Isker Cupcake

Projekt af

- **Ye Long he**, cph-yh24@cphbusiness.dk, https://github.com/Stiikish
- **Betül Iskender**, cph-bi32@cphbusiness.dk, https://github.com/betuliskender
- **Mia Falk**, cph-mf325@cphbusiness.dk, https://github.com/SnackMinister

Projekt udarbejdet 4.-8. april 2022.
Rapport udarbejdet 19.-21, april 2022.

INDHOLDSFORTEGNELSE

INDLEDNING

Olsker Cupcakes er et dybt økologisk iværksættereventyr fra Bornholm, som har ramt den helt rigtige opskrift på de perfekte cupcakes. Med store ambitioner og drømme, håber Olsker Cupcakes at kunne brede sig ud, ikke kun på Bornholm, men til resten af Danmark.

Et par hipsters fra København har taget den lange tur til Bornholm, forbi bageriet og indsamlet nogle krav og lavet en halvfærdig mockup af en tiltænkt forside.

Denne opgave tager udgangspunkt i forskellige krav fra Olsker bageri. De vil gerne have implementeret en online webshop. Den skal give deres kunder mulighed for bestille online og dernæst afhente i butikken.

Projektet er udarbejdet i Java, MySQL, JDBC og kører på en lokal Tomcat server som håndtere vores udkast af en online webshop. Målgruppen for dette projekt er andre medstuderende.

BAGGRUND

Firmaet ønsker at opnå en bredere kundekreds, ved at få Olsker Cupcakes online og samtidig gøre det nemmere for deres kunder at bestille cupcakes fra deres bageri. Ved at gå online, får kunder mulighed for at vælge forskellige toppings og bunde, for derefter at hente deres ordre fysisk i butikken.

TEKNOLOGIVALG

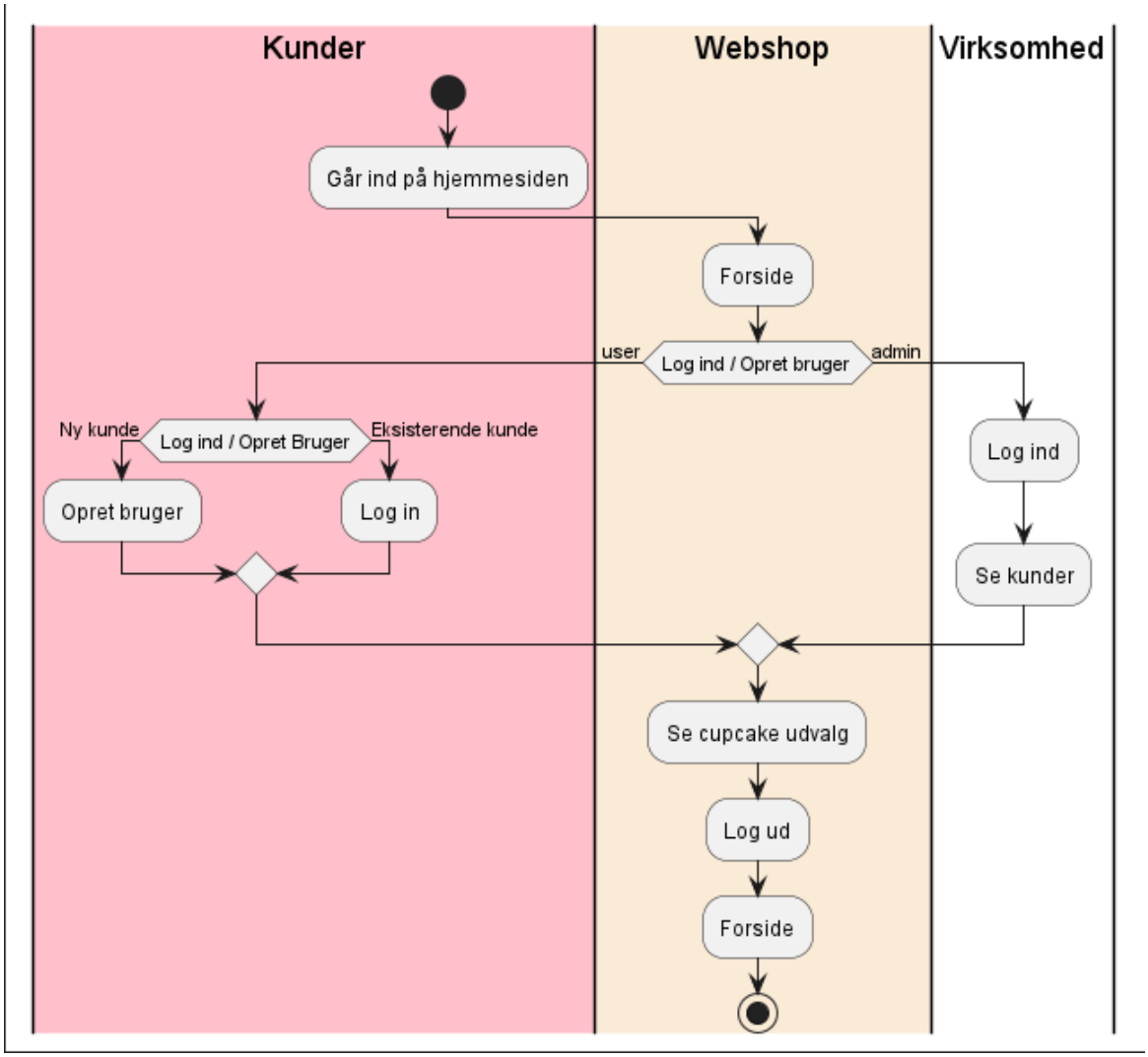
Følgende teknologier er anvendt i Cupcake projektet.

- Opgaven er lavet i **IntelliJ IDEA 2021.3.2** med programmeringssprog **Java version 17.**
- **Servlets** er implementeret til at håndtere klientens forespørgsler og svar (*request, response*).
- **MySQL Workbench version 8.0** til datahåndtering.
    - f.eks. at hente og gemme en brugers oplysninger m.m.
- **Tomcat Apache version 9.0.62** til at håndtere serverdelen og få hjemmesiden op at køre.
- **JDBC** til at få forbindelse til databasen og **JSP** til at vise vores **HTML** sider.
- **CSS** og **Bootstrap 4.0/5.0** til styling af hjemmesiden.

KRAV

- **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 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 e-mail og kodeord. Når jeg er logget på, skal jeg kunne se min e-mail 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 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.

AKTIVITETS-DIAGRAM

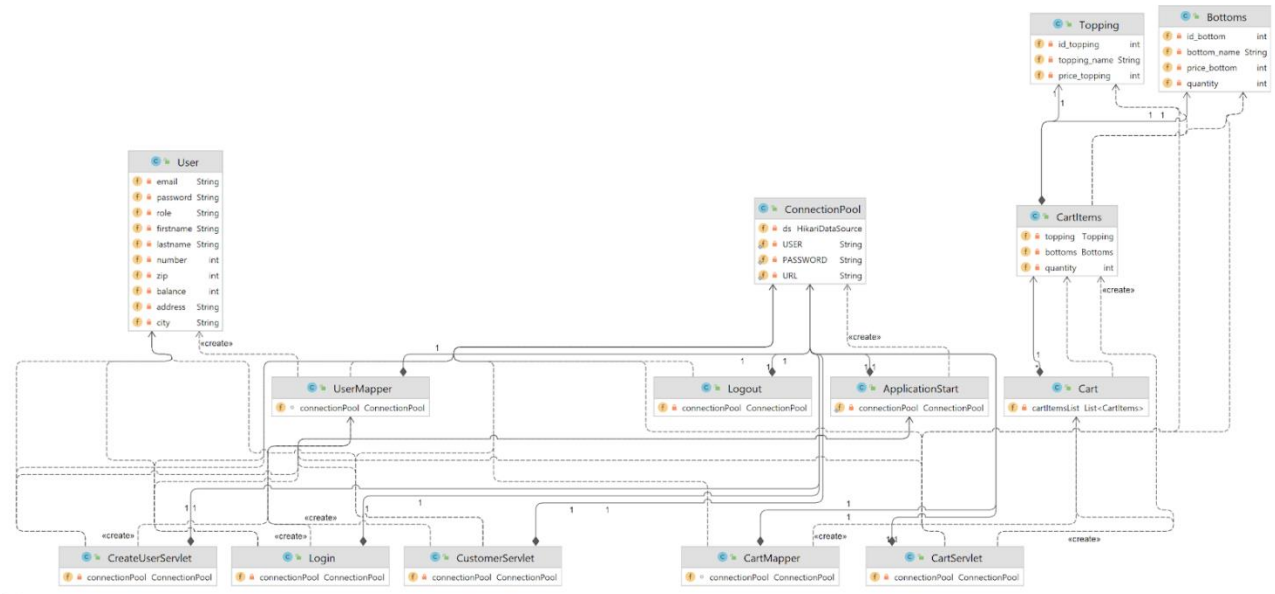


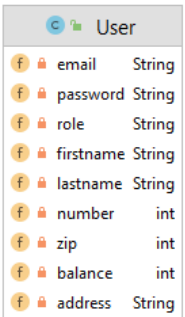
Ovenstående aktivitets-diagrammet viser, hvilke valgmuligheder man har som bruger eller admin.

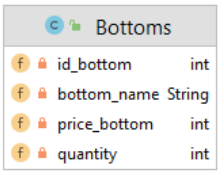
Før man er logget ind, vil man som bruger stå på forsiden. Her er der mulighed for enten at logge ind, som admin eller hvis man er en eksisterende kunde, eller oprette en ny konto, hvis man er ny kunde.

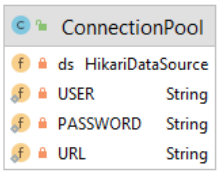
Det er ligeledes også muligt at se webshoppens udvalg. Under udvalget ses forskellige cupcake toppings, som Olsker Cupcakes tilbyder. Når man har valgt en topping, skal man derefter vælge hvilken slags bund der ønskes samt antal. Er man logget ind som admin, har man mulighed for at se alle kunder oprettet i webshoppen. Admin har også mulighed for at slette en kunde, hvis kunden ikke længere ønsker at være tilknyttet webshoppen.

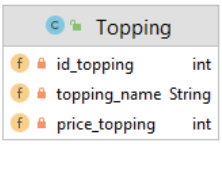
DOMÆNEMODEL

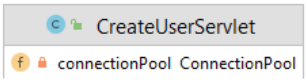


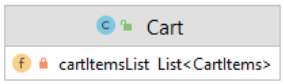


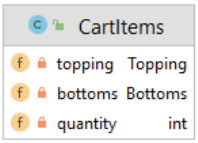


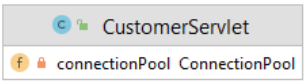


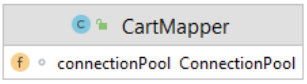


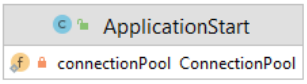


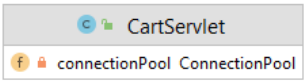


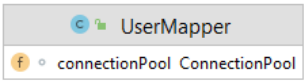


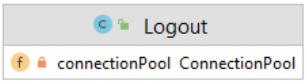


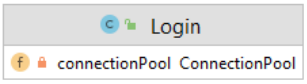






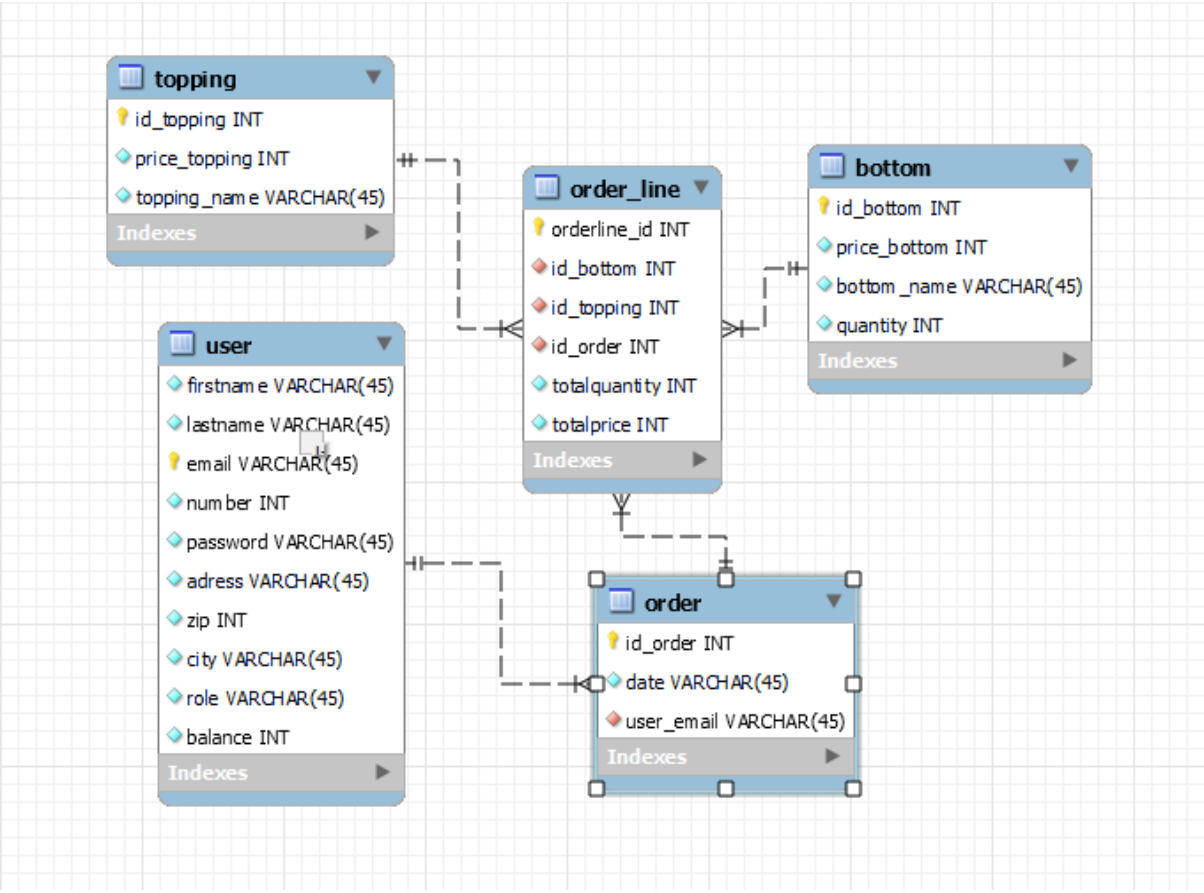






Ovenstående domænemodel, viser alle relationerne mellem de forskellige klasser. Til at starte med var der hverken klasser, som holdte top- og bundobjektet eller til at samle cupcaken. For at kunne lave en samlet cupcake, blev koden nødt til at refaktureres til, at der blev et topobjekt og et bundobjekt. Det kunne derefter videreføres til klassen 'ordrelinje', som derefter videreføres til en 'kurv' klasse.

EER-DIAGRAM



I projektet er der valgt, ikke at normalisere til 3. normal form. Ud fra opgavebeskrivelsen, fremgår det ikke, at være nødvendigt at normalisere yderligere i 'user' tabel med hensyn til *adresse*, *by* og *postnummer*. Kundens informationer bliver ikke brugt yderligere, udover ved oprettelsen. Man kan normalisere det til 3. normal form på et senere tidspunkt, hvis der skal laves flere funktionaliteter, som f.eks. at kunne sende en ordre hjem til en kunde.

Der er valgt at bruge 1- mange relationer i vores EER-diagram, da det giver bedst mening, i forhold til hvor mange informationer de enkelte tabeller skal indeholde. F.eks. at en kunde kan have, ikke kun én, men mange ordrer og en ordre sagtens kan indeholde flere toppings/bunde og ikke kun én.

I de fleste tabeller bruges et autogenereret ID som nøgle, med undtagelse i 'user', hvor der bruges e-mail som nøgle. Det bliver gjort, da der laves et check på hvorvidt en bruger er en kunde eller en admin.

Der var mange overvejelser under processen af EER-diagrammet, hvor der blev talt om hvilke tabeller, som skulle med.

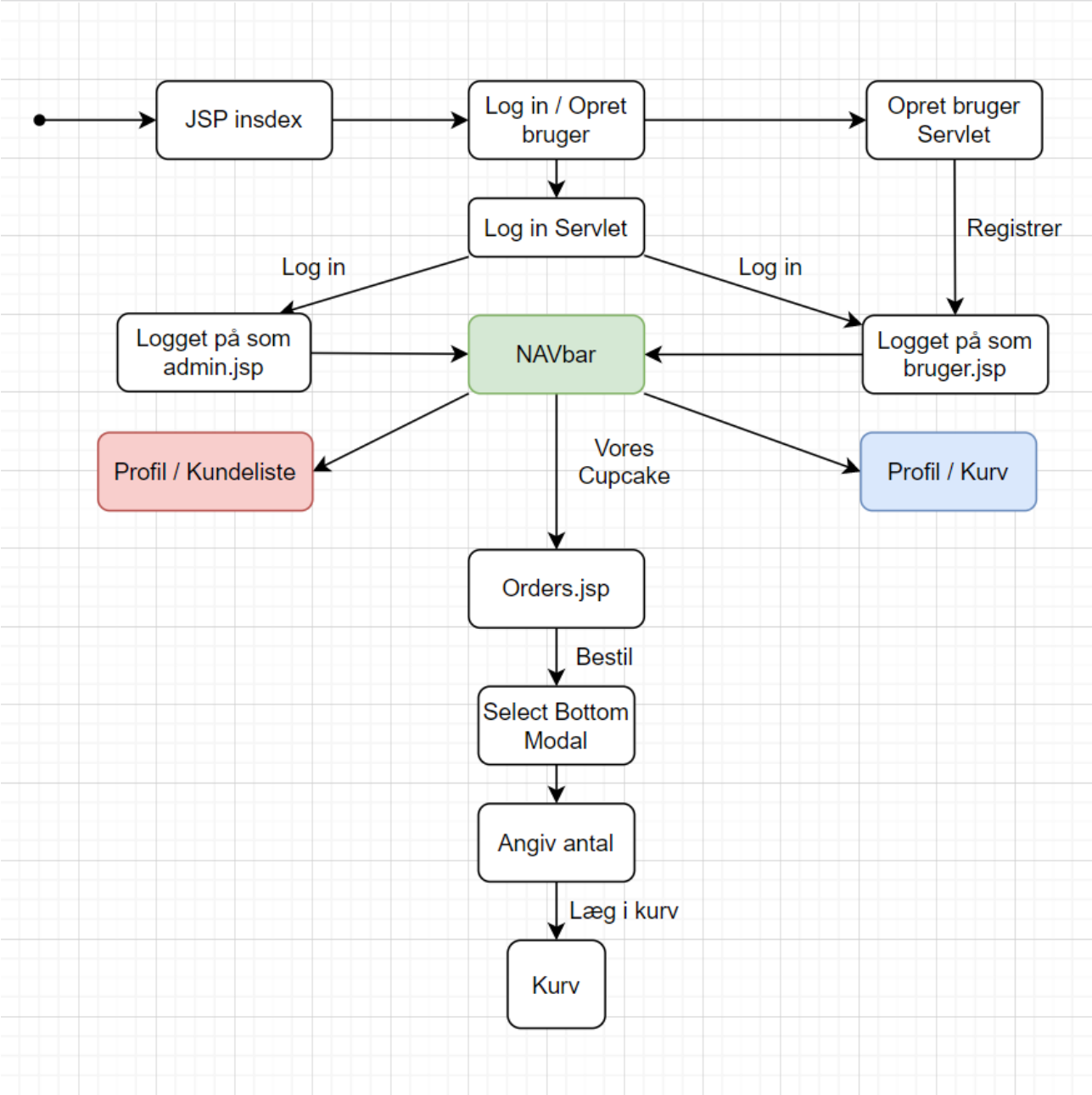
Fordi projektets størrelse ikke er større og der ikke forefindes flere funktionaliteter, end at en kunde skal kunne bestille en ordre af cupcakes bestående af topping/bund, er det blevet gjort så simpelt som muligt.

I 'ordrelinje' tabellen, fås der fremmednøgler fra 'topping' og 'bottom', for at få de informationer fra de to tabeller ind på 'ordrelinje ' tabellen.

'Order_line_id' kommer med videre som fremmednøgle i 'order ', da den samlede information skal sendes videre. I 'order ' kommer en fremmednøgle ind i form af e-mail fra 'user', da 'user' og selve bestillingen skal forbindes.

→ TJEK MED JON I MORGEN OM ORDER_LINE_ID

NAVIGATIONS-DIAGRAM



Ud fra ovenstående navigations-diagram, er NAVbaren markeret med grøn, da det er hvor det fremgår, om man er logget ind som bruger eller admin. Det kommer til udtryk, som kan ses i den røde(admin) og blå(bruger) boks. Som admin kan man ved sin profil se en kundeliste og ved bruger ses en kurv.

SÆRLIGE FORHOLD

Der er i projektet brugt forskellige *scopes*, afhængig af hvilke informationer en bruger eller admin skal kunne se. Hvis ikke man er logget på, er der valgt at bruge *application scope*, så at alle kan se udvalget af cupcakes, både topping, bunde og priser. Dog er der ikke muligheden for at lægge i kurv, medmindre man er logget ind.

Når man logger ind, bruger vi *session scope*. I *session scopes* bliver der gemt informationer ud fra en brugers oplysninger. Fx. hvis en kunde logger ind, vil en session gemme oplysninger om den kunde ud fra kundens e-mail, navn etc. Det samme vil ske hvis man loggede ind som admin. Via *session scope* kan der udover at gemmes oplysninger, også hentes oplysninger om kunden/admin.

Der er ikke blevet brugt nogle former for *kryptering* i projektet, da det ikke er undervist om på 2. semester. Den eneste sikkerhed der forekommer, er når der kørers 'login' og 'betal' på *doPost* serveren i stedet for *doGet*, så følsomme oplysninger ikke vises på URL.

Ved 'login' laves et check på den bruger som prøver at logge ind, for at se hvilken 'rolle' brugeren har. Den data som brugeren indtaster ved login, bliver sendt til vores 'login' servlet, som kontrollere på e-mail og password. Da 'rollen' er sat på e-mail, kan der herved kontrolleres hvilke funktioner brugeren skal have adgang til.

STATUS PÅ IMPLEMENTATION

Tabellen nedenfor viser status på implementering af de forskellige *user stories* som kommer fra Olsker Cupcakes. Da der i gruppen har været problemer med at få noget af funktionaliteten til at virke, er der funktioner som ikke er blevet implementeret, som er en relativ vigtig del af opgaven.

Funktionaliteter som 'læg i kurv', 'bestil ' og 'betal ', er ikke implementeret. Derfor er der blevet prioriteret, det som gruppen kan finde ud af at implementere. Der er tilføjet en ekstra funktion for admin - at admin kan fjerne en vilkårlig bruger, såfremt bruger ikke længere ønsker at være en online kunde. Der er dog oprettet både 'Servlet ' og 'JSP ' til 'users', der gør at de kan redigere deres oplysninger. Dertil er der også lavet 'Servlet ' og 'JSP ' til 'users', der gør at de kan tilføje flere penge til deres konto. Begge er dog ikke fuldt implementeret endnu.

User stories:	Implementeret	Ikke implementeret
US-1. Som **kunde** kan man bestille og betale cupcakes med en valgfri bund og top, sådan at man senere kan afhente ordren.		Ikke implementeret
US-2. Som **kunde** kan man oprette en konto/profil for at kunne betale og gemme en ordre.	Implementeret. Man har mulighed for at oprette en konto der gemmer ens oplysninger, såsom saldo, e-mail etc.	
US-3. Som **admin** kan man indsætte et vilkårligt beløb på en kundes konto, direkte i MySQL, så en kunde kan betale sin ordre.	"Implementeret". Det blev aftalt i plenum, at bruger selv indtaster et vilkårligt beløb ved oprettelse af ny konto.	
US-4. Som **kunde** kan man se ens valgte ordrelinjer i en indkøbskurv og se den samlede pris.		Ikke implementeret
US-5. Som **kunde** eller **admin** kan man logge på med e-mail og kodeord. Når man er logget på, skal man kunne se ens e-mail på alle sider.	Implementeret. Man kan både som admin og bruger se sine oplysninger under profil. Virker på alle sider.	
US-6. Som **admin** kan man se alle ordrer i systemet og se hvad der er bestilt i hver enkelte ordrer.		Ikke implementeret

US-7. Som **admin** kan man se alle kunder og deres ordrer i systemet, sådan at man kan følge op på ordrer og holde styr på kunder.	Halvt implementeret. Som admin kan man se alle kunder, der er oprettet i systemet, men man kan ikke se deres ordrer, da ordre ikke er implementeret	
US-8. Som **kunde** kan jeg fjerne en ordre fra indkøbskurven, så ordren kan justeres.		
US-9. Som **admin** kan man fjerne en ordre, så systemet ikke kommer til at indeholde ugyldige ordrer. F.eks. hvis kunden aldrig har betalt.	Halvt implementeret. Som admin kan man fjerne en bruger, hvis en bruger ikke længere ønsker at have en konto hos bageriet. Ordre er ikke implementeret.	

PROCES

Ved udlevering af opgaven, valgte vi tre at gå sammen, da vi mente at vi lå nogenlunde ens niveaumæssigt. Derved kunne vi få kodet mere i samme hastighed og lære mere under processen.

Til selve projektet var det meningen at der skulle bruges *Git*. Dog opstod der komplikationer på første dagen, hvor vi fik udleveret start koden. To ud af tre af gruppens medlemmer ikke kunne få det til at fungere på deres PC. Det betød at der på anden dagen, blev nødt til at ændres taktik i forhold til brug af *Git.* I stedet blev det til individuelle versioner, hvor én i gruppen så kunne pushe til *GitHub* og de andre kunne hente versionen ned på deres egen PC. Det har været at arbejde på i stedet for samme projekt. Dette tog lidt længere tid, da der ofte var kode som skulle rettes til for at få projektet op at kører på andre computer. Vigtigt var at databasen og dens tabeller blev kaldt det samme.

Selve processen herefter gik godt, og der blev hurtigt lagt en plan for både mockup og EER-diagram. Vi endte med at gå lidt væk fra det udleverede mockup og i stedet med nogle forskellige mockup's og designs, da de forskellige projekter blev rodet med hjemme og som hygge projekt.

Der blev ikke brugt nogen former for *kanban board* til at holde styr på deadlines og opgaver, da vi ikke havde lang tid til at udvikle det endelige produkt. I stedet brugte vi *Discord* til daglig kommunikation samt fremmøde på skole.

Ud fra ovenstående udfordringer med funktionalitet af projektet på de forskellige PC'er, fandt vi dog en måde at løse projektet på. Der blev opretholdt en god stemning og kommunikation undervejs, både ved udvikling af ideer, løsninger på problematikker, fremmøde og sammensætning. Undervejs i processen fremgik det ofte, at gruppens medlemmer var et godt match. Der var flere gange gruppemedlemmer havde samme vision og tankegang og tilgang til projektet. Selvom der blev arbejdet individuelt hjemmefra, blev der ved fremmøde vist meget ens projekter, udtryk og udseende, både på front- og backend.

Det der kunne have været bedre i processen, er mere inddragelse af *Git* under processen. Vi fandt hurtigt ud af fordelene, hvis vi havde brugt mere tid på Git. Der kunne have været meget tid at spare, som kunne være brugt på flere implementeringer af krav.

Det vi har lært af denne opgave er, at det er vigtigt at få lagt et godt fundament. Få et godt udgangspunkt i form af planlægning af projekt, der giver et bedre overblik af processen. Dette gør at man er bedre gearet til uforudsete udfordringer, som kunne opstå.

F.eks. skulle vi have været bedre til at lægge en plan, eller evt. bruge et *kanban board* til at holde overblikket, deadlines eller ting som manglede at blive implementeret.

Vi har også lært, at funktionalitet er den vigtigste del af processen. Specielt når det omhandler kunder og hvilke krav, der stilles til dette. Udseende er noget, som kan implementeres på et senere tidspunkt. Hvad vi skal gøre anderledes næste gang, er at starte med backend og få funktionaliteten op og køre, inden der fokuseres for meget på frontend delen.