Cupcake

Deltagere:

Hold: A Gruppe: 7

Navn: Frederik Korsgaard Dupont, mail: cph-fd101@cphbusiness.dk, github: freddo-404

Navn: Marcus Nishan Ekeroth, mail: cph-me313@cphbusiness.dk, github: Marcus-Ekeroth

Navn: Frederik Per Moestrup, mail: cph-fm124@cphbusiness.dk, github: FrederikMoestrup

Navn:Jonas de Zoete Kongsted Nielsen, mail:cph-jn470@cphbusiness.dk, github:DeZoete

Dato for start af projekt: 02-04-24

Dato for start af rapport: 09-04-24

Demovideo: https://youtu.be/W5-zDLh8fus

Indholdsfortegnelse

Indledning:

Dette projekt omhandler en kundes ønske om en hjemmeside til at passe sammen med deres cupcake forretning, sådan at man som bruger er i stand til at bestille, før man så afhenter sin ordre i butikken. Denne rapport tager udgangspunkt i en læser som har et kvalifikationsniveau svarende til en datamatiker studerende på 2. semester.

I dette projekt er der anvendt følgende software:

- IntelliJ IDEA 2023.2.2 (Ultimate Edition)
- Thymeleaf 3.1.2
- Javalin 6.1.3
- Jetty 11.0.20
- PostgresQL 8.3
- Java 17

Dette projekt er dedikeret til kunden, Olsker Cupcakes, som er en cupcake forretning på Bornholm. Hjemmesidens formål er at bidrage til kundens forretning, ved at implementere følgende tilgængeligheder for brugeren:

- At kunne lave en bruger eller logge ind til en eksisterende bruger
- At kunne bestille med valgfri top og bund
- At kunne lave om på sin ordre
- At kunne betale for sin ordre

Krav:

Formålet for hjemmesiden er at have en positiv indflydelse på forretningen. Hjemmesiden tillader brugere at forudbestille deres ordre, sådan at de ikke behøver at bestille i butikken og kan hente den, når den er klar, frem for at skulle vente. Dermed er hjemmesiden med til at skabe en bedre oplevelse for brugeren, hvilket forhåbentligt bidrager til at øge forretningens salg af produkter.

Hjemmesidens primære krav er at opfylde følgende user stories:

US-1: Som kunde kan jeg bestille og betale cupcakes med en valgfri bund og top, sådan at 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 email og kodeord. Når jeg er logget på, skal jeg kunne se min email på hver side (evt. i topmenuen, som vist på mockup'en).

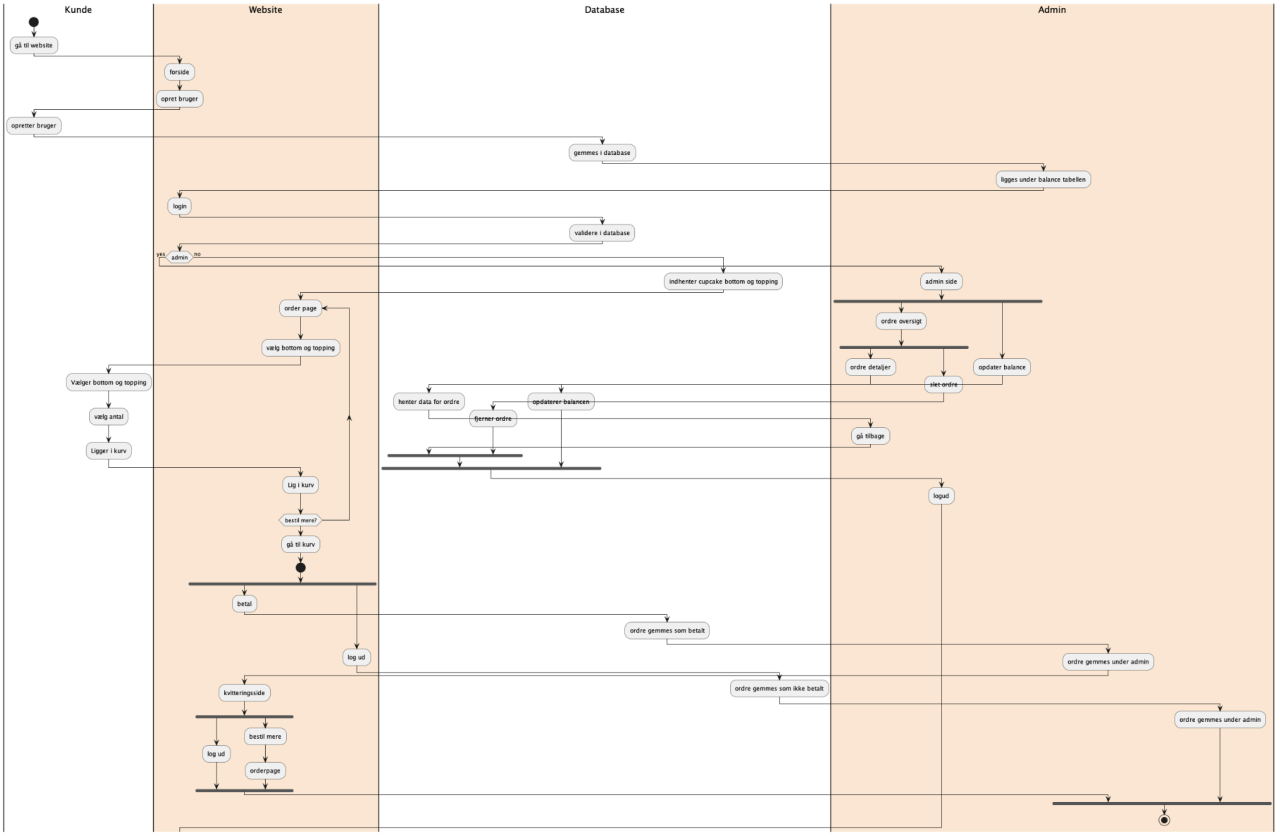
US-6: Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.

US-7: Som administrator kan jeg se alle kunder i systemet og deres ordrer, sådan at 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.

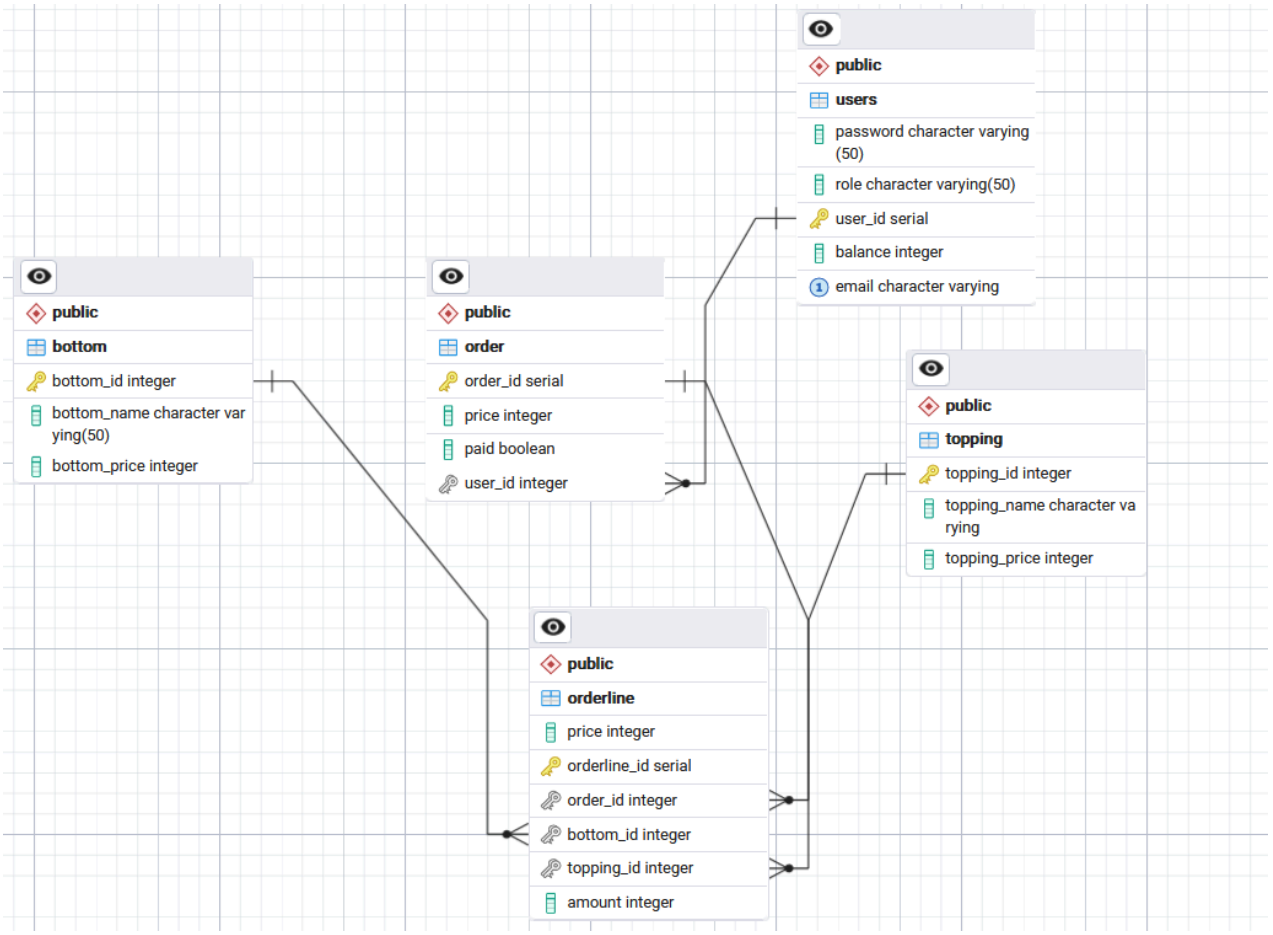
US-9: Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde ugyldige ordrer. F.eks. hvis kunden aldrig har betalt.

Aktivitetsdiagram:



Ovenstående diagram repræsenterer et aktivitetsdiagram, der anvendes som et værdifuldt redskab inden for softwareudvikling, forretningsprocesmodellering og systemanalyse. Aktivitetsdiagrammer er essentielle for at visualisere trinene eller aktiviteterne i en given proces. Dette diagram i særdeleshed beskriver processen fra det tidspunkt, hvor en bruger besøger hjemmesiden, til de forskellige ruter, der følges gennem systemet for at udføre specifikke handlinger, og identificerer ligeledes aktiviteterne i andre dele af systemet. Brugen af "svømmebaner" er blevet implementeret med det formål at skabe et mere struktureret og overskueligt overblik over systemet ved at segmentere processerne under forskellige sektioner af systemet.

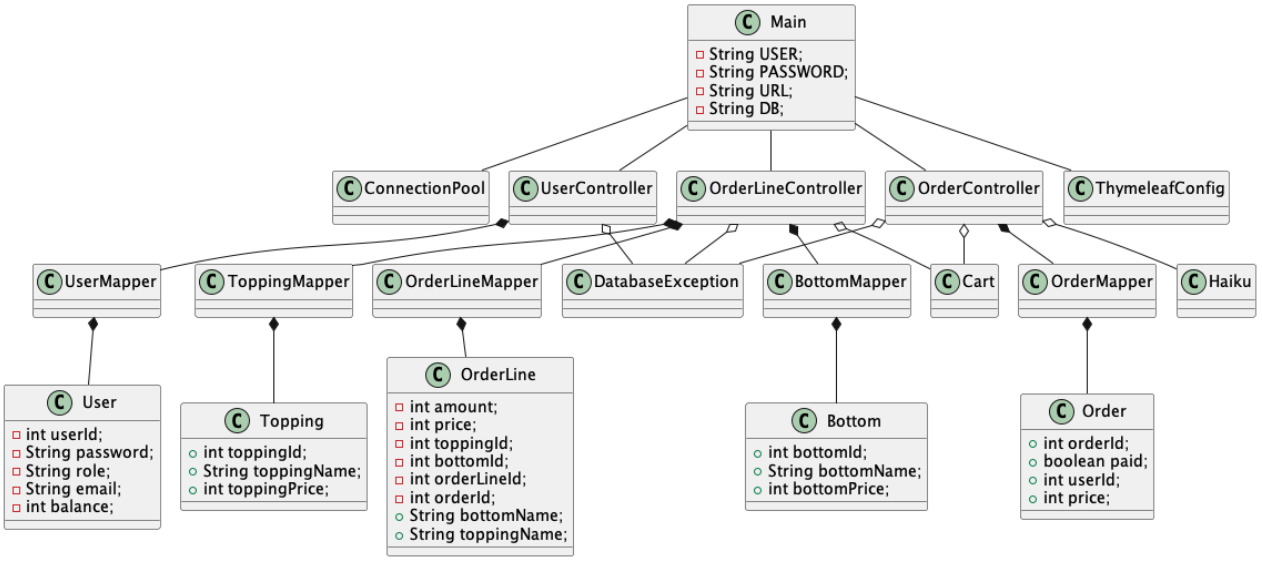
ERD:



Ovenstående model viser ER-diagrammet over databasen. ER-diagrammer bliver ofte brugt som et værktøj til at vise et overblik over hvordan de forskellige tabeller interagerer med hinanden. Dette kunne fx være hvilken type relation de har med hinanden, eller hvilke foreign keys der bliver brugt. I vores database har vi struktureret det, sådan at en order kan bestå af flere orderlines, hvor en orderline er en specifik cupcake, samt hvor mange af dem der er. Her er også anvendt en foreign key, så hver orderline tilhører en order igennem order id. Order indeholder også en foreign key fra user, så man kan se hvilken user der har lavet hvilken order. De sidste foreign keys er et topping id og et bottom id under orderline, hvilket fortæller hvilken topping og bottom useren har valgt til den specifikke orderline. I forhold til relationer i ER-diagrammet er der kun brugt en-til-mange relationer, hvilket ofte kan være en god ting, da det kan være svært at arbejde med mange-til-mange relationer. Databasen opfylder ikke 3. normalform, da price indenunder orderline er afhængig af hvilket topping og bottom id, der er valgt. Det samme er sandt for price under order. For at opfylde 3. normalform, ville man blive nødt til at tage prisen direkte fra topping og bottom tabellerne. Det ville nok involvere noget brug af joins i SQL'en, og ville være en del mere besværligt for os end den løsning vi valgte med price. Det samme gælder for vores price i order,

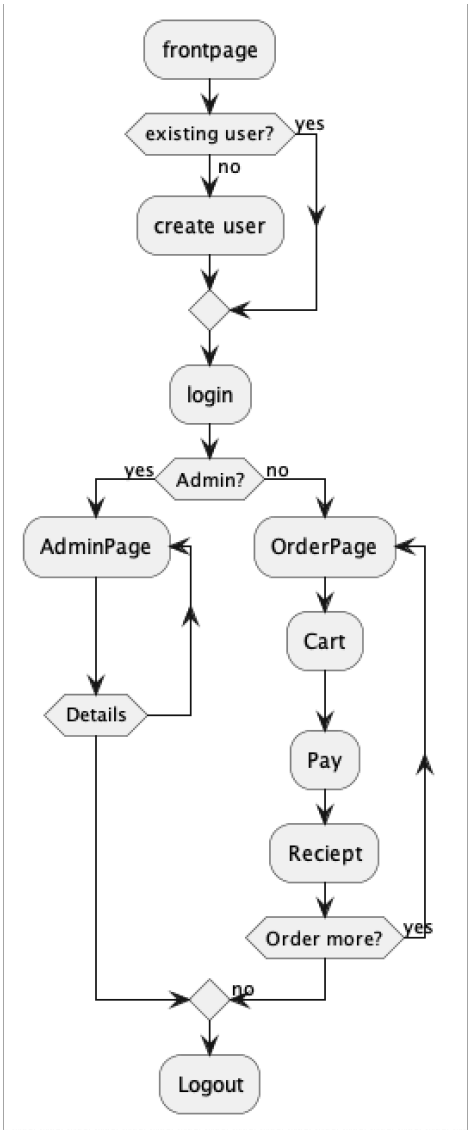
da den også er afhængig af price i orderline. Grunden til vi har gjort det sådan er bl.a. så man i admin nemt kan hente alle orders ned og se deres pris for hver af dem. Ellers skulle man finde alle orderlines tilhørende en order, og alle priserne på bottom og toppings til hver orderline, og derefter bruge de priser lagt sammen ganget med amount for at finde prisen for hver orderline, og så lægge prisen for hver orderline sammen for at finde den samlede order pris. Alle attributter i tabellerne er sat til not null, da ingen af dem umiddelbart ville give mening at kunne have en null værdi.

Domænemodel:



Den ovenstående model repræsenterer en domænemodel for koden, der anvendes til at skabe et overblik over klasserne. Selvom domænemodellen ikke er lige så detaljeret som et klassediagram, fremhæver den stadig forholdet mellem de forskellige klasser. Vores overvejelser har primært drejet sig om, hvordan relationerne mellem de forskellige klasser skulle være organiseret. Vi konkluderede, at sammensætning var den passende relationstype, da de fleste klasser ikke ville kunne eksistere uden hinanden. Der var dog undtagelser, som f.eks. Database Exception, der aggregerer til de forskellige controllere. Disse kan eksistere selvstændigt, men er grundlæggende ved arbejdet med databaser. Cart aggregerer også i forhold til OrderLineController og OrderController, da disse kan eksistere selvstændigt, men er afgørende for brugeroplevelsen på siden. Haiku er ligeledes ikke essentiel for siden, men udgør en ekstra funktion med henblik på brugeroplevelsen. Diagrammet kunne have gavn af flere pile, da de forskellige klasser har adgang til alle entities via metoderne. Vores overvejelser har været “overflødige” grundet dette var det sidste vi lavede.

Navigationsdiagram:



Ovenstående er sidens navigationsdiagram. Navigationsdiagrammer bruges til at gøre det nemmere at kunne danne sig et overblik over hvordan man navigerer fra side til side. En ting som skal klargøres lidt er, at man fra stort set alle sider efter login kan logge ud, men vi havde nogle udfordringer med at vise det ordentligt på diagrammet. Man kan også navigere fra cart tilbage til orderpage. I diagrammet er der en if, der spørger om man er admin. Hvis man logger på en user som er admin bliver man dirigeret over til adminpage, og hvis man ikke er admin ryger man over på orderpage.

Særlige forhold:

Vi gemmer vores user i en session når vi logger ind. På den måde ved vi altid hvilken user, der er logget ind. Vi har også et cart objekt der er gemt i session, som er vores shoppingvogn, der holder styr på hvilke orderlines, som en user har valgt. Ved at have en cart, så kan man holde styr på en brugers bestillinger uden at skulle gemme det i vores database hele tiden. Vi har også gemt vores informationer om bottoms og toppings i en session attribut, så vi kun behøver at hente den information ned fra databasen en gang.

For at løse US-2 skal man kunne gemme en ordre. Oprindeligt overvejede vi at tilføje en knap på indkøbskurvens side, som kunne gemme kurven i databasen. Udfordringen er lidt, at vi også har brug for at kunne slette den fra databasen senere. Dette ville ske ved at logge ind. Således ville den gemte ordre blive lagt i kurven og slettet fra databasen. Ulempen ved denne tilgang er tydelig: Hvis en ordre er blevet gemt, og brugeren logger ind og logger ud igen uden at fuldføre købet eller gemme det igen, så slettes ordren fra databasen, og brugeren står tilbage uden at have den gemt nogen steder.

For at løse dette valgte vi at automatisk gemme kurven, når brugeren logger ud. Dette sikrer, at man altid kan se, hvad der var i kurven ved den tidligere session. En fordel ved denne tilgang er, at kurven med den ubetalte ordre bliver gemt, men man skal kun snakke med databasen, når man logger ind, logger ud eller betaler.

En alternativ løsning kunne være at tilføje en knap, som beskrevet tidligere, men i vores nuværende system oprettes der altid en ny ordre i databasen, når der betales. Hvis knap ideen skulle være succesfuld ville det kræve at man kan lave ubetalte ordrer om til betalte ordrer, men hvis man også skal kunne ændre på sin gemte ordre så skal man lige pludselig til at ændre ordrens totale pris og lave nye orderlines i databasen, hvilket ville kræve ændringer af den måde betalingen virker i vores system. Dette ville formentligt være nemmere hvis alt var på 3. normalform i databasen.

Når man logger på, så tager vi mailen og passwordet som brugeren har skrevet, og bruger det som input i en metode i vores UserMapper. Denne metode prøver at finde en bruger med email og password der matcher i databasen. Hvis dette lykkes kan man logge ind, ellers kan man ikke. I forbindelse med log ind så er den eneste rigtige sikkerhed at man ikke kan se det password, som brugeren skriver på siden. Der bliver skrevet “****” i stedet for “1234” for eksempel.

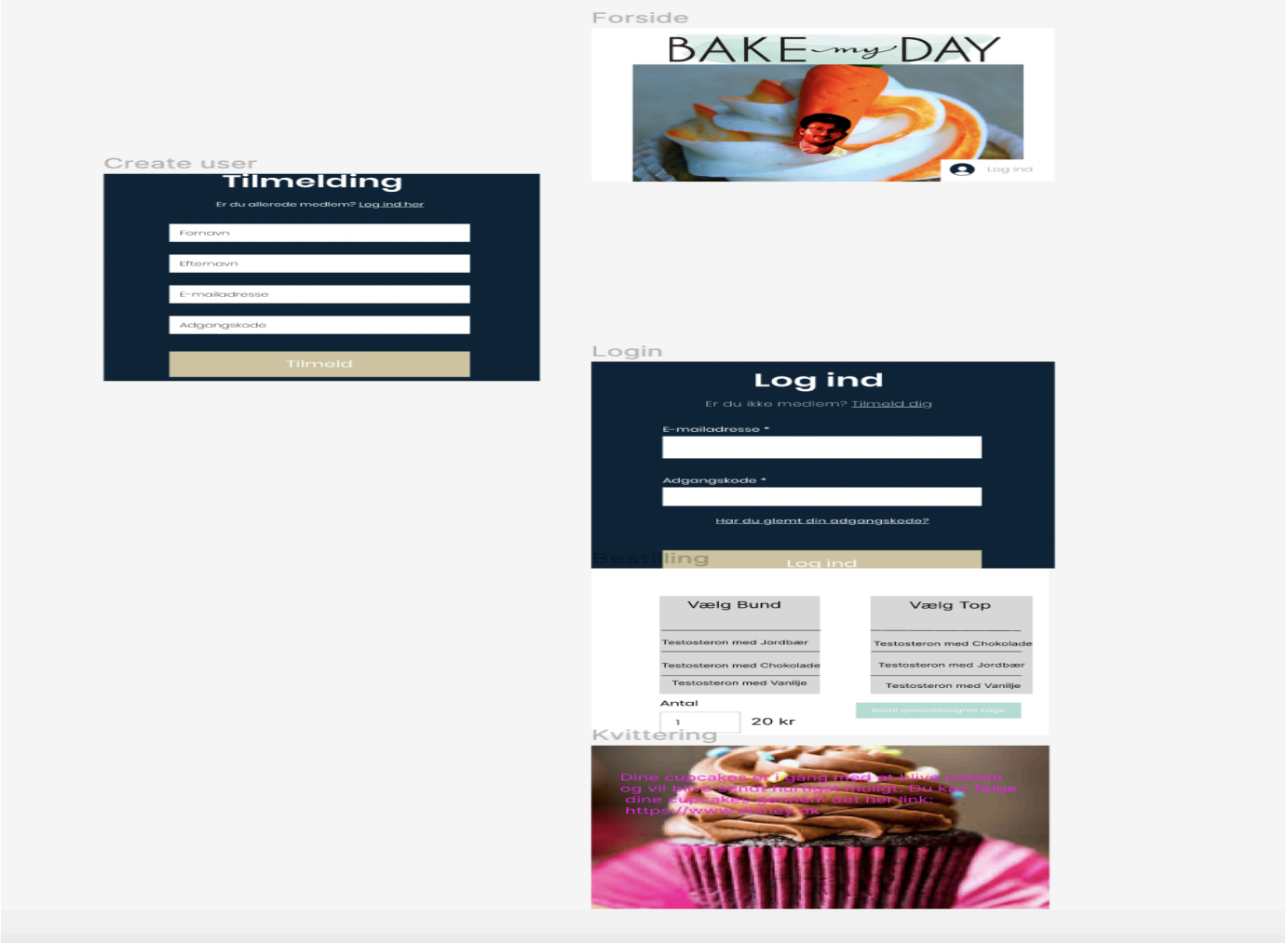
Status på implementation:

Vi har i dette projekt formået at tage hensyn til alle user stories, som blev givet. Koden er i en acceptabel tilstand med de rigtige tilgængeligheder til brugere/admins, og siden er blevet stylet. Vi har forsøgt at håndtere de vigtigste exceptions, altså fejl som kan forekomme på hjemmesiden, når man navigerer den. Det gælder især alle tekstfelter, hvor man kun må skrive bestemte ting, eller hvor man ikke må lade dem stå tomme. En enkelt af den type fejl blev ikke løst, da vi ikke fik implementeret noget til at forhindre brugeren i at skrive bogstaver og andre specialtegn i "amount" tekstfeltet.

Proces:

I dette projekt valgte vi at forsøge at lave arbejdsprocessen mere struktureret end forrige projekter. Som en nævneværdig del af strukturen brugte vi et kanban board. Kanban-boardets hovedformål er at bidrage til et bedre arbejdsflow for holdet. På et kanban-board vil man typisk sætte begrænsninger for, hvor mange opgaver der kan være på hver sektion. Idéen er, at ved at begrænse mængden af opgaver man arbejder på samtidigt, kan man mere målrettet færdiggøre dem. Endvidere hjælper begrænsningen til at hjælpe andre gruppemedlemmer som sidder ved flaskehalse i projektet. I vores projekt var kanban boardet mere værdifuldt i det overblik, den giver over projektet, så alle gruppemedlemmer altid ved hvem der laver hvad, og hvad der mangler at blive lavet. Dette var med til at vi mere effektivt kunne arbejde på hver vores ting samtidigt.

Det kunne have været bedre, hvis vi havde haft en mere grundig gennemgang af user stories og krav, før vi begyndte at implementere databasen. Ved at tage udgangspunkt i vores user stories kunne vi have skabt en mere præcis og relevant databasestruktur fra begyndelsen. Det ville have sparet os meget tid, da vi ville undgå at skulle ændre databasen flere gange undervejs i projektet. En mere omfattende analyse af vores user stories ville have givet os et bedre indblik i, hvilke data der var nødvendige, hvordan de skulle organiseres, og hvordan de ville blive brugt på hjemmesiden. Dette ville have bidraget til en mere effektiv og målrettet arbejdsproces, hvilket muligvis ville have resulteret i et bedre slutprodukt, da vi ville have haft mere tid til bl.a. ekstra features vi evt. kunne have fået med.



I starten af projektet blev der også lavet et mockup i Figma. Tanken var, at hjemmesidens visuelle design ville struktureres efter det originale billede, men det endte ikke med at slutproduktet kom til at ligne det. Dette skyldes i høj grad, at vi på det tidspunkt havde ændret mening i forhold til, hvordan vi ønskede hjemmesiden skulle se ud, da vi gik ret hurtigt over designet og bare skabte noget så vi vidste, hvad vores webside som minimum skulle indeholde af HTML-sider.

I vores gruppe blev vi hurtigt enige om vores prioriteringsrækkefølge. Vi ville lægge mest vægt på at først prøve at opfylde alle kravene der er for koden, og sikre os at der ikke er nogle bugs/exceptions. Først efter hjemmesiden var i en acceptabel tilstand, ville vi begynde på ting vi lagde lavere på vores liste af prioriteter, som fx styling og ekstra features. Vi kan klart tage med at vi skulle have gennemtænkt vores Figma yderligere ift. at tilføje evt. en kurv, logud, gå tilbage til

bestilling, osv. Så vi alle havde en bedre visuel forståelse ift. hvilke features der skulle tilføjes til projektet.

Planen for stylingsprocessen som start var at have et stylesheet for hver HTML-side vi havde, hvilket vi fandt ud af slet ikke gav mening ift. hvor meget mere effektivt det var at have et universelt stylesheet. Eksempelvis har websiden mange knapper, de knapper skulle alle have samme design og derfor var det nemmere at give knapperne på alle HTML-siderne samme klasse navn, så man kunne tage fat i dem gennem stylesheetet og ændre efter behov. Så næste gang skal vi have gennemtænkt vores figma bedre og starte ud med at have et stylesheet i stedet for en for hver HTML-side.

Noget vi klart burde have lagt mere vægt på var planlægning, før vi bare kaster os ud i at kode. Vi har også i tidligere projekter ikke brugt nok tid på planlægning, og vi har nu fundet ud af, at det godt kan betale sig. Når vi ikke planlægger vores projekter ordentligt ender det som regel med, at man skal bruge ekstra tid senere hen i forløbet på at finde ud af de ting man burde have fundet ud af i planlægningsprocessen. Det er derfor den vigtigste ting, vi har fået ud af fra at arbejde på dette projekt.