Olsker Cupcake

Gruppe 3

Projektstart:	16 / 04 / 2021
Projektslut:	26 / 04 / 2021

Rapporten er udarbejdet den: 27 / 04 / 2021

Informationer

Navn	**Navn**	**Navn**
Frederik Bilgrav Andersen	Janus Stivang Rasmussen	Gustav Ejby Bidstrup
Mail	**Mail**	**Mail**
cph-fa116@cphbusiness.dk	cph-jr270@cphbusiness.dk	cph-gb77@cphbusiness.dk
Github	**Github**	**Github**
github.com/FrederikBA	github.com/Janussr	github.com/GustavBidstrup
Hold	**Hold**	**Hold**
Hold A	Hold A	Hold A

Indholdsfortegnelse

Indledning

Vi har fået som opgave at udvikle en webshop til cupcake-bageriet *Olsker Cupcakes* beliggende i byen Olsker på Bornholm.
Projektet er et java webprojekt udviklet ved hjælp af *Java Servlets, HTML, CSS* og frameworket *Bootstrap*.

Baggrund

Vores webshop er udviklet til det for nyligt opstartede cupcake bageri på Bornholm i byen Olsker, der går ved navnet *Olsker Cupcakes*. Olsker Cupcakes har nye ambitioner, om at udvide og udvikle deres virksomhed. Virksomheden vil gerne have udviklet en splinterny webshop for at øge deres salg og følge med tidens mange teknologiske fremskridt.

Olsker Cupcakes har derfor søgt om hjælp udefra til at udvikle deres projekt og til dette, har de stille nogle krav: For det første, skal der kunne gennemføres en handel. Hos Olsker Cupcakes kan man selv sammensætte sin egen cupcake med valgfri bund og topping. Første krav var derfor, at kunden på hjemmesiden selv kunne vælge en valgfri bund samt topping og yderligere at kunne bestille og betale den sammensatte cupcake. Det er vigtigt for Olsker Cupcakes, at deres kunder kan oprette en profil på webshoppen og de vil gerne have, at der skal være et krav om at have en bruger for at betale deres ordre. Det skal være let for kunden at bruge hjemmesiden. Der skal være nem adgang til diverse informationer som balance til rådighed og brugernavn og det skal være muligt at rette de ordrer som kunden foretager sig, inden der bliver betalt.

Desuden skal hjemmesiden også kunne tilgås af en medarbejder af Olsker Cupcakes med en administrator konto. Olsker Cupcakes ønsker at kunne bruge hjemmesiden til holde styr på alle ordrer der er foretaget gennem tiden. Her skal der kunne ses hvad der er blevet købt og til hvilken pris, hvem der har købt ordren og hvornår ordren er placeret. De kunne også tænke sig en kundeoversigt hvor der kan håndteres ting som indsætning af balance til køb af cupcakes og yderligere at kunne se den individuelle kundes ordrer, så det er nemmere at slå oplysninger op.

Teknologivalg

Der er blevet brugt IntelliJ Ultimate Edition IDEA 2021.1, til at lave projektet. Vi har brugt programmeringssproget Java Development Kit version 11.0.6. Vi har lavet vores database med MySQL Workbench 8.0, og koblet den på Java ved brug af JDBC. Vores JSP(Java Server Pages) består af HTML5 og CSS.
Ved hjælp af en droplet fra Digital Ocean, og Apache Tomcat 9.0, er vores projekt online og deployed på nettet, sådan at alle kan tilgå den.

Krav

Kravet fra Olsker Cupcakes er, at de vil have en online shop, da de er ved at udvide og vil have en større kundekreds. De ønsker et brugervenligt system, så det er nemt for alle aldre at bestille cupcakes.

Vi har af Olsker Cupcakes fået udleveret nogle userstories der lyder således:

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 en ordre.

US-3: Som administrator kan jeg indsætte beløb på en kundes konto direkte i MySql, 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 mail og kodeord. Når jeg er logget på, skal jeg kunne min email på hver side (evt. i topmenuen, som vist på mock up'en).

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

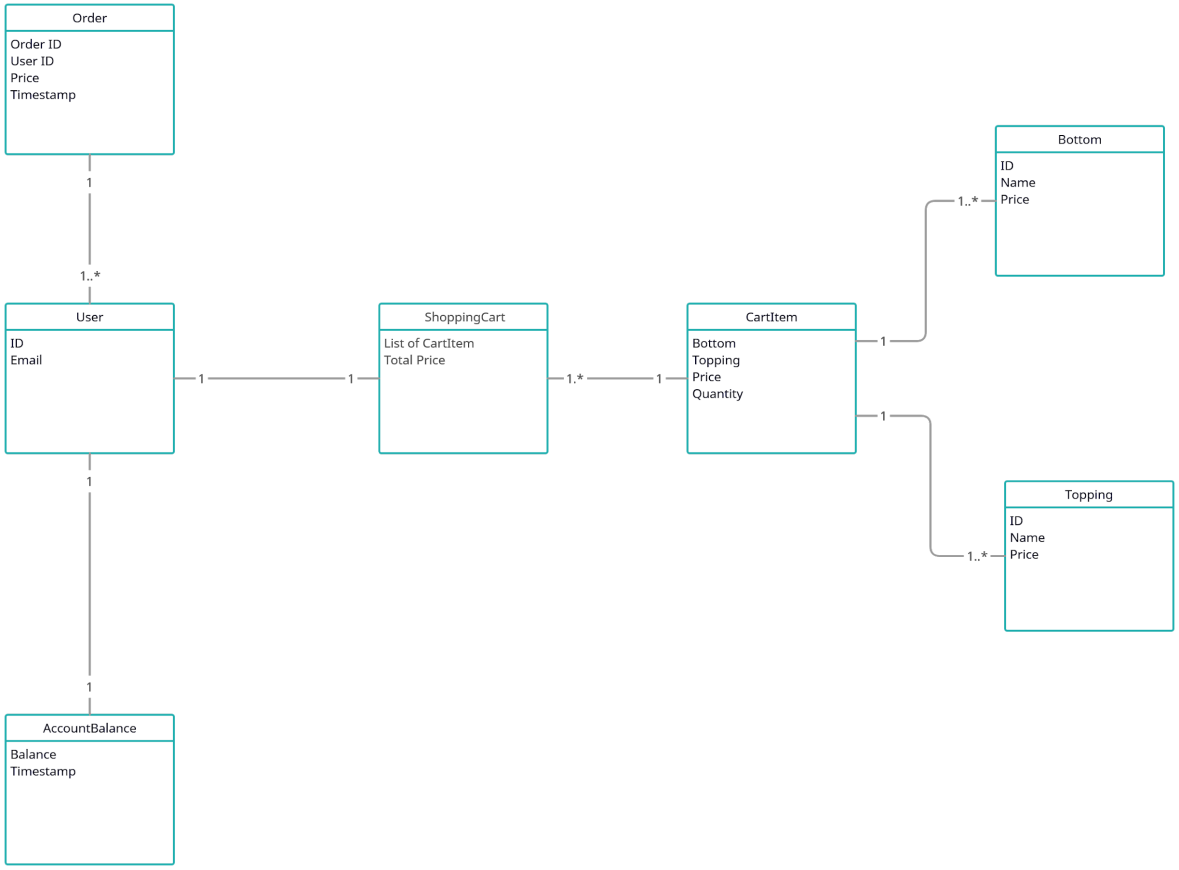
US-7: Som administrator kan jeg se alle kunder i systemet og deres ordrer, sådan at jeg kan følge op på ordrer og holde styr på mine kunder.

US-8: Som kunde kan jeg fjerne en ordre 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.

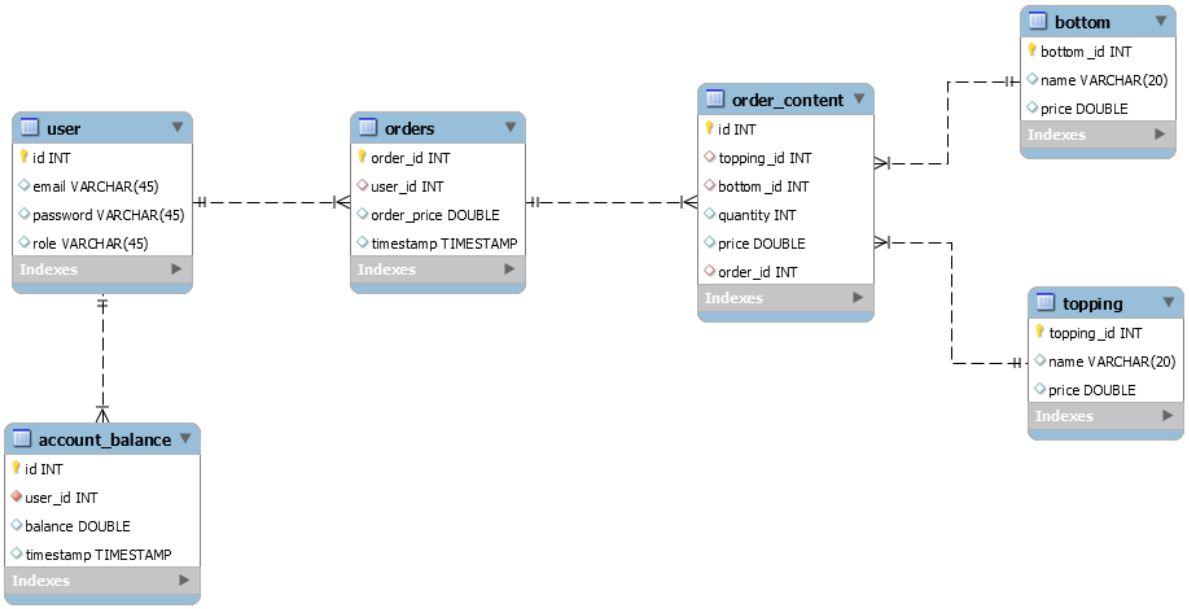
UML

Domænemodel



Vores domænemodel viser de domæne- eller entitetsklasser vi benytter os af i programmet, samt deres relationer. Herunder: *User, AccountBalance, Order, ShoppingCart, CartItem, Bottom, Topping*.

ER-Diagram



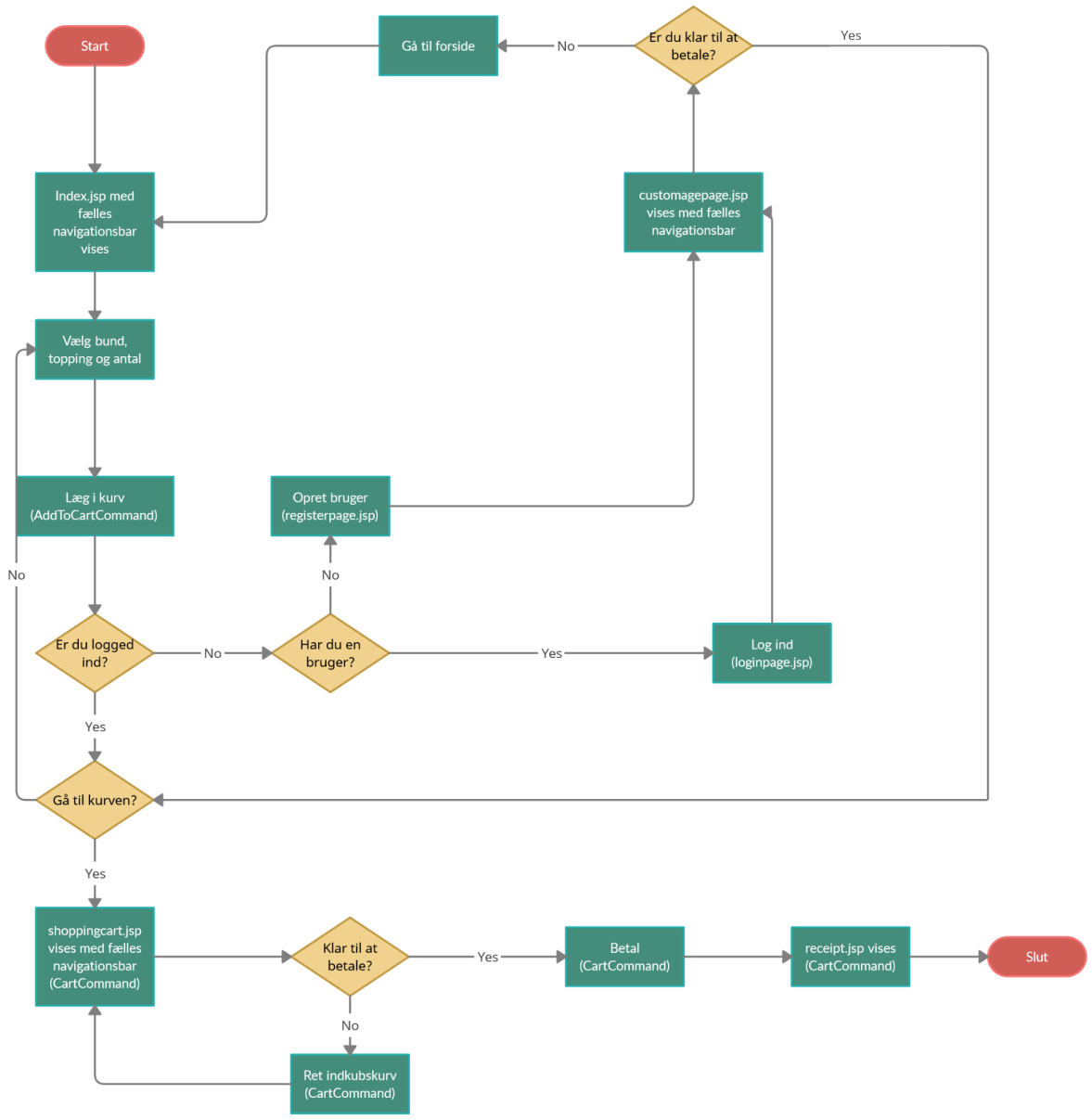
Ovenfor ses vores ER-Diagram, der viser hvordan vi har valgt at opsætte vores database, hver tabel har en *primary key*, som kan ses på dem der er markeret med en gul nøgle. Vi forbinder vores tabeller med *foreign keys*. Vores database er opsat efter den tredje normalform.

Relationerne i vores database er bygget således at én user kan have flere account_balance elementer. Vi har gjort det sådan, for at det skulle fremstå som en transaktion der får et nyt timestamp, hver gang en bruger har fået opdateret deres balance.

Vores user har også en én til mange relations i forhold til orders, da en bruger jo godt kan have flere ordrer gemt i systemet.

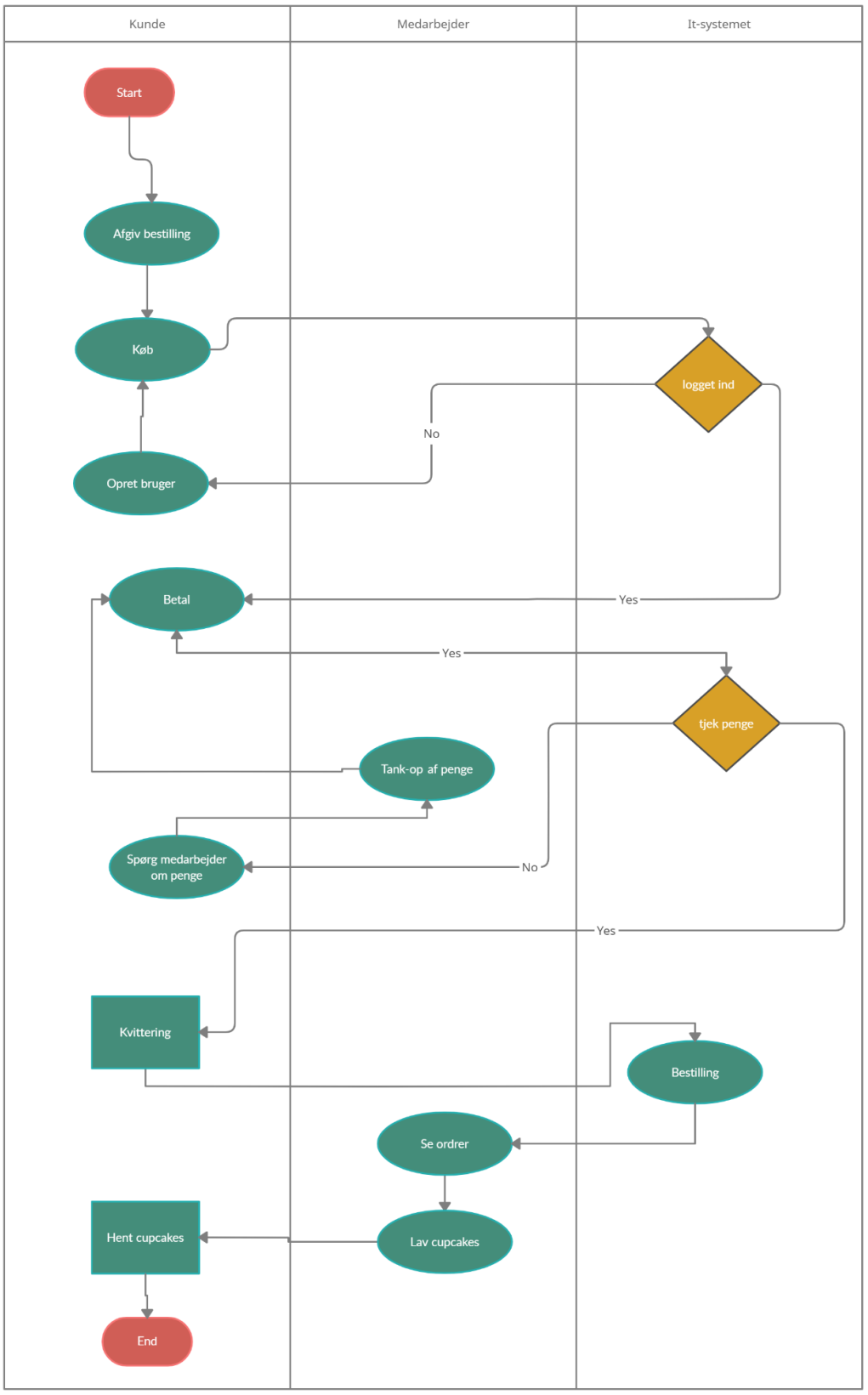
Vores order tabel har en én til mange relation til den normaliserede tabel order_content, da én ordre godt kan have flere forskellige sammensatte cupcakes. Det samme gælder for order_content tabellen med henholdsvis, topping og botttom som også har en mange til én relation.

Navigationsdiagram (Kunde)



Vores navigationsprogram beskriver den proces der sker når en **kunde** først tilgår hjemmesiden og tager derfor udgangspunkt kun udgangspunkt i kundens aktivitet og altså ikke en admin. Vi har benyttet os af en fælles navigationsbar på hver side, hvor det er muligt at logge ind og navigere sig tilbage til forsiden eller til kurven skulle det ønskes. Vi har i denne omgang undladt at lave et ekstra navigationsdiagram der beskriver en admins aktivitet.

Aktivitetsdiagram



Aktivitets diagrammet som ses ovenfor, bliver brugt til at se fremgangsmåden i arbejdsprocessen, fra at kunden bestiller en cupcake til at de henter deres ordrer. Dette er vores eksempel på TO-BE aktivitets diagrammet for Olskers cupcake.
Som modellen viser, starter kunden med at give en bestilling, og derefter køber ordren, når den har det indhold de er tilfredse med. Hjemmesiden(It-system) tjekker så om kunden er logget ind, hvis kunden ikke er det, er der mulighed for at oprette bruger, eller logge ind. Hvis kunden opretter en bruger, bliver de sendt tilbage til køb, og har nu mulighed for at gå videre, da de nu har en konto.
Når de så er logget ind, kan de betale deres ordrer, hvilket fører til endnu en ´decision´, om brugeren har penge nok på kontoen til at betale deres ordrer. Hvis ikke, skal de spørge medarbejderen, om at få penge på kontoen. Når så medarbejderen har gjort det, kommer brugeren tilbage til betal. Når de så har betalt, og de har penge nok, bliver der vist en kvittering for brugeren, og hjemmesiden(it-systemet) registrerer bestillingen, hvorefter medarbejderen kan se ordren, lave cupcakes der hører til ordren, som kunden til sidst kan hente i butikken.

Særlige Forhold

Vores projekt er bygget med java servlets som indeholder nogle attributter kaldet scopes. I opgaven har vi benyttet os af henholdsvis. *RequestScope, SessionScope og ApplicationScope*.
Disse tre *scopes* er en måde at hente informationer frem på en hjemmeside som der tilføjes fra java servletten. Hvert scope har en bestemt levetid: Request Scopet vil kun hentes i det siden genindlæses og har altså den korteste levetid. Session Scopet lever i en såkaldt "*session"* som lever så længe brugeren interagerer med siden. Application Scopet bliver startet fra det tidspunkt hjemmesiden bliver deployet og lever så lang tid at hjemmesiden er online.

I projektet har vi benyttet os mest af session scopet. Det bruger vi bl.a. til fremvisning af lister, for eksempel. cart items, ordrer og kunder. Vi benytter det også på vores indkøbskurv da den skal leve så lang tid kunden bruger hjemmesiden, så der kan navigeres frem og tilbage mellem indkøbskurven og forsiden, hvis flere cupcakes skal bestilles.

Request scopet benytter vi enkelte steder hvor siden kun skal tilgås en enkelt gang og blot fremvise nogle informationer. Det er sider som *"ordercontentpage"* og *"userorderpage"* hvor vi i stedet har valgt at håndtere listerne i request scopet.
Til sidst har vi benyttet os af application scopet en enkelt gang. Dette bruges når vi henter vores *bottoms* og *toppings* ned fra databasen og fremvises på siden så brugeren kan vælge dem i en dropdown-menu.

Vi har også brugt *exceptions* på dansk; fejlhåndtering. Fejlhåndtering bruges til at håndtere eventuelle fejl der kan opstå, når en bruger navigere hjemmesiden. Et eksempel på dette er i vores *addToCardCommand* hvor vi ved hjælp af fejlhåndtering, kan sikre os, at brugeren formår at få alle de nødvendige værdier med der skal bruges for at sammensætte en cupcake før at den kan lægges i kurven. Vi har også brugt fejlhåndtering på selve indkøbskurven, for at sikre os at kunden er logget ind på systemet og, at kunden har nok penge på deres konto før købet kan gennemføres. Dog kunne vi godt ønske at have fejlhåndteret yderligere, det kunne for eksempel være på login-systemet især. Her kunne man blandt andet tjekke om den mail man opretter sig med enten er gyldig eller hvis den allerede er i brug på hjemmesiden.

Vores database er normaliseret efter tredje normalform. Eksempler på dette, er at vi har opdelt tabeller som for eksempel vores *user* og *order* tabel i flere mindre tabeller. I vores user tabel, har vi valgt at implementere en tabel der gemmer den balance som brugeren bruger på hjemmesiden til at handle med kaldet *AccountBalance*. Denne tabel har et bruger ID fra vores bruger tabel, så vi ud fra hver bruger kan se deres beløb til rådighed. Når vi i programmet ved hjælp af *JDBC* for eksempel skal indsætte et beløb hos en bruger, kræver det, at vi henter informationer ned fra begge tabeller, da vi skal bruge id'et i bruger tabellen til at indsætte i vores *AccountBalance* tabel.

Status på Implementation

Vi har nået og opfyldt alle de user-stories og krav der blev stillet til projektet. Men det er ikke ens betydning med at der ikke er plads til forbedring, for det er der altid.

Diverse mangler vi som kunne være gode at implementere vil som det første nok være mere fejlhåndtering og validering af brugerinput. Som forklaret i ovenstående sektion er det fejlhåndtering ved blandt andet login-systemet der mangler. Vi kunne også godt tænke os yderligere fejlhåndtering de steder, vi har brugerinput for at sikre os, at de værdier brugeren skriver, rent faktisk er det intenderede input. For eksempel, sikre sig at der bliver skrevet et tal og ikke et bogstav eller tegn, når man skal vælge antal cupcakes der skal lægges i kurven.
Vi har stylet siden, og den er også responsiv til store, mellem og små størrelser og det er vi sådan set også tilfredse med, da vi har fulgt det mock-up vi fik til rådighed. Men der kunne godt forbedres på det så vi opnår en (måske) bedre user-experience.
Vores kode vil have god gavn af lidt refaktorering så det bliver mere læseligt og forståeligt. Vi havde lidt spørgsmål til designet af startkoden hvilket gjorde at vi var lidt tilbageholdende med brug af diverse service-klasser.
Vi har ikke taget brug af hverken unit- eller integrationstest hvilket er ærgerligt, men da det ikke opstod som krav, brugte vi mere tid på at lære startkoden ordentligt at kende så vi var mere fortrolig i vores kode.

Proces

Vores planer for projektet før vi startede, var at være så produktive fra starten som muligt, så vi kunne få et forspring. I praksis gik det tilfredsstillende godt, vi fik en god start og var motiveret til at bruge mange timer om dagen på projektet, om det var sammen på discord og zoom eller hver for sig.
Noget der gik godt var kommunikationen i gruppen, og at der var god mulighed for at få hjælp i gruppen, hvis de andre kunne svare på spørgsmål man måtte have.
En ting der kunne være gået bedre var vores plan for at følge scrum boardet, det blev lavet, og fik tasks på sig, men det virkede til at det ikke var så nødvendigt i et lille projekt som dette, da der var god kommunikation i gruppen, så det blev sat lidt til side.
Det vi har fået ud af hele processen, med denne opgave er et større overblik om hvordan alt hænger sammen i et større projekt.. Vi har også fået en bedre forståelse for den arkitektur som vi skal bruge til eksamensprojektet. Vi føler alt er gået som planlagt, så der kommer ikke til at blive store forandringer til næste projekt, udover at vi vil tage mere brug af scrum boardet.