att hantera utvecklandet i en och samma utvecklingsmiljö är att det gör en överskådlig bild av vad som händer hela vägen från front-end till back-end och databas då intelliJ möjliggör ett kontinuerligt övervakande av databasens tabeller.

3.2.5 Figma och Grafiska Gränssnitt

För att underlätta arbetet med mjödloggbokens grafiska gränssnitt skapades en prototyp i Figma vilket tillåter skapandet av trådskisser för att se hur sidorna formaterar sig på olika skärmstorlekar innan själva utvecklandet började. Med hjälp av Consat kunde en grundlig prototyp för MeadLoggr fastställas, vilket underlättade processen när prototypen skulle översättas till kod i React, och mycket av förarbetet gick åt till att skapa en så välutvecklad prototyp som möjligt och ha kontinuerliga design-möten och återkopplingar med Consat. Prototypen gjordes interaktiv, där det fanns en möjlighet att skriva in värden i designen för att få fram en bättre känsla av funktionaliteten som sedan implementerades i React.

3.3 Användartest och Utvärdering av Resultat

3.3.1 Användartest

För att kunna utvärdera frågeställningen har en uppsättning av uppgifter skapats och tilldelats till testpersoner att lösa med hjälp av gränssnittet. Testerna gjordes live på plats eller via zoom tillsammans med testpersonerna för att lätt kunna se och dokumentera reaktioner eller specifika objekt i gränssnittet testpersonerna fastnade på. Fortsättningsvis, genomfördes tester på både Figmaprototypen och på den faktiska webbplatsen för att ge insikt i hur utvecklingen bör fortskrida för att uppnå en så användarvänligt gränssnitt som möjligt, i enlighet med den agila arbetsmetoden. Vidare för att utvärdera användartesterna och erhålla resultat om gränssnittets intuitivitet och effektivitet, togs en uppsättning kriterier fram från principer för användarvänligt gränssnitt, avsnitt 2.3.2:

- Ett lättnavigerat gränssnitt med tydliga namnsättningar som leder användaren vidare till nästa avsnitt. Till exempel bör "Hem"-knappen leda användaren till startsidan.

- Användaren bör kunna känna igen och använda webbapplikationens funktioner baserat på tidigare vanor och kunna upprepa samma uppgift. Till exempel skall användaren kunna skapa ett nytt projekt i MeadLoggr med hjälp av erhållna erfarenheter från mjödbryggeri eller genom informationsrutorna på webbapplikationen och kunna reproducera försöket.

- Användaren bör kunna navigera genom webbapplikationen effektivt utan hinder som distraherar användarens uppmärksamhet på grund av överflödig eller osammanhängande information.

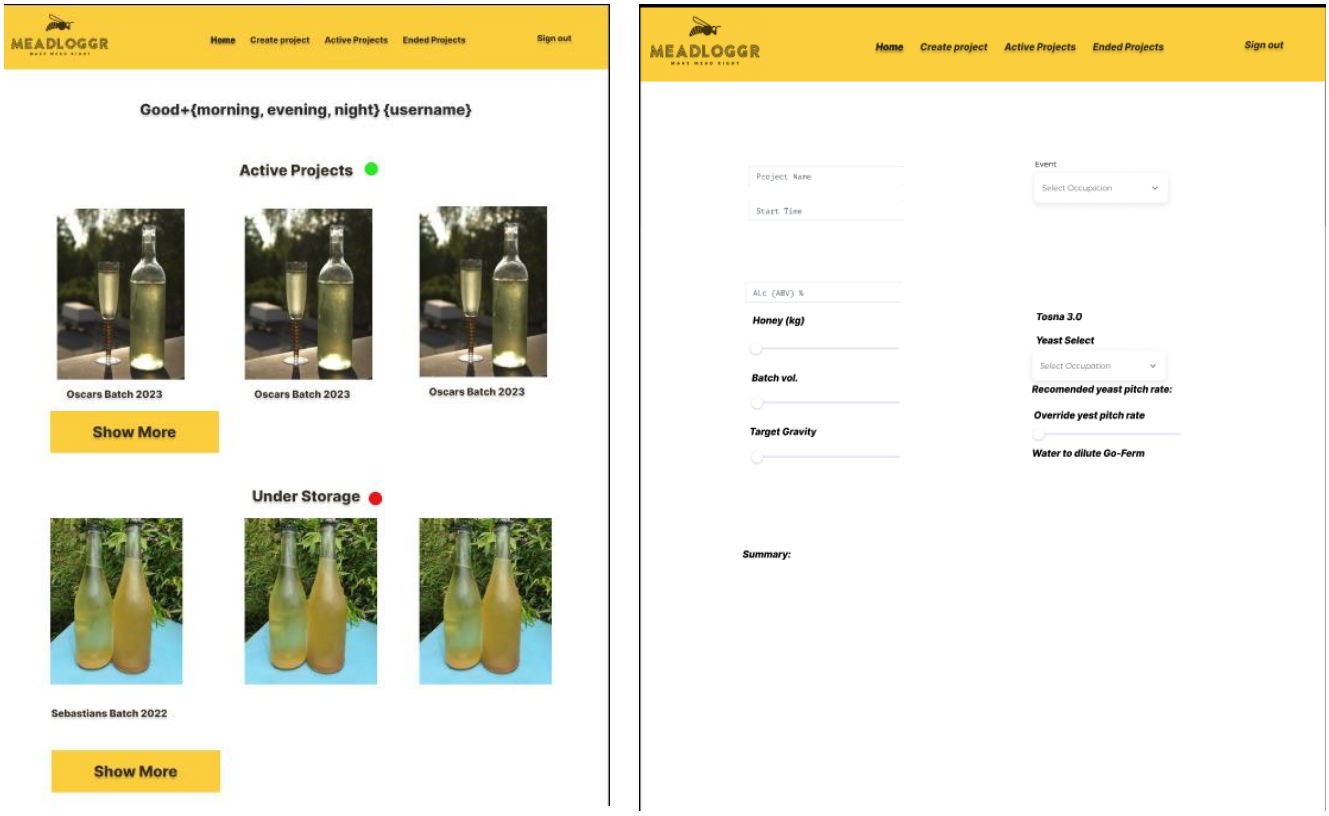
- En sammanhängande struktur bör finnas, enligt tidigare nämnda gestaltlagar, vilket gör olika delar av webbapplikationen lätt att hantera. Liknande funktioner bör ha en liknande struktur för att underlätta användning.

4. Systemkonstruktion

Som nämnt i inledningen är detta kapitel en avvikelse från rapportens huvudsakliga fokus, för att gå in på fler detaljer om hur MeadLoggr byggdes upp.

4.1 MeadLoggrs Prototypgränssnitt

I metodkapitlet diskuterades hur en interaktiv prototyp i programmet Figma skapades innan den skulle byggas ordentligt i front-end. Nedan följer två bilder tagna från prototypen:



Figur 2: Home page (t.v.) och Create MeadLog (t.h.) från Figma prototyp.

Figur 2 illustrerar MeadLoggrs *home page*, även kallad startsida, samt *Create MeadLog*. Överst finns ett navigeringsfält som möjliggör användarens övergång till andra delar av applikationen. Sidornas olika kopplingar och beroenden knyts samman i navigeringsfältet, och genom att klicka på en av knapparna tar användaren sig vidare till önskad sida. Det slutliga gränssnittet av MeadLoggr kommer att presenteras nedan i resultatet.

4.2 MeadLoggrs Tekniska Implementation

I den tekniska implementationen återges hur MeadLoggr programmerades och med vilka designmönster samt hur databasen skapades.

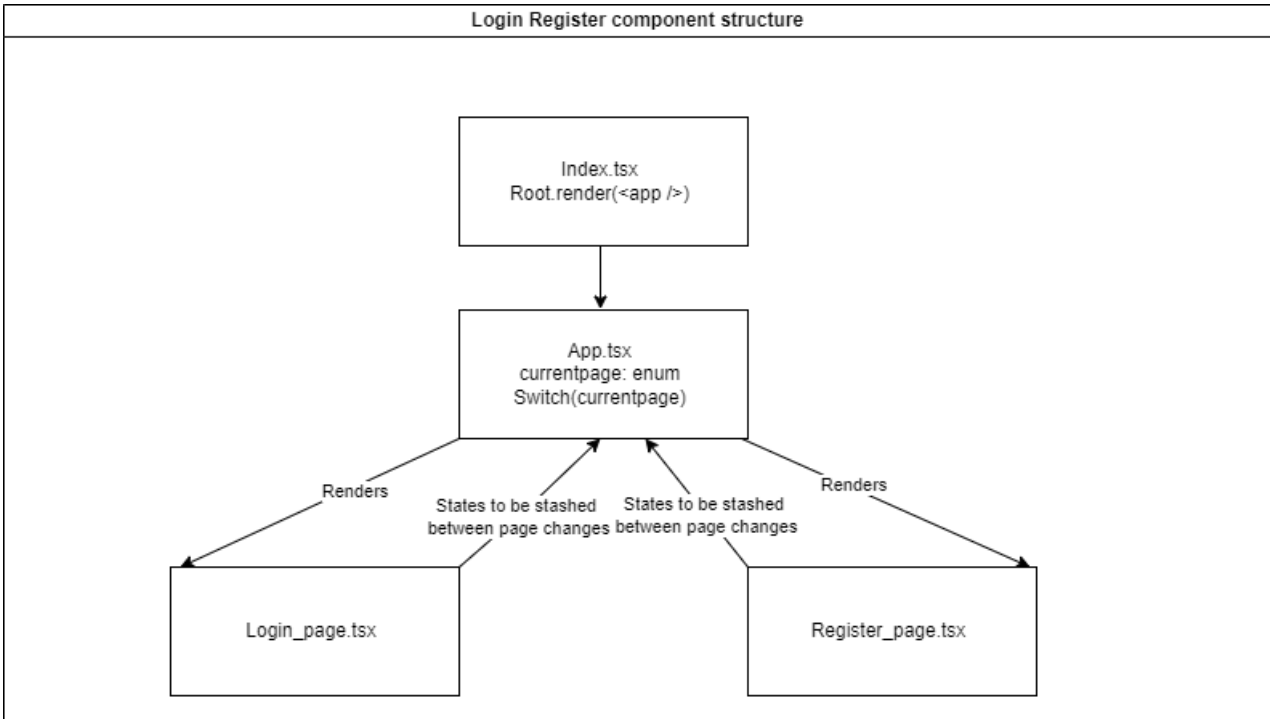
4.2.1 Designmönster

I front-end, vilket programmerades i React, användes view-controller designmönstret medan modellen placerades i back-end och implementerades i NodeJS. Front-end hämtar information från back-end med ett http-getanrop och uppdaterar i sin tur vyn. Vid användarinmatningar till gränssnittet tas dessa ändringar emot i view controller och

skickar vidare informationen via ett http-postanrop till modellen som i sin tur hanterar databasen.

4.2.2 Front-End Uppbyggnad

React är ett JavaScript-bibliotek som bygger på att gränssnitt utgörs av komponenter. För att minimera kod är det önskvärt att skapa generiska komponenter vilka kan återanvändas många gånger under utvecklingen. MeadLoggrs gränssnitt har en egen komponent för varje ny sida som återfinns i programmet. Alla sidor ligger på samma komponentnivå med en komponent App över sig vilket är en controllerkomponent som hanterar rendering av aktuell sida och sparningar av tillstånd mellan sidövergångar. Över app ligger indexfilen vilken renderar appinnehållet på html dokumentet. I figur 3 presenteras hur loginsystemet är uppbyggt med vilka komponenter och hur de kommunicerar. MeadLoggr utvidgades med samma logik för att inkludera fler sidor. Vissa sidkomponenter har ytterligare underkomponenter i sin hierarki för att effektivt separera olika funktioner, vilket möjliggör återanvändning av dessa komponenter på flera platser. Ett exempel är *mead_calculator*, vilken används på både *Create MeadLog* -sidan och *calculator*-sidan. Vidare diskuterades i avsnitt 4.2.1 att view controller placeras i front-end. Detta innebär att varje sidkomponent är en kombination av view och controller. På så sätt kan enskilda komponenter, beroende på den implementerade funktionen, skicka data till modellen i back-end.



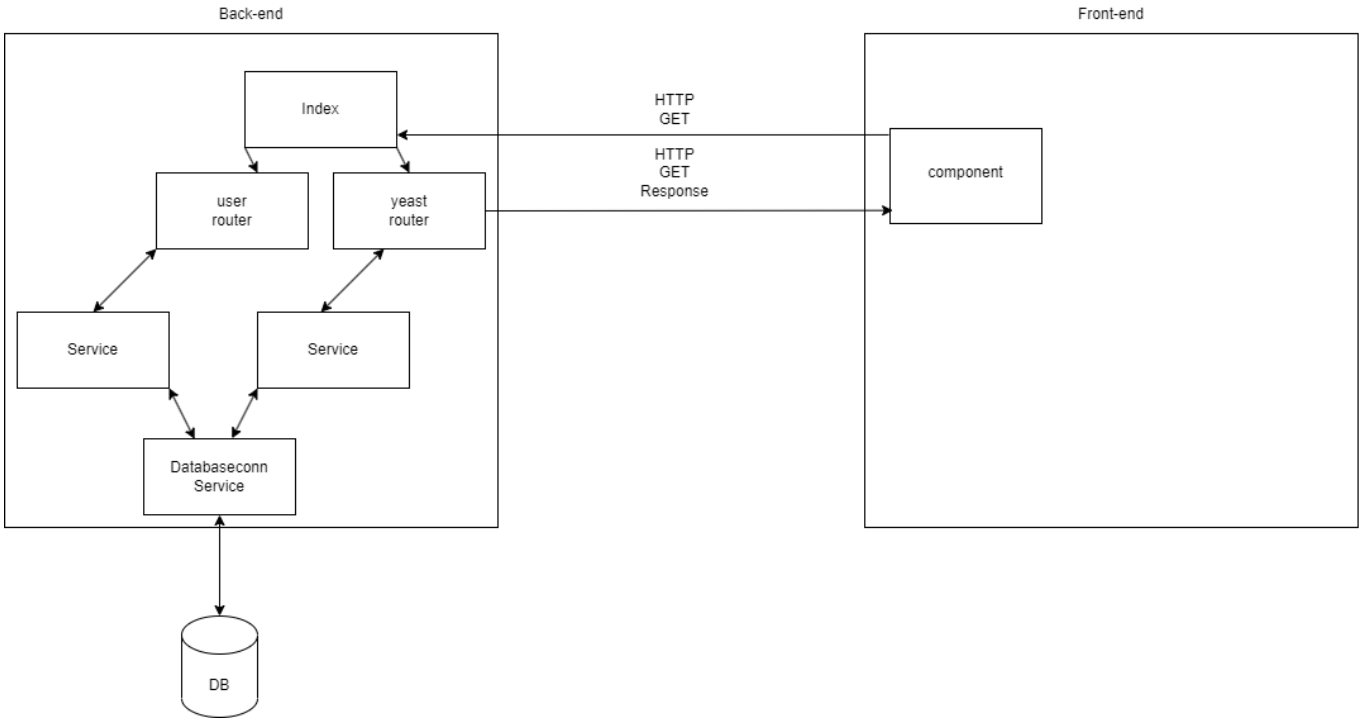
Figur 3: komponentstruktur av login systemet

4.2.3 Back-End Uppbyggnad

I back-end återfinns indexklassen som alla http-request anländer till. Index i sin tur dirigerar förfrågningen vidare till antingen user eller yeast routerklassen beroende på URL:en. Varje router har en uppsättning metoder beroende på den angivna åtgärden och har en serviceklass som hanterar logiken samt hämtar och sätter in information i databasen. När serviceklassen har utfört en åtgärd returnerar routern en http-kod eller data till front-end beroende på utfallet.

För att kunna hantera databasen återfinns ytterligare en serviceklass *Databaseconn* som hanterar en anslutningspool. Poolen innehåller klienter som utför queries på databasen. När den använda klienten är klar ”släpps” den tillbaka in i poolen för återanvändning. *Databaseconn* följer singleton principen eftersom den enbart ska ha en anslutningspool och optimerar därför resursanvändandet då det är en onödig kostnad att skapa flera pooler från en prestandasynpunkt. I figur 4 presenteras ett blockdiagram på en http-get. URL:en kan ha haft följande sökväg **http://localhost:8080/yeast/getyeastinfo** och när request:en anländer i index dirigeras den vidare till yeast router på grund av */yeast* i URL:en. I yeast router exekveras metoden *getyeastinfo* som använder sig av service för att utföra query på databasen och returnerar resultatet till *getyeastinfo* som i sin tur returnerar en http-kod 200 tillsammans med jästinformationen i en array med http-get svaret till front-end. Om infon om jästen däremot inte kunde hämtas på grund av SQL exception eller liknande returneras koden 400 i http-get svaret till front-end. Detta exempel fungerar som en bra representation för att förstå hur back-end utvidgas för att täcka in andra http-requests.

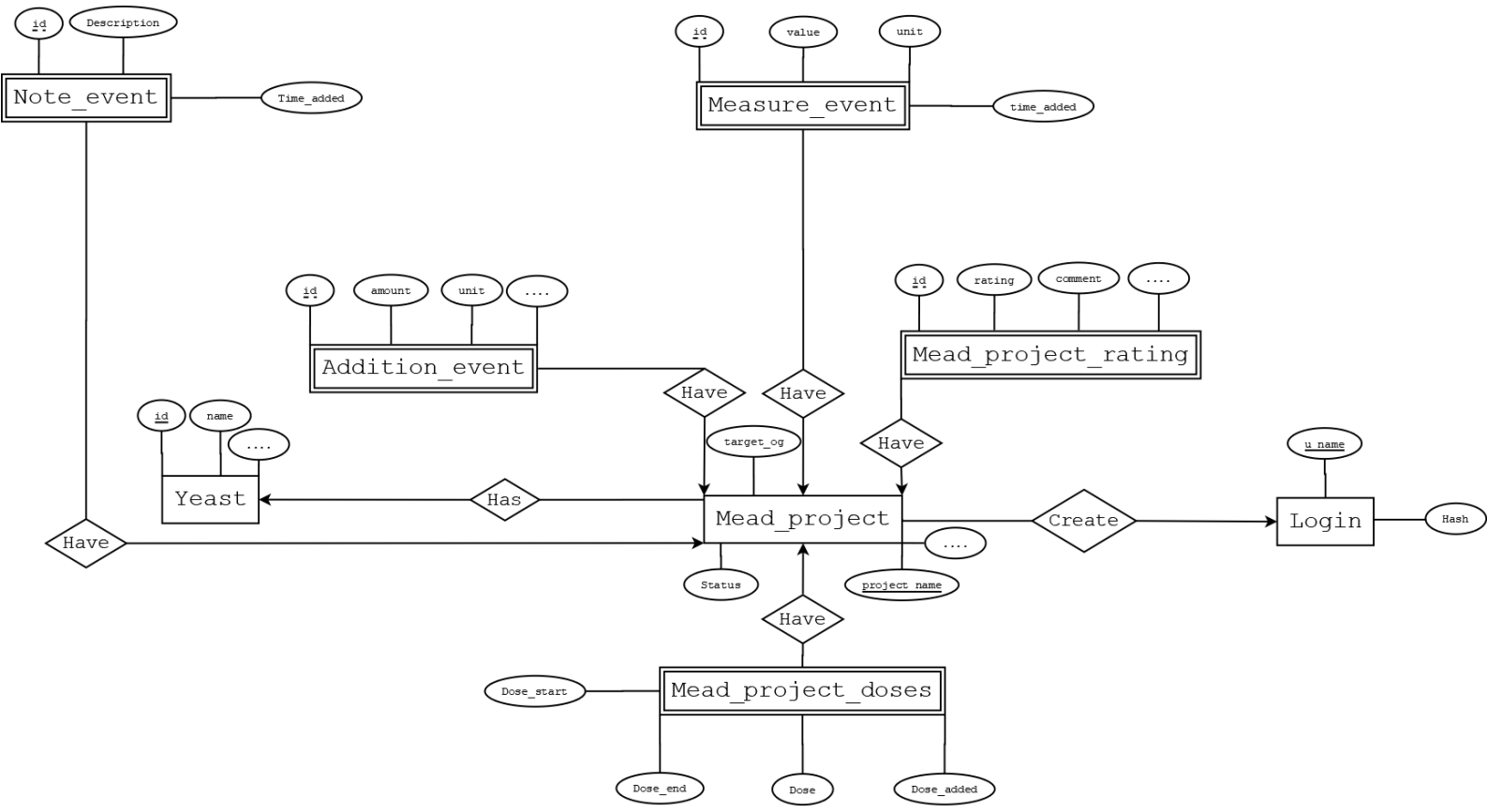
Sammantaget återfinns en routerklass för det övergripande objektet vi ska utföra en åtgärd på. Varje routerklass har en uppsättning av serviceklasser vilka sköter logik och använder sig av databaseconn som interface mot databasen.



Figur 4: Överskådligt blockdiagram på http kommunikationen mellan front och back-end.

4.2.4 Databasens Uppbyggnad

Vid konstruktion av databasen skapades ett ER-diagram. I figur 5 visas hur olika typer av jäst modelleras i databasen. Administratören lägger in nya jästar med egenskaper och front-end hämtar aktuell jäst tillsammans med dess attribut. Vidare i figur 5 beskrivs hur sparningar av mjödprojekt går till. Mjödprojekten går igenom olika faser och under dessa kan användare registrera events. Det återfinns tre olika typer av events: *Note_event*, *Measure_event* och *Addition_event*. Alla event tables är unika till exakt ett mjödprojekt. Under fasen fermentation ska näringsdoser adderas till mjödet. För att beskriva detta finns *Mead_project_doses* table som är kopplad till ett unikt mjödprojekt. På liknande vis är betyget ett mjödprojekt fick i fasen ended kopplat till ett unikt mjödprojekt. Det skall noteras att events, *Addition_event* och *Mead_project_rating* har kopplingar till flera mjödprojekt men inte kan existera utan att ett projekt skapas. Därför är alla dessa svaga entiteter i diagrammet. Slutligen återfinns tabellen *Login* i figur 5 vilket sparar information om användare.



Login har användare associerade med hash för att utföra inloggning. I webbappen så exkluderades en *Settings*-tabell på grund av tidsbegränsningar. Därför refererar ett *Meadprojekt* till användare från *Login*-tabellen och inte en *användar*-tabell.

Status ENUM (“CREATED”, “FERMENTATION”, “STORAGE”, “ENDED”).

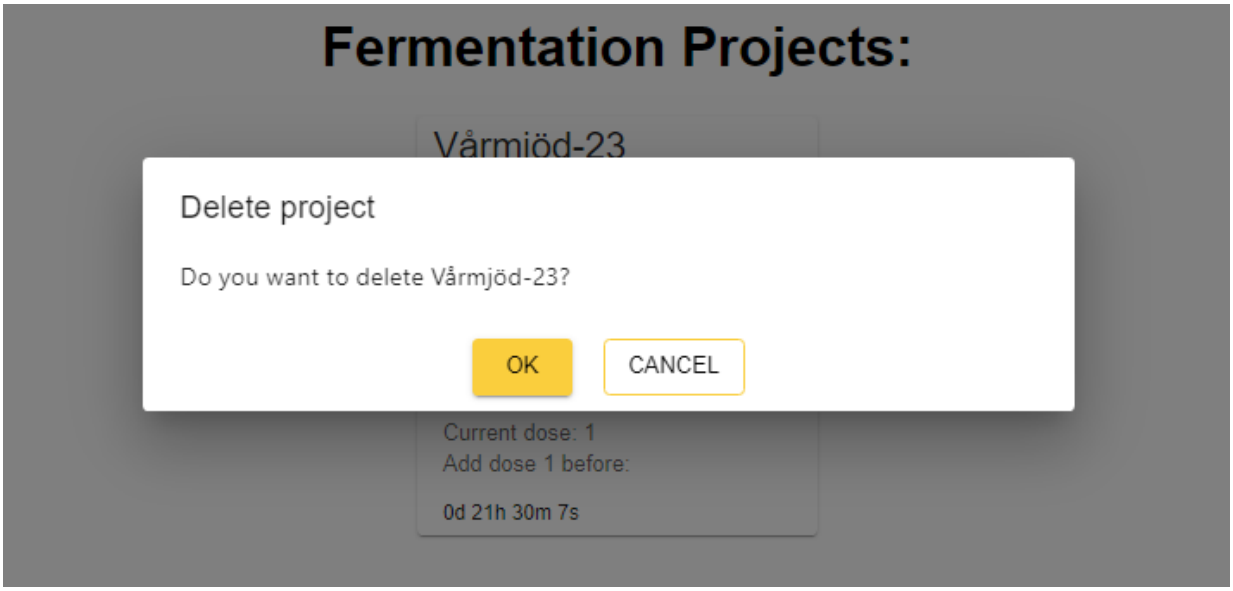
Figur 5: ER-diagram på hur ett meadprojekt modelleras och information relaterade till användare. Attribut …. innebär att entiteten har flera attribut än vad som visas i diagrammet.

5. Resultat

För att kunna utvärdera frågeställningen, på vilket sätt en webbapplikation kan utvecklas med hjälp av designlagar för att skapa ett intuitivt gränssnitt presenteras resultatet av designen och hur principer för användarvänligt gränssnitt och gestaltlagarna har tillämpats. Till sist ges en redogörelse för vad testpersonerna tyckte om gränssnittet.

5.1 MeadLoggrs Slutgiltiga Design

För att säkerställa att användaren känner igen sig i applikationen används ett enhetligt färgtema. Knappar och andra objekt har en gul/orange färg medan beskrivande texter är svarta. I figur 6 visas ett exempel på denna design, vilken syftar till att undvika förvirring genom att använda olika färgvariationer. För att fängsla användarens uppmärksamhet är knapparna framträdande i storlek och färgton, och specifikt i dialogrutor där användaren måste välja mellan exempelvis "OK" och "CANCEL," färgas "OK"-knappen helt gul för att omedelbart dra till sig användarens blick och ge den högre prioritet än "CANCEL"-alternativet vilket enbart har en gul ram. Design genomförs konsekvent för att skapa tydliga kontraster och säkerställa läsbarheten på alla sidor enligt gestaltlagen figur/bakgrund. *Startsidan* och *Create Meadlog* utgör de två huvudsakliga designmallarna som återanvänds på övriga sidor i applikationen.



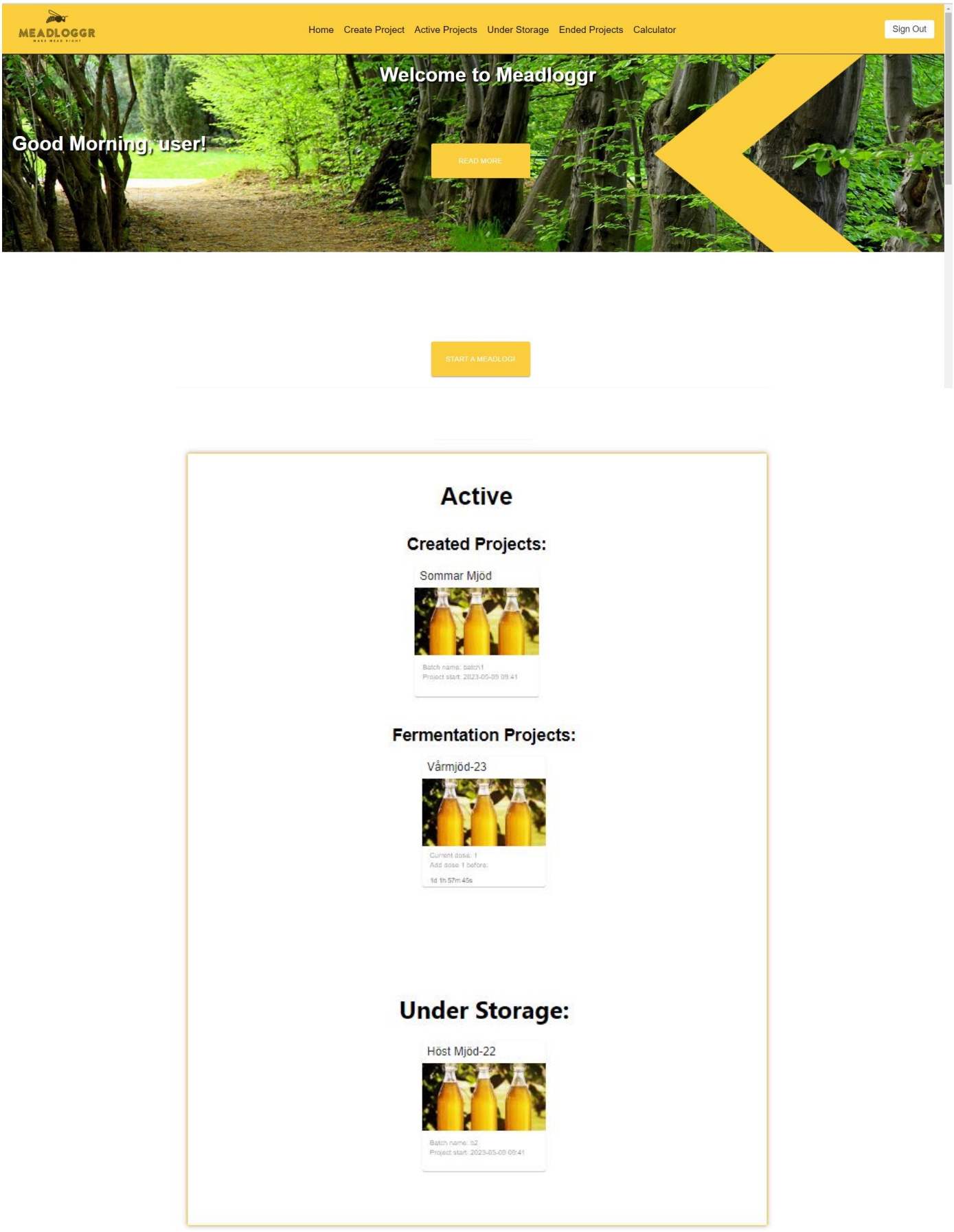
Figur 6: Dialogruta med CTA-knapp

5.1.1 Startsidan

Från startsidan (se figur 7 nedan) finns ett fast navigeringsfält längst upp på hemsidan, vilket innebär att navigeringsfältet har samma position längst upp på sidan även om användaren skulle skrolla nedåt. Från navigeringsfältet kan användaren ta sig till de olika delarna genom att klicka på respektive knapp, och alltid ta sig till startsidan genom att klicka på *Home* eller logotypen för MeadLoggr. I navigeringsfältet har principen om tidigare erfarenheter tillämpats då det är naturligt från tidigare appar att *log out* återfinns i högra hörnet. Även vid ett klick på logotypen för MeadLoggr, vilken är placerad till vänster, ska navigering till startsidan ske.

Under navigeringsfältet återfinns en bannerbild med en hälsningsfras till användaren och vid klick på *show more* presenteras en introduktion till webbsidan. Anledningen till att detta avsnitt ligger under navigeringsfältet vid första anblicken är för att det anses som logisk och i ”rätt ordning” att presentera hur applikationen används. Fortsättningsvis, döljs texten med *show more*-knappen för att undvika överflödig information vid ett frekvent användande av gränssnittet.

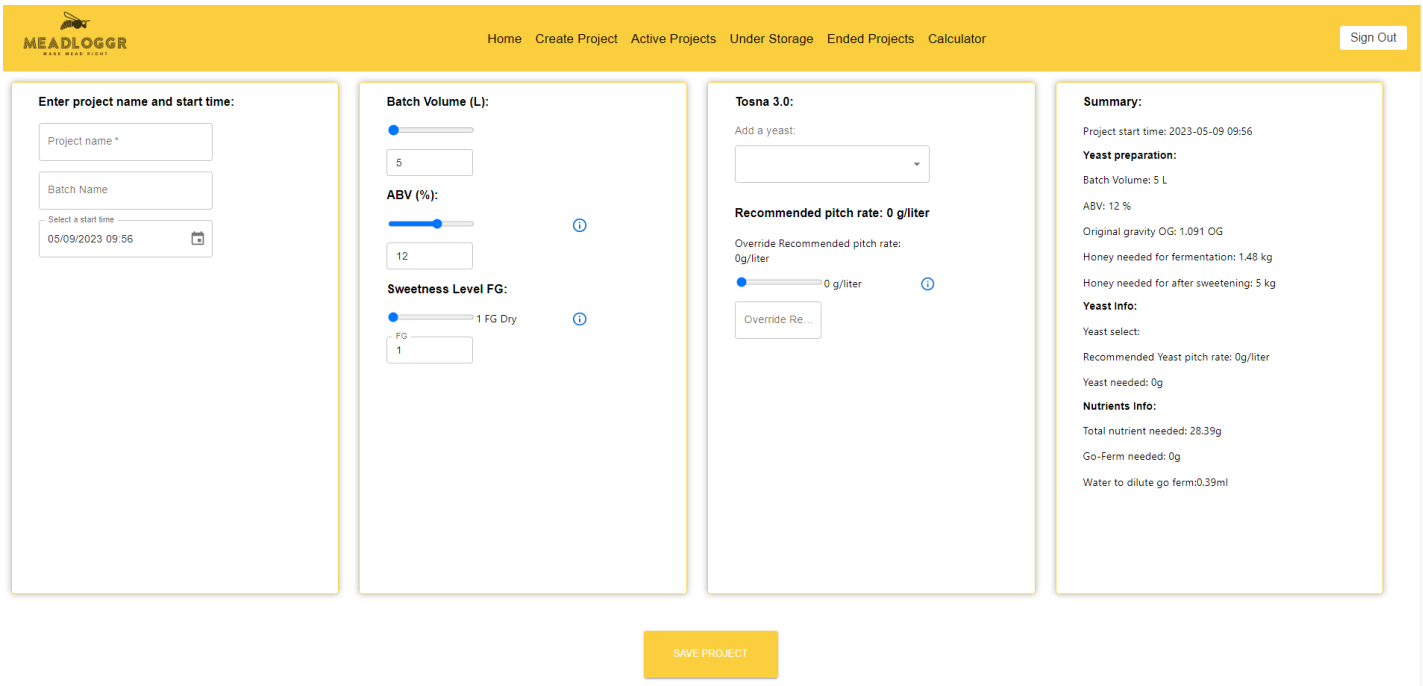
Vid skroll nedåt på sidan under bannerbilden återfinns en Call to Action (CTA)-knapp, vilket är en interaktiv knapp eller länk på en webbsida med syfte att locka användaren till att vidta en specifik åtgärd eller handling. I detta specifika fall är CTA-knappen utformad för att guida användaren till att navigera till sidan där man kan skapa ett mjödprojekt. Det anses vara näst viktigast efter presentationstexten. Under CTA knappen återfinns användarens mjödprojekt. För att lätt kunna se var de olika projekten är i processen tillämpas en hierarki av olika textstorlekar. I figur 7 inkluderar *Active* både *Created projects* och *Fermentation Projects.* Lagen om kontinuitet har tillämpats för att alla dessa kategorier ska hänga ihop då de har placerats på linje under varandra. Om flera projektkort återfinns i samma kategori, till exempel *Fermentation Projects*, placeras den senast skapade alltid till vänster horisontellt intill befintligt projektkort. Vidare för att markera att projekten inom varje kategori hör ihop har således lagen om närhet tillämpas.



Figur 7: Startsidan

5.1.2 Create Project

Sidan *Create MeadLog* är central för hela applikationen (se figur 8). På sidan kan användare planera ett projekt och mata in önskade värden på en sats när det gäller ex. satsvolym och ABV och sedan få ut i *summary*-spalten hur mycket av varje enskild ingrediens det behövs för att nå det önskade resultatet. Informationen är uppdelad i fyra olika kategorier med en egen kolumn. Inom kolumnen tillämpas lagen om kontinuitet för att markera att respektive inmatningsfält tillhör den specifika kategorin. Mellan kolumner tillämpas lagen om närhet. Designen bygger även på att presentera information i den ordning en bryggare kan tänkas vilja mata in informationen och börjar därmed att samla in allmän projektinformation för att sedan övergå i mjödspecifika inmatningar och avslutar med en sammanfattning. Slutligen har informationssymboler placerats ut för att tydliggöra begrepp inom bryggning för nybörjare. Genom att klicka på dessa symboler, baserat på deras förväntade tidigare erfarenheter av symbolen, kan användarna öppna informationstexten för att få mer detaljer. Informationssymbolerna hänger ihop med respektive begrepp genom lagen om kontinuitet eftersom begrepp med motsvarande informationssymbol ligger på en horisontell linje.



Figur 8: Create MeadLog sidan

5.2 Resultat från Användartester

5.2.1 Extern Handledare inom Mjödbryggning

Den externa handledarens feedback berörde mestadels tekniska aspekter gällande mjödprocessen. Värt att notera är att sista användartestet på den externa handledaren gjordes innan en uppnådd MVPA. Under testet var feedbacken att göra det tydligare vad *Active Project* innefattade och det resulterade i skapandet av den texthierarki vilken beskrevs i tidigare avsnitt 5.1.1 Startsidan.

5.2.2 Mjödentusiasten

Ett användartest gjordes på en person med kunskap och erfarenhet inom såväl mjödbryggeri, som med andra bryggningprocesser. Testet gjordes precis innan en MVPA var uppfylld men designen var fulländad i detta skedde. Personen i fråga kunde lätt navigera sig på MeadLoggr och tyckte att det var ett lättanvänt användargränssnitt. Dock ansåg personen att själva funktionerna på MeadLoggr har ”startat i fel ände”, och menar att det intressanta med att brygga mjöd inte är att kunna beräkna volym, alkohol eller sötningsgrad, utan att ta fram ett mjöd, med en viss smak, för ett visst tillfälle. Mjödentusiasten menar på att en anledning till detta beror på att oavsett hur mängden honung beräknas så kommer honungen alltid vara det som påverkar smaken på ett oförutsägbart sätt då den kan vara framtagen på olika sätt, på olika tider av året. Att klassa ett mjöd som ett ”höstmjöd” eller en ”sällskapsdryck” skulle enligt personen vara ett bättre alternativ för mjödbryggare för att beskriva mjödets karaktär.

5.2.3 Ölbryggaren

I det sista användartestet deltog en person med smeknamnet "ölbryggaren", som hade erfarenhet inom ölbryggning och viss insikt om mjödprocessen. Generellt tyckte ölbryggaren att sidan *Create MeadLog* var intuitiv och effektivt presenterade information i kronologisk ordning. Dock, vid första intrycket kände sig användaren något överväldigad och hade svårigheter med att uppfylla kriterie 2, att kunna känna igen och använda webbapplikationens funktioner baserat på tidigare försök och enkelt kunna utföra samma uppgifter trots den omfattande mängden information och inmatningsfält. Detta ledde till en upplevelse av kognitiv överbelastning på *Create MeadLog*-sidan. Utöver detta var feedbacken inriktad mot mjödprocessen och hur tid loggas i programmet.

6. Diskussion och Slutsats

6.1 Diskussion

Syftet och målet med MeadLoggr var att ta fram en webb-applikation som kan förbättra bryggeriprocessen för en individ som vill brygga mjöd och erbjuda en innovativ lösning för att dokumentera och organisera sina bryggeriprojekt. Detta uppnåddes, med hjälp av de definierade delmålen. MeadLoggr är interaktivt och uppbyggt med principer från UX-design samt gestaltlagar för att säkerställa att den är användarvänlig, lätthanterad och lättnavigerad. Bidragande orsaker till att målen uppnåddes anses ligga i en bra grundprototyp i Figma och kontinuerliga design-möten med Consat. Dessutom utvärderades gränssnittet regelbundet under utvecklingsfasen.

Däremot för att undvika den kognitiva överbelastningen ölbryggaren nämnde under användartestet anses ett bättre alternativ vara att introducera informationen i *Create MeadLog* stegvis i en så kallad ”setup wizard”. En sådan lösning diskuterades i samråd med Consat och hade inneburit att man kan gå fram och tillbaka mellan steg och minska mängden information som presenteras samtidigt. Ett exempel hade varit att i steg 1 utföra allmän projektinsamling, vilket inkluderar projektnamn och starttid, för att sedan i nästkommande steg behandla mjödspecifka parametrar fram tills man nås av sammanfattningen.

Slutligen betraktas de tre användartesterna som otillräckliga för att med säkerhet besvara den ställda frågeställningen, hur man kan skapa ett lättnavigerat gränssnitt med designlagar. Om projektet hade tilldelats mer tid skulle det ha varit möjligt att involvera en bredare målgrupp inom mjödtillverkning för att samla mer omfattande data och därigenom möjliggöra en noggrannare och mer omfattande analys av frågeställningen. Däremot anses de tre testerna givande för projektets framgång och testpersonerna i fråga opartiska då endast den externa handledaren var känd sedan tidigare. Gällande kriterierna för ett giltigt gränssnitt så bidrog användartesterna med en del insikt i hur MeadLoggr var uppbyggt. Av de fyra punkter som togs fram i början visade det sig att samtliga punkter blev uppfyllda vid varje test, men där kriterie 2, ” Att användaren känner igen och har lärt sig webapplikationens funktioner utifrån sina första försök”, i första hand var svårare att uppfylla. I två fall klarade användarna att utföra uppgifter efter andra försöket själva, medan det tog tre gånger för ölbryggaren. En viktig faktor att ha i åtanke är att MeadLoggr, under de tillfällen då användartesterna utfördes, fortfarande var under konstruktion. Feedbacken användes för att förtydliga och omstrukturera delar av applikationen, vilket ledde till en mer användarvänlig design baserat på återkopplingen. En av de förbättringar som gjordes var att flytta och tydligare gruppera den information som visas på sidan *Create MeadLog* vilket kunde åtgärdas för att förvirring inte skulle uppstå i den färdiga produkten.

6.2 Avgränsningar och Förbättringsområden

Som nämnt i metodkapitlet utfördes användartester på personer både med och utan kunskaper inom mjödbryggning. Huvudfokuset låg hos den externa handledaren, som gav kritik på vad som funkade bra, respektive vad som funkade dåligt kontinuerligt under hela utvecklingsfasen. Exempelvis så fanns det synpunkter på komponenten event, där en användare kan lägga bland annat mätvärden och tillsättningar till mjödet, då han ansåg att komponenten huvudsakligen ska finnas med i steget där mjödet fermenteras, och inte innan det när musten tillverkas.

MeadLoggr befinner sig för närvarande i stadiet som tidigare benämnts MVPA, vilket står för ”minsta livskraftiga produkt” i utvecklingsfasen. Eftersom det inte fanns tillräckligt med tid för att slutföra webbapplikationen och inkludera både MVPA och MVPB, kunde inte MVPB genomföras. I stället klassificerades tankarna om hur MeadLoggr skulle vidareutvecklas som förbättringsområden och framtida planer. Bland dessa förbättringsområden finns en del av programmet som var tänkt att grafiskt representera ett projekt och alla mätvärden för bryggeriprocessen, vilket skulle möjliggöra studier av vad som ger bästa resultat med vilka parametrar. Vissa delar av MVPA, som inte ansågs vara kritiska för att nå ett "proof of concept", såsom att matcha de matematiska formlerna med liknande hemsidor för att uppnå så noggranna resultat som möjligt, uteslöts från utvecklingsprocessen.

Inspirationen till MeadLoggr kom från en liknande sida som heter MeadMakr. Gränssnittet på MeadMakr är inte så visuellt tillfredsställande vilket gjorde att utvecklandet av MeadLoggr aktivt tog hänsyn till de tidigare beskrivna gestaltlagarna för att bygga på det intuitiva gränssnittet. För att ytterligare optimera och utveckla MeadLoggrs intuitiva gränssnitt hade en heatmap används som analysverktyg om användartesterna utförts på nytt. En heatmap är en grafisk representation som visar var användare interagerar mest med ett digitalt gränssnitt. Genom att använda färgskalor visualiserar den var användare klickar, hovrar eller spenderar mest tid, vilket ger värdefull information om användarbeteenden och interaktionsmönster.

Ett annat förbättringsområde är att ta fram mer exakta beräkningar för kalkylatorn som tar fram måtten av alla ingredienser i mjödet, då MeadLoggr just nu bygger på MeadMakrs öppna källkod. Även om samma formler och beräkningar används som på MeadMakr blir resultatet inte detsamma, vilket kan bero på att det finns delar av källkoden som inte visas eller att beräkningarna tar andra värden i åtanke som inte är uppenbara. Den största skillnaden mellan applikationerna är att MeadLoggrs resultat på exempelvis beräknad mängd honung skiljer sig avsevärt från motsvarande kalkylator. Då det finns många olika stilar på mjöd och då mycket är beroende på sockerhalten i honungen kan formlerna se olika ut, och vid testning av applikationen tillfrågades mjödentusiasten om han hade kunskap om dessa formler. Han svarade att det går åt 8L honung för att brygga en sats på totalt 20L, vilket motsvarar två femtedelar. I beräkningarna i kalkylatorn används en konstant på ca. 0,412 vilket stämmer överens med mjödentusiastens beräkning. MeadMakr tar med största sannolikhet även andra konstanter med i beräkningarna.

MeadLoggr är en webbapplikation som utvecklades för att köras lokalt på en enhet med tillgång till källkoden från GitHub och en installerad PostgreSQL terminal för att skapa den lokala databasen. Under planeringsfasen diskuterades möjligheten att ladda upp webbapplikationen på en webbserver för att göra den tillgänglig för allmänheten. Tyvärr fanns det inte tillräckligt med tid för att genomföra detta innan implementationsfasen var över. Framtida planer innefattar att köra MeadLoggr på en webbserver. En möjlig lösning, baserat på tidigare erfarenheter från andra kurser, är att använda plattformstjänsten Docker för att packa MeadLoggr i en container och sedan ladda upp den på en molntjänst som till exempel Amazon Web Services (AWS). Amazons webservice stödjer också hosting för databasen. Detta skulle möjliggöra att MeadLoggr kan köras på en webbserver och vara tillgänglig för allmänheten. Till skillnad från vad som var planerat så spenderades inte alls lika mycket tid på testning av kod som planerat innan start. Planen var att genomgående skriva tester på koden upp till en total på ca. 71 arbetsdagar men då implementationen, utvecklingen och problemlösning tog mycket tid

så ansågs testerna som en avgränsning. Det finns test i front-end för log-in och registreringen vilket testar att gränssnittet renderas korrekt samt att förväntade anrop skickas till back-end vid användarinteraktion. Motsvarande tester skrevs i back-end för att testa insättningar till databasen och att hashningen fungerar korrekt och kastar fel om ett lösenord inte matchar med det sparade hashvärdet. Fler Jest-tester än så implementerades inte på grund av tidsbrist då det ansågs viktigare att uppnå en MVPA.

GitHub är ett kraftigt verktyg med många bra funktioner, men stundvis kunde arbetsprocessen i GitHub leda till konflikter när utvecklingen och ändringar sker i samma fil eller sökväg, så kallade merge-konflikter. Oftast berodde de på misskommunikation och bristande kunskap i hur de mer avancerade funktionerna med till exempel ”branching” i GitHub fungerar. Vid stora ändringar från båda håll kunde en merge-konflikt innebära att flera timmar fick gå åt att manuellt klippa och klistra kod från de olika versionerna för att komma fram till den lösning som egentligen skulle finnas uppladdad.

6.3 Hållbarhet och Etik

Hållbarhet och etik är två viktiga aspekter att beakta även i projekt som MeadLoggr, där huvudfokus ligger på mjödbryggning och användarupplevelse. Genom att ha hållbara och etiska principer i åtanke genom hela projektet kan MeadLoggr bidra till att främja ansvarsfullt agerande och en mer hållbar framtid.

Det finns både etisk försvarbara och oförsvarbara aspekter med att skapa en webbsida för mjödbryggeri. Den främsta är det oförsvarbara, att främja en kultur av att brygga alkohol, vilket kan leda till konsekvenser såsom alkoholism. Dock är det försvarbart då det är ett ekologiskt och närproducerat alternativ, då det inte tillverkas av stora företag som tjänar pengar på att sälja egen alkohol.

När det kommer till hållbarhet, var en central del av MeadLoggr att främja hållbara metoder inom mjödbryggning. Genom att tillhandahålla en plattform för loggning av bryggeriprojekt, möjliggjorde MeadLoggr för användarna att vara mer medvetna om exakta mått på ingredienser och råvaror, och därmed minimera onödigt avfall och resursförbrukning. Dessutom kunde användarna genom MeadLoggr ha en bättre överblick över sina ingredienser och förpackningar, vilket underlättade för dem att välja mer miljövänliga alternativ och stödja lokala och ekologiska producenter.

En viktig aspekt av etik för MeadLoggr är att främja inkludering och mångfald. Genom att erbjuda en plattform som är tillgänglig för alla, oavsett funktionsförmåga eller kunskapsnivå, kan MeadLoggr bidra till en mer inkluderande mjödbryggningsgemenskap. Det är viktigt att se till att användargränssnittet och funktionerna är utformade med tillgänglighetsstandarder i åtanke för att skapa en inkluderande och varierad användarupplevelse.

För att säkerställa hållbarhet och etik är det också viktigt för MeadLoggr att kommunicera ärligt och transparent med användarna. Genom att tillhandahålla tydlig och korrekt information om produktens funktioner, förväntningar och eventuella begränsningar kan MeadLoggr bygga förtroende hos sina användare och främja en etisk användarrelation.

MeadLoggr källkod är just nu inte öppen för allmänheten, utan ligger i ett privat GitHub repository. Dock, Om MeadLoggrs källkod hade gjorts öppen, så skulle det möjliggöra att

vem som helst kan använda och modifiera produkten utan kostnad, vilket kan leda till innovation och bättre lösningar inom andra områden som ex. ölbryggeri i framtiden. Det finns dock etiska frågor kring begränsningar och missbruk av koden. För att hantera detta kan licensvillkor och tydliga riktlinjer vara nödvändiga.

6.4 Slutsats

Med hjälp av principer för användarvänliga gränssnitt och flera användartester har MeadLoggr utvecklats för att skapa ett intuitivt gränssnitt. Även om användbarhet inte är direkt mätbart visade resultaten från användartesterna att applikationen är lättanvänd och lättnavigerad, utifrån de kriterier som togs fram för att testa hur användbart gränssnittet för MeadLoggr är. För att stärka detta påstående ytterligare behövs dock fler användartester. För närvarande uppfyller MeadLoggr sin funktion om att underlätta loggningen av mjödbryggeri och mjödprojekt, och med hjälp av gestaltlagar inom webbdesign och principer för ett intuitivt gränssnitt så kan det säkerställas att MeadLoggr är ett lyckat projekt som fullföljde sitt syfte och mål.

7 Källhänvisning

[1] Pereira, A., Oliveira, J., Mendes-Ferreira, A., Estevinho, L., & Mendes-Faia, A. (2017). Mead and Other Fermented Beverages. Hämtad från https://www.researchgate.net/publication/311997473_Mead_and_Other_Fermented_Beverages

[2] Jones, A. (Utgivningsår saknas). The Basic Mead Recipe. MeadMakr. Hämtad från http://www.meadmakr.com/meadmakr-guide/part-iii-the-basic-recipe/

[3] Mead Made Right. (2015). NUTRIENT ADDITIONS. Hämtad från https://www.meadmaderight.com/nutrient-additions

[4] Castelyn, S., Daniel, F., Dolog, P., & Matera, M. (2009). Engineering Web Applications (1. uppl.). Springer. Hämtad från https://link-springer-com.proxy.lib.chalmers.se/book/10.1007/978-3-540-92201-8

[5] Subramanian, V. (2019). Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node (2. uppl.). Apress. Hämtad från https://link-springer-com.proxy.lib.chalmers.se/book/10.1007/978-1-4842-2653-7

[6] Zametti, F. (2022). Modern Full-Stack Development: Using TypeScript, React, Node.js, Webpack, Python, Django, and Docker (2. uppl.). Apress. Hämtad från https://eds.p.ebscohost.com/eds/detail/detail?vid=3&sid=401a4deb-2754-47cd-a983-29cc4023b9dd%40redis&bdata=JnNpdGU9ZWRzLWxpdmUmc2NvcGU9c2l0ZQ%3d%3d#AN=3456353&db=edsebk

[7] Fenton, S. (2017). Pro TypeScript: Application-Scale JavaScript Development (2. uppl.). Apress. Hämtad från https://link-springer-com.proxy.lib.chalmers.se/book/10.1007/978-1-4842-3249-1

[8] Salahaldin, J., Vannahme, A., & Volkov, A. (2015). Learding PostgreSQL: Create, Develop and Manage Relational Databases in Real World Applications Using PostgreSQL (1. uppl.). Packt Publishing ltd. Hämtad från https://eds.s.ebscohost.com/eds/detail/detail?vid=11&sid=3bc7a9e3-10b6-456b-9b1e-ef04c51b9c31%40redis&bdata=JnNpdGU9ZWRzLWxpdmUmc2NvcGU9c2l0ZQ%3d%3d#AN=clec.EBC4191180&db=cat07472a

[9] Hossain, M. (2015). CORS in Action: Creating and Consuming Cross-origin APIs (1. uppl.). Manning Publications. Hämtad från https://eds.s.ebscohost.com/eds/detail/detail?vid=11&sid=6e1bd377-d842-4dd0-b055-9d921fc233fc%40redis&bdata=JnNpdGU9ZWRzLWxpdmUmc2NvcGU9c2l0ZQ%3d%3d#AN=2948911&db=edsebk

[10] The Axios Project. (2020). Getting Started. Hämtad från https://axios-http.com/docs/intro

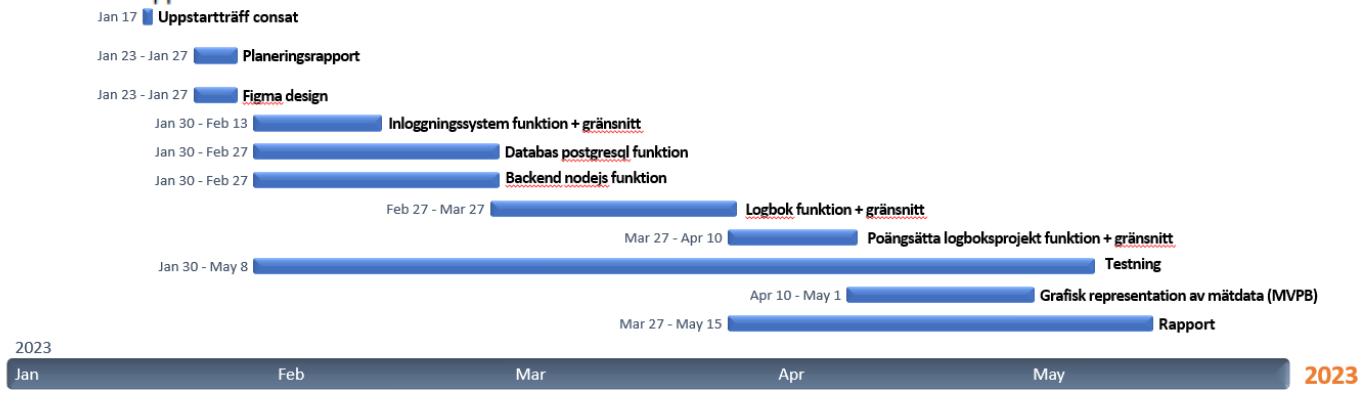
[11] Graham, L. (2008). Gestalt Theory in Interactive Media Design (1.uppl.). Journal of Humanities & Social Sciences. Hämtad från http://www.guillaumegronier.com/2021-psychologiegenerale/resources/Graham2008.pdf

[12] Tomlin, W. C. (2018). UX Optimization: Combining Behavioral UX and Usability Testing Data to Optimize Websites. (1. uppl.). Cedar Park, Texas, USA: Apress Berkeley, CA. Hämtad från https://eds.p.ebscohost.com/eds/detail/detail?vid=1&sid=485e0bb5-540a-454b-a013-f035eaa1feba%40redis&bdata=JnNpdGU9ZWRzLWxpdmUmc2NvcGU9c2l0ZQ%3d%3d#AN=clec.SPRINGERLINK9781484238677&db=cat07472a

8 Bilagor
8.1 gantt schema

			Totalt
Startdatum:	2023-01-01	**Arbetsdagar**	219
Start projektvecka:	1	**Arbetstimmar**	238

Aktivitet	**Startdat.**	**Slutdat.**	**Dagar** mån-fre
Uppstartträff consat	2023-01-17	2023-01-17	1
Planeringsrapport	2023-01-23	2023-01-27	5
Figma Design	2023-01-23	2023-01-27	5
Inloggningssystem funktion + gränsnitt	2023-01-30	2023-02-13	11
Databas postgresql funktion	2023-01-30	2023-02-27	21
Backend node funktion	2023-01-30	2023-02-27	21
Logbok funktion + gränsnitt	2023-02-27	2023-03-27	21
Poängsätta logboksprojekt funktion + gränssnitt	2023-03-27	2023-04-10	11
Testning	2023-01-30	2023-05-08	71
Grafisk representation av mätdata (MVPB)	2023-04-10	2023-05-01	16
Rapport	2023-03-27	2023-05-15	36



INSTITUTIONEN FÖR DATA -OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2023
www.chalmers.se

GÖTEBORGS
UNIVERSITET

CHALMERS