

Projekt start: 21. oktober 2024

Projekt slut: 31. oktober 2024

Cupcake projekt - DAT2F24



Gruppe B

Navn	E-mail	Github
Mikkel Dam Binau	cph-mb743@cphbusiness.dk	https://github.com/Neoblueorred
Mathias Kroghave Falcham	cph-mf398@cphbusiness.dk	https://github.com/Falcho
Frederik Micheal Franck	cph-ff72@cphbusiness.dk	https://github.com/MojoOno
Ingrid Karen Svendsen	cph-is142@cphbusiness.dk	https://github.com/ingridsv

Video link:

<https://cphbusiness.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=60771b51-15b9-455c-9f08-b21900d7f17f&start=0>

Indholdsfortegnelse

Indledning.....	2
Baggrund.....	2
Teknologi valg.....	3
Krav	4
Aktivitetsdiagram.....	5
ER Diagram.....	6
Domænemodel.....	7
Navigationsdiagram.....	8
Særlige forhold.....	9
Status på implementation.....	10
Proces.....	11
Bilag:.....	13

Indledning

I denne rapport, hvor målgruppen er en datamatiker studerende på 2. semester, dokumenteres udviklingen af en hjemmeside der skal leveres til Olsker Cupcakes. Formålet med hjemmesiden er at kunden skal kunne vælge en topping, en bund og et antal af de valgte kombinationer, hvorefter kunden skal kunne lægge den i sin kurv og bestille for de forskellige kombinationer af toppings og bunde kunden har valgt.

Baggrund

Olsker Cupcakes, som er et lille økologisk bageri, har kastet sig ud i at starte virksomhed på Bornholm. De specialiserer sig i cupcakes og har i den forbindelse brug for hjælp til at kunne facilitere disse gennem en hjemmeside.

Kunden har bedt om en løsning som kan afhjælpe dem med følgende:

Det skal være muligt at oprette sig som bruger på deres hjemmeside, for derfor at kunne bestille en ordre af cupcakes som man kan afhente og betale i butikken. Ydermere ønsker kunden et system hvor administratoren kan sætte penge ind på kundernes konto, for derefter at kunne betale for sine ordrer på hjemmesiden.

Som kunde skal man kunne vælge mellem en bund, top og antal cupcakes, for så at kunne lægge dem i en kurv. Fra kurven kan man bestille sine ordrer til afhentning. Har man penge på kontoen, betaler man når man bestiller.

Alt dette skal gemmes i en database, så administratoren kan danne sig et overblik over ordrene i virksomheden.

Teknologi valg

Vi har valgt, at benytte os af de teknologier, som vi er blevet præsenteret i løbet af datamatikeruddannelsen.

IntelliJ Ultimate 2024.2.3

Javalin (version 6.3.0)

Thymeleaf (version 3.1.2)

Docker (version 4.28.0)

PgAdmin4 (version 8.3)

PostgreSQL (version 42.7.2)

Jackson (version 2.17.0-rc1)

HikariCP (version 5.1.0)

Maven (version 3.10.1)

GitHub

GitHub Projects

Krav

Firmaets forventning er, at systemet skal forenkle deres bestillingsproces og gøre det muligt for kunder at tilpasse deres bestillinger online. Deres håb er at kunne øge salget ved at tilbyde en mere fleksibel og brugervenlig platform for kunderne.

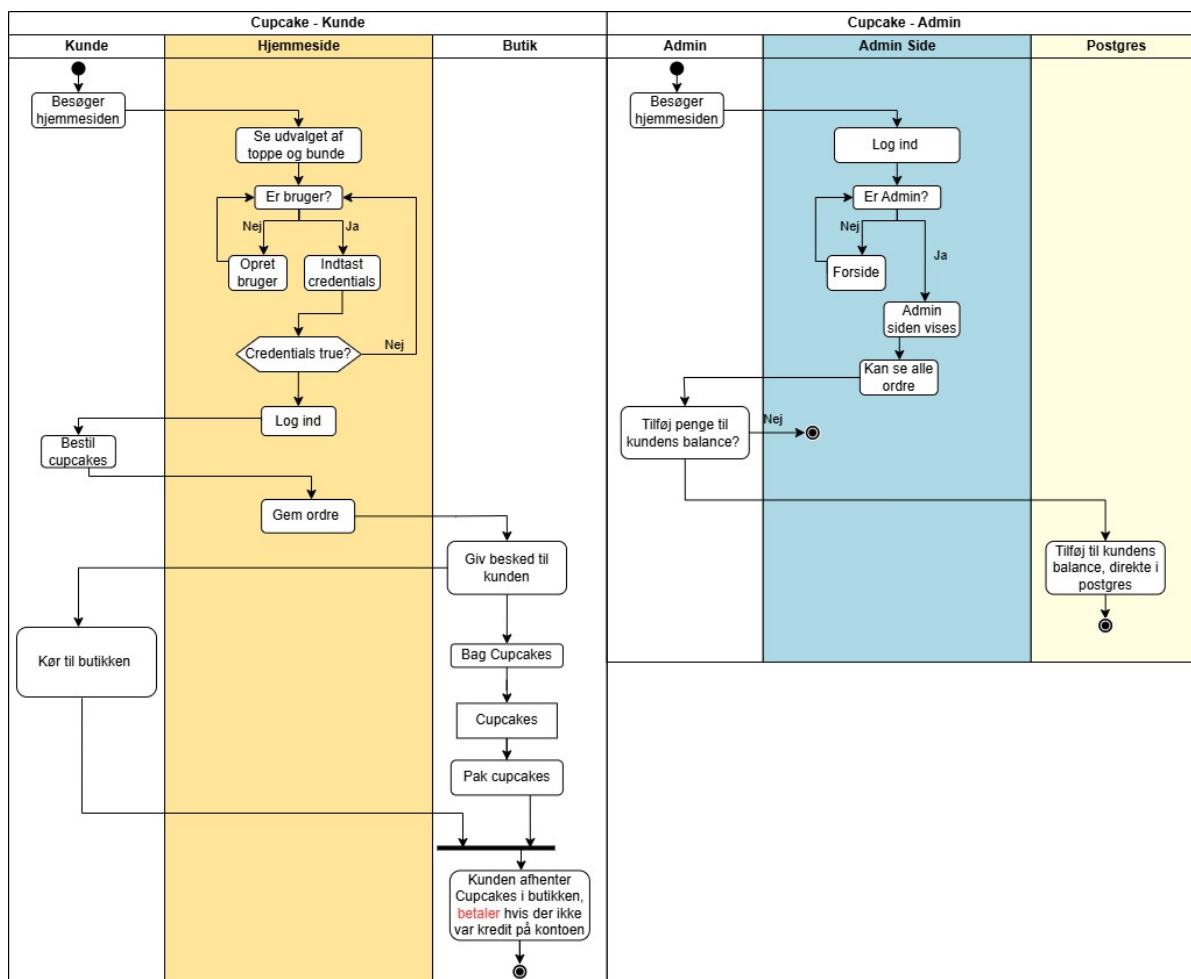
US-1:	Som kunde kan jeg bestille og betale cupcakes med en valgfri bund og top, så jeg senere kan køre forbi butikken i Olsker og hente min ordre.
US-2	Som kunde kan jeg oprette en konto/profil for at kunne betale og gemme en ordre.
US-3:	Som administrator kan jeg indsætte beløb på en kundes konto direkte i Postgres, så en kunde kan betale for sine ordrer.
US-4:	Som kunde kan jeg se mine valgte ordrelinjer i en indkøbskurv, så jeg kan se den samlede pris.
US-5:	Som kunde eller administrator kan jeg logge på systemet med e-mail og kodeord. Når jeg er logget på, skal jeg kunne se min email på hver side (evt. i topmenuen, som vist på mockup'en).
US-6:	Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.
US-7:	Som administrator kan jeg se alle kunder i systemet og deres ordrer, så jeg kan følge op på ordrer og holde styr på mine kunder.
US-8:	Som kunde kan jeg fjerne en ordrelinje fra min indkøbskurv, så jeg kan justere min ordre.
US-9:	Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde ugyldige ordrer. F.eks. hvis kunden aldrig har betalt.

Aktivitetsdiagram

Aktivitets Diagrammerne er også vedhæftet separat under bilag for bedre overskuelighed.

Kunden starter med at besøge hjemmesiden, hvor udvalget af toppe og bunde samt priserne hertil vises. Hvis kunden ikke allerede har en konto, skal denne oprette en, de eksisterende brugere kan logge ind med deres e-mailadresse og adgangskode. Når loginoplysningerne er verificeret, kan kunden sammensætte toppe og bunde og lave en bestilling fra kurven, ordren bliver sendt til databasen. Butikken giver besked til kunden, hvornår ordren kan afhentes og begynder at bage samt pakke kundens ordre. Kunden tager til butikken for at afhente ordren, og betaler hvis der ikke var kredit på kontoen.

Admin starter også på hjemmesidens forside og logger ind som administrator. Når admin er logget ind, har systemet tjekket, at det rent faktisk er en admin. Her kan admin se alle ordrer, som der er blevet lagt, og se om de er blevet betalt ved bestillingen. Hvis kunden vil indsætte penge på deres konto, kan admin gøre dette direkte i databasen, hvorefter beløbet er tilgængeligt med det samme.



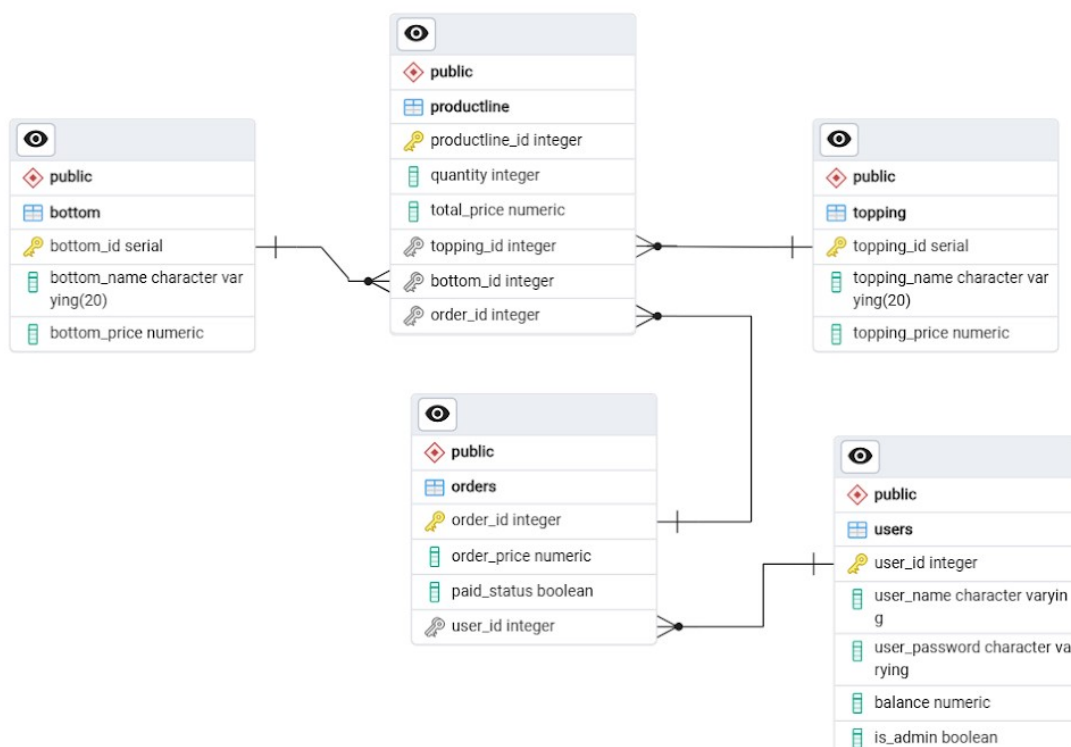
ER Diagram

ER-diagrammet giver et overblik over hele databasens struktur og indeholder alle de nødvendige tabeller og relationer for at understøtte cupcake-bestillingssystemets funktionalitet. Diagrammet, som man kan se på billedet, inkluderer nøgle-tabeller såsom users, orders, productline og bottom samt de relationer, der binder disse tabeller sammen.

Tabellerne er designet til at være i 3. normalform (3NF), hvilket betyder, at hver kolonne i en tabel afhænger fuldstændigt af tabellens primærnøgle og ikke af andre kolonner. Dette sikrer større datasikkerhed og reducerer unødvendig data dublering.

Som det ses på ER-diagrammet, bliver 1-til-mange-relationen gennem tabellen productline benyttet. Denne tabel forbinder orders med kombinationen af bottom og topping ved at tillade flere kombinationer af bunde og toppings i en enkelt ordre. Dette gør det således muligt at tilføje flere varianter af cupcakes i samme ordre uden at duplikere data i orders. Productline fungerer derfor som en mellemledsstabel, der sikrer fleksibilitet i databasen.

Som man også kan bemærke i ER-diagrammet, bliver der kun benyttet 1-til-mange-relationer. Dette skyldes, at vi ønsker at opnå en enkel og letforståelig datastruktur. Samtidig er der heller ikke flere veje til at nå samme tabel. Dette hjælper med at holde databasen konsistent, da hver tabel kun kan nås på én måde. Ved at have enkle og entydige forbindelser mellem tabellerne minimerer vi risikoen for inkonsistente data.



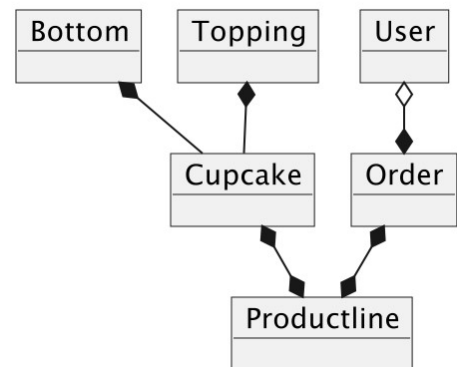
Domænemodel

Diagrammet nedenfor viser de centrale klasser i systemet. Modellen indeholder følgende klasser: User, Order, ProductLine Cupcake, Bottom og Topping.

Som ses på domænemodellen er næsten alle relationer “must-have” med undtagelse af “User” relation til “Order”.

Dette er fordi at man godt kan have en bruger uden at have en ordrer tilknyttet. Dette er derfor en “can-have” relation, da User kan eksistere uden Order.

I starten af projektet tænkte vi at det kunne være smart at lave en klasse hvor vi samlede Bottom og Topping for at holde det overskueligt og struktureret.

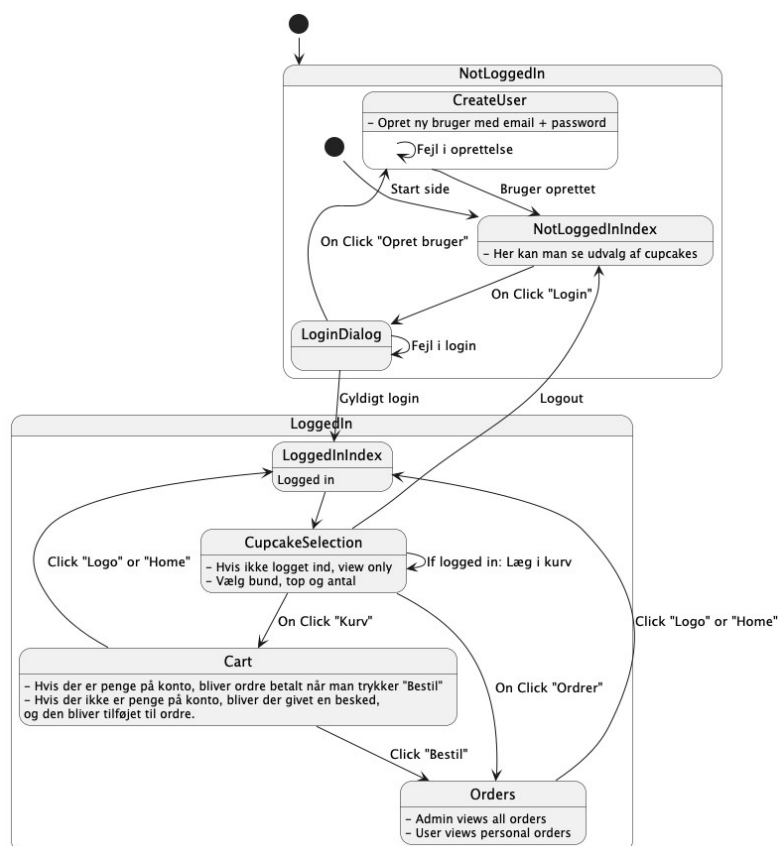


Navigationsdiagram

Nedenfor ses et navigationsdiagram over vores program. Flowet i programmet er delt op i to states; "Not logged in" eller "logged in".

Vi starter med at starte programmet/websitet op, hvor vi ryger ind i vores "not logged in" state. Her er der en index, som er forsiden, som kan navigere os rundt.

Når vi logger ind, kommer vi ind i vores "logged in" state. Herfra er der en index som vi kan navigere rundt fra.



"Lavet med PlantUML i IntelliJ"

Vi har i programmet valgt at have en fælles navigationslinje som har forskellige links alt efter om man er logget ind som admin, eller som bruger. Dette har vi valgt at gøre da vores design er lavet sådan, at en "admin" er en "bruger", dog med flere beføjelser. De lever i samme univers, så at sige. Navigationslinjen er derfor den samme, men hvilke knapper man kan se varierer. Ydermere er indholdet på diverse sider også forskellige, hvilket også kan ses i vores "logged in" state under "Orders". Her kan en "admin" se alle ordrer, men en "bruger" kan kun se egne ordrer.

Særlige forhold

Session data

I dette projekt gemmes visse oplysninger i sessionen for at sikre en effektiv brugeroplevelse. Brugernavn gemmes for at identificere den aktuelle bruger, og brugerens rolle lagres for at skelne mellem almindelige brugere og administratorer. Derudover opbevares indkøbskurven i sessionen for at holde styr på de cupcakes, som brugeren har tilføjet.

Exceptions håndtering

For at håndtere exceptions anvender vi en kombination af try-catch blokke til at fange og logge fejl og brugerdefinerede fejlmeddelelser, som vises til brugeren via Thymeleaf-skabeloner. Vi har implementeret en exception handler, `DatabaseException`, som vi bruger til at sikre stabilitet ved uforudsete fejl.

Validering af brugerinput

Brugerinput valideres og håndteres for det meste i controller-klasserne. Grundlæggende validering sker i controlleren, hvor try-catch blokke anvendes til at håndtere fejl. Opstår der fejl, vises en fejlmeddelelse til brugeren via Thymeleaf for at sikre brugervenlig feedback. Ved oprettelse af bruger sikres valideringen af korrekt input i HTML-en via et input tag. Der valideres ikke, om det er en korrekt email adresse.

Sikkerhed

Hashing af adgangskoder er ikke implementeret i det nuværende projekt, så adgangskoder gemmes i klar tekst i databasen. Ved login sammenlignes den indtastede adgangskode direkte med den gemte. Efter et vellykket login gemmes brugeren i sessionen for at sikre, at sessionen kan fortsætte, så længe brugeren er logget ind.

Database

I databasen er der to definerede brugertyper: almindelige brugere og administratorer. Almindelige brugere kan oprette og se deres egne ordrer, mens administratorer kan se alle ordrer og administrere brugere. Brugertyperne er implementeret i JDBC med roller og adgangskontrol i SQL-forespørgslerne for at sikre differentieret adgang.

Andre elementer

Andre væsentlige elementer i projektet inkluderer brugen af Thymeleaf som skabelonmotor, hvilket letter implementeringen af dynamiske HTML-sider. Dataforbindelsen optimeres med HikariCP som connection pool, og SLF4J bruges til logging af applikationshændelser, hvilket bidrager til effektiv fejlfinding og overvågning.

Status på implementation

Funktionalitet

I vores program har vi nu implementeret user story (US) 1 til 6, hvilket betyder, at vi har et funktionelt program, der opfylder de grundlæggende krav. Status på US 7 kan diskuteres, da ordresiden viser, hvilken kunde der har afgivet ordren, men uden at ordrene er sorteret efter kunder. Vi har også klargjort fundamentet for, at arbejdet med US 8 og 9 kan begynde.

Styling

Der er stadig nogle styling justeringer, vi skal have på plads. Enkelte elementer afviger fra vores mock-ups, og der er udfordringer med at få knapperne på navigationslinjen placeret horisontalt med hinanden. Derudover mangler vi at implementere et kurveikon på en af siderne. Vi har arbejdet på at gøre programmet responsivt, så det vises korrekt på både laptop og mobil, men det er endnu ikke testet på mobil.

Mangler

Efter vores codefreeze har vi opdaget, at det er muligt at oprette to brugere med samme brugernavn/email. Dette kan vi løse ved at sætte en unique constraint i databasen. Vi har også fundet inkonsistenser af sproget i programmet, som vi skal have strømlinet til kun at være dansk. For eksempel er nogle knapper og beskeder til brugerne stadig på engelsk.

Vores klassediagram stemmer ikke overens med de metoder, vi har brugt, da vi under arbejdet opdagede, at nogle af de metoder, vi troede, vi skulle bruge, viste sig at være overflødige, mens andre metoder, vi ikke havde forudset, blev nødvendige.

Disse mangler kan implementeres i en af de næste versioner af programmet.

Proces

Før vi begyndte

Målet for projektet var ikke at levere et færdigt produkt, men at fokusere på processen og samarbejdet for at lære og forbedre os. Selvom vi var bevidste om dette, havde vi alligevel høje – og måske også for høje – forventninger til os selv om, hvad der skulle afleveres.

Allerede inden projektet startede, lavede vi en forventningsafstemning omkring, hvordan vi skulle arbejde, hvor vi skulle mødes, og hvordan vi skulle håndtere situationer, hvis nogen ikke kunne opfylde de krav, vi havde sat til hinanden. Derudover lavede vi en overordnet tidsplan for projektet. Vi planlagde både tid til forberedelse, selve kodningen, codefreeze, arbejdet med rapporten, og til sidst den afsluttende forberedelse til fremlæggelsen.

Forberedelsen

Under forberedelsen blev vi enige om vores git-protokoller, code of conduct, acceptance criteria og definition of done. Vi udarbejdede også diverse diagrammer, såsom klassediagrammet, for at sikre, at vi alle havde en fælles forståelse af, hvordan programmet skulle fungere.

Kodningsprocessen

Kodningen var delt op i sprints for hver user story, som blev yderligere specificeret med en kort beskrivelse af, hvordan de skulle løses, dog uden pseudo-kode. Vi vurderede, hvilke user stories der krævede mest arbejde og tid, og vi tildelte de første fire user stories, så hver af os arbejdede individuelt med én. Når man blev færdig, fortsatte man til den næste i rækken. Målet var at færdiggøre US 1-6, men vi havde ambitioner om at nå alle ni.

Codefreeze

Vi aftalte codefreeze til mandag kl. 16. På dette tidspunkt havde vi implementeret det, der beskrives i afsnittet 'Status på implementation'. Allerede fredag havde vi en fornemmelse af, at vi kun ville nå US 1-6, men valgte stadig at holde fri i weekenden.

Rapporten

Tirsdag formiddag gennemgik vi programmets funktioner samt hvad vi havde nået og ikke nået, så alle var opdaterede, og vi kunne bruge det i rapporten. Vi fordelte rapportens forskellige punkter og valgte hver især de sektioner, vi havde mest lyst til at arbejde med. Vi samledes og gennemgik indholdet, når alle var færdige, med fokus på kvalitet frem for kvantitet.

Det der gik galt



Selvom vi mente, at forberedelsen var grundig, fandt vi hurtigt ud af, at der manglede vigtige detaljer. Vi havde for eksempel ikke koordineret navngivningen af diverse elementer som attributter, lister, objekter og andre parametre. Dette kunne have gjort kodningen lettere og reduceret fejlsøgning, da metoderne i mapperne, controlleren og HTML kunne arbejde bedre sammen. Vi mistede hurtigt overblikket over koden og sammenhængen mellem metoderne. Vi vidste, hvordan programmet skulle fungere, men de sidste brikker viste sig svære at få på plads, hvilket skabte betydelige udfordringer og tog mere tid end forventet.

Det vi tager med os

“Hvor der handles, der spildes.” Denne talemåde beskriver måske bedst vores proces. Vi kastede os ud i projektet med det mål, der var sat for os, og vores egne ambitioner. Vi havde tillid til, at vores forberedelse var tilstrækkelig, men lærte, at mere tid på forarbejdet kan forhindre misforståelser. Desuden har vi lært vigtigheden af at have en fælles code of conduct og at følge den, da manglende ensretning kan skabe uventede problemer for de andre deltagere. Dette kunne have sparet os tid på fejlsøgning og reparationer, som i stedet kunne have været brugt på at implementere en eller to user stories mere.

Bilag:

Figma mockup:



[Login](#) [Opret bruger](#)

Velkommen ombord



Øens bedste cupcakes. Vælg og bestil her:

Vælg bund ▾


Vælg topping ▾

Obs. Du skal være logget ind for at bestille

Login



Bestil



Login

Brugernavn@mail.com

Password

Login

Opret bruger



Opret bruger:

Opret bruger



Log ud

jobe@cphbusiness.dk



Velkommen ombord

Øens bedste cupcakes. Vælg og bestil her:

Vælg bund



Vælg topping



Vælg antal



Læg i kurv



Ordrer

Bestil

jobe@cphbusiness.dk



Kurv:

Bestil



Ordrer

Bestil

jobe@cphbusiness.dk



Dine ordre:

Eksempel 1 cupcake ordre



Ordre

Kunder

admin@admin.dk

Alle ordre:

Eksempel 1 cupcake ordre

Ret Klar Slet

Eksempel 2 cupcake ordre

Ret Klar Slet

Eksempel 3 cupcake ordre

Ret Klar Slet



Ordre

Kunder

admin@admin.dk

Kunder:

Eksempel 1 Kunde

Ret

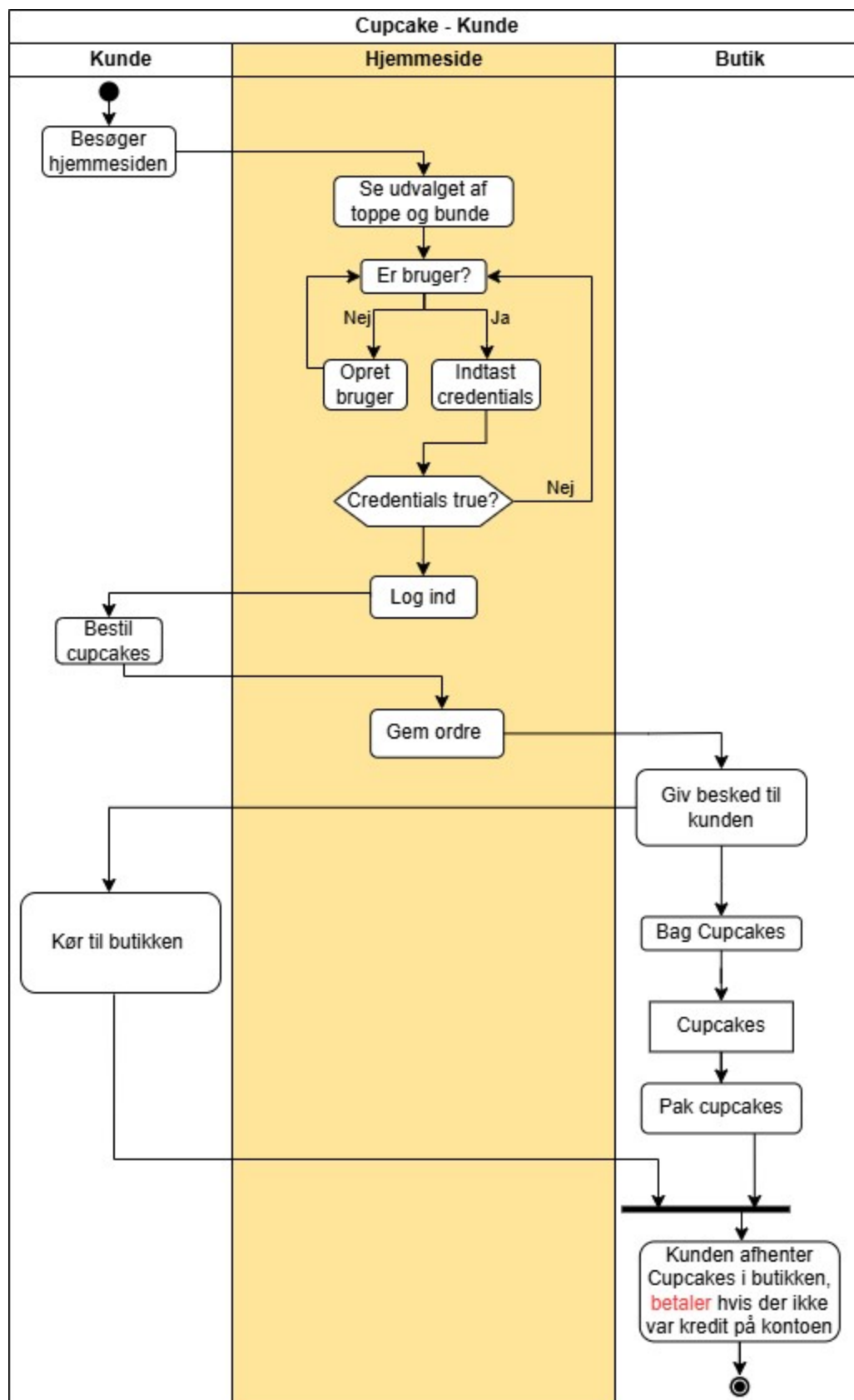
Eksempel 2 Kunde

Ret

Eksempel 3 Kunde

Ret

Aktivitetsdiagram - Kunde



Aktivitetsdiagram - Admin

