Cupcake Project

Olsker Cupcakes

Developers:

Toby Hartzberg

cph-th452@stud.ek.dk

https://github.com/Obliwobbi

Jesper Andersen

cph-ja472@stud.ek.dk

https://github.com/JesperTAndersen

Morten Jensen

cph-mj1228@stud.ek.dk

https://github.com/Mortenjenne

Daniel Hangaard

cph-dh258@stud.ek.dk

https://github.com/DHangaard

Videogennemgang af projektet:
https://youtu.be/a4y-N_2l_ko?si=6Er8Ix5Pw4QAjPMz

31. oktober 2025

Indholdsfortegnelse

Indledning

Denne rapport omhandler udviklingen af et webbaseret system for virksomheden *Olsker Cupcakes*. Formålet er at udvikle en prototype på en webshop, der gør det muligt for kunder at bestille cupcakes online samt for administratorer at håndtere ordrer og kunder.

Projektet er udarbejdet som en del af datamatikeruddannelsens 2. semester og anvender teknologier som Java, Javalin, Thymeleaf og PostgreSQL. Rapporten henvender sig til en fagfælle – en datamatikerstuderende på tilsvarende niveau – som ikke kender Olsker Cupcakes-projektet, men har kendskab til de anvendte teknologier og metoder.

Baggrund

Olsker Cupcakes er et lille, dybdeøkologisk bageri beliggende i Olsker på Bornholm, drevet af Jonas Møller og Emil Vang. Virksomheden fokuserer på håndlavede cupcakes af lokale og økologiske råvarer og har et stærkt fokus på bæredygtighed, kvalitet og fællesskab.

Kundens ønske er at få udviklet en digital platform, hvor kunder kan se sortimentet og bestille cupcakes online. Systemet skal give mulighed for, at kunden kan sammensætte sin egen cupcake ved at vælge bund og topping, samt betale og afhente bestillingen i butikken. Målet er at gøre det lettere for kunderne at handle og samtidig øge virksomhedens synlighed og salg.

Teknologi valg

Tech stack

Teknologiområde	**Navn / Version**
Version Control	Git
Programmeringssprog	Java
Java Development Kit	17
Integrated Development Environment	IntelliJ IDEA 2025.2.4
Build Tool	Apache Maven 3.10.1
Database	PostgreSQL 42.7.2

Teknologiområde	Navn / Version
Web Framework	Javalin 6.1.3
Template Engine	Thymeleaf 3.1.2
Frontend	HTML 5, CSS 3, JavaScript (ES6)
Testing Framework	JUnit 5.10.2

Krav

Vision

Olsker Cupcakes' vision med dette projekt er at etablere en online platform, hvor kunder nemt kan bestille cupcakes fra virksomhedens sortiment.
Formålet er at øge synligheden og tilgængeligheden af Olsker Cupcakes' produkter, styrke brandets tilstedeværelse digitalt og derved skabe grundlag for mersalg.
Systemet udgør første skridt i virksomhedens digitale transformation og skal fungere som fundament for fremtidige udvidelser — eksempelvis lagerstyring, kundeprofilering og dataanalyse samt en mobil app.

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).

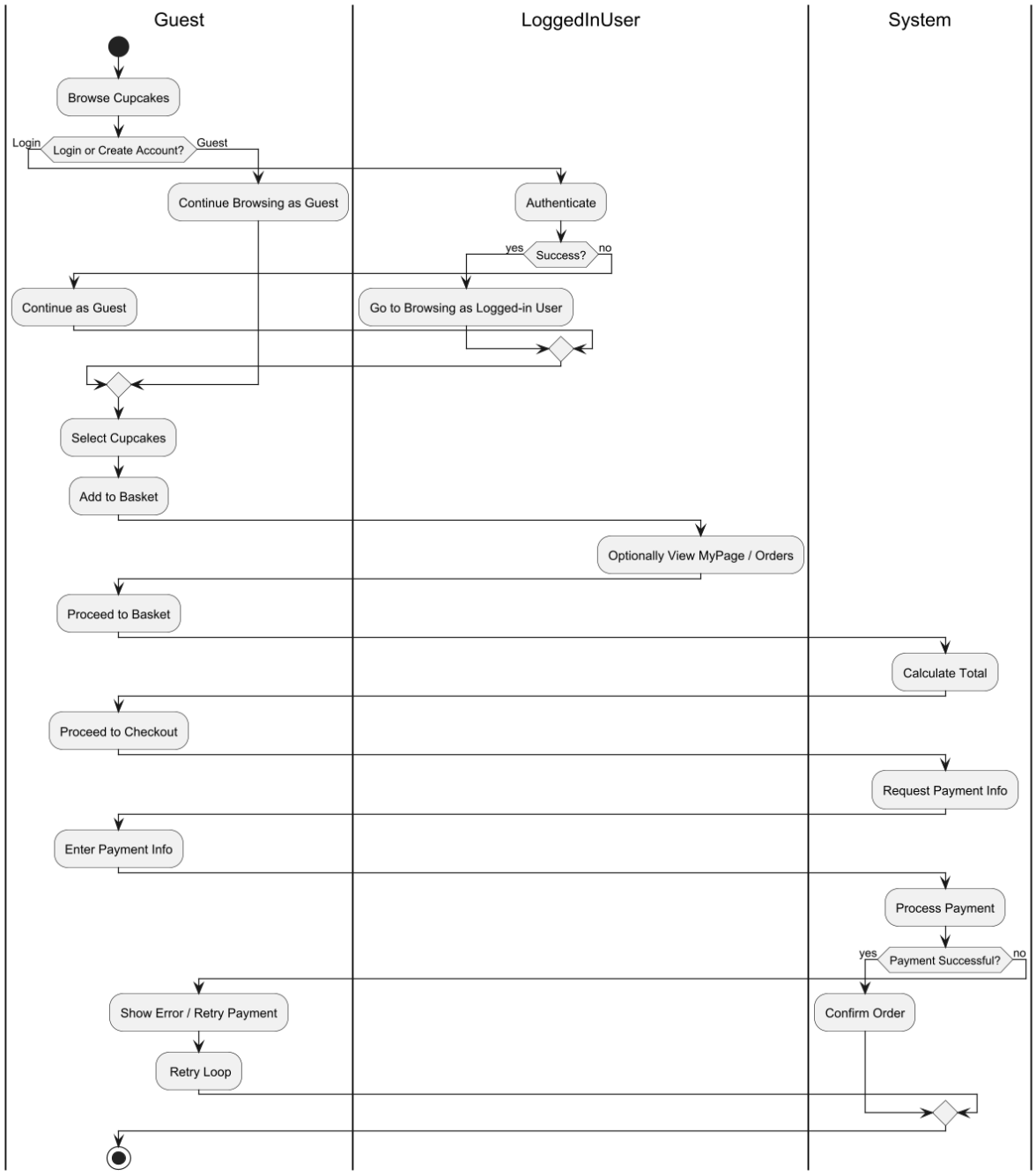
US-6 Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.

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.

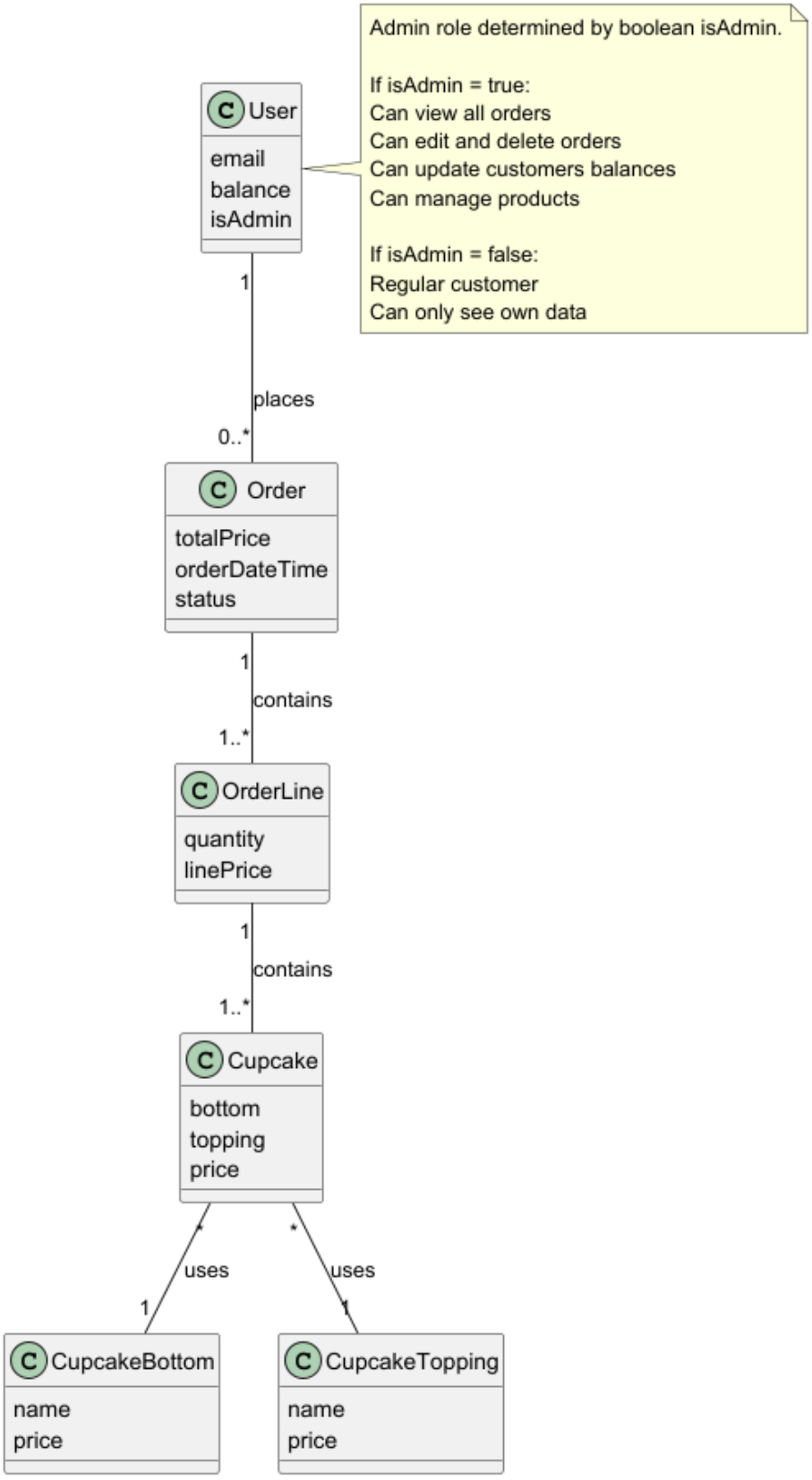
US-9 Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde ugyldige ordrer. F.eks. hvis kunden aldrig har betalt.

Aktivitetsdiagram



Domæne model og ER diagram

Domænemodel



Relationer og multiplicitet

I domænemodellen er der identificeret centrale relationer mellem entiteterne User, Order, OrderLine, Cupcake, CupcakeBottom og CupcakeTopping. Disse relationer danner grundlaget for systemets struktur og beskriver, hvordan objekternes sammenhæng.

User – Order
En User kan have placeret flere Orders, men hver Order tilhører præcis én User.
Dette afspejler, at en kunde kan have flere ordrer over tid, men at en ordre altid er knyttet til én specifik kunde.

Order – OrderLine
En Order består af mindst én OrderLine, men kan indeholde mange.
Hver OrderLine hører kun til én Order, da den beskriver en specifik cupcake inden for den pågældende ordre.

OrderLine – Cupcake
Hver OrderLine indeholder én Cupcake.
Dette betyder, at en ordrelinje kan repræsentere flere af den samme cupcake-type, men altid relateret til én konkret Cupcake-instans.

Cupcake – CupcakeBottom og CupcakeTopping
En Cupcake består altid af præcis én Bottom og én Topping.
Samtidig kan en Bottom eller Topping bruges i mange cupcakes.

Overvejelser om relationer

Den relation, der blev overvejet mest, var forbindelsen mellem OrderLine og Cupcake. I den tidlige fase blev det overvejet at lade OrderLine pege direkte på CupcakeBottom og CupcakeTopping, men i stedet blev der introduceret en separat Cupcake-klasse.

Fordelen ved dette design er, at prisen kan beregnes centralt i Cupcake-klassen, hvilket reducerer kompleksitet og dubleret logik i OrderLine. Samtidig gør det modellen mere fleksibel, da Cupcake senere kan udvides med yderligere egenskaber, såsom special toppings, rabatter eller drys uden at påvirke de eksisterende ordrerelationer.

ER diagram

Overvejelser bag ERD-modellen

Denne ERD-model er designet med fokus på normalisering, dataintegritet og fleksibilitet. Formålet er at skabe en database, der understøtter håndtering af ordrer, brugere og produkter på en effektiv og konsistent måde.

Normalisering og relationer

Databasen er struktureret efter principperne for tredje normalform. Tabellen users gemmer stamdata om brugere, mens orders er adskilt herfra for at muliggøre, at én bruger kan have flere ordrer. Tabellen orderlines gør det muligt at knytte flere toppings og bunde til en enkelt ordrelinje. Tabellerne toppings og bottoms fungerer som opslags-tabeller, der definerer de tilgængelige produkter. Endelig indeholder zip_codes information om postnumre og byer, hvilket sikrer, at disse data kun vedligeholdes ét sted og dermed undgår redundans.

Fremmednøgler

Der er etableret flere fremmednøgler for at bevare referentiel integritet:

- *users.zip_code* refererer til *zip_codes.zip_code*
- *orders.user_id* refererer til *users.user_id*
- *orderlines.order_id* refererer til *orders.order_id*
- *orderlines.topping_id* refererer til *toppings.topping_id*
- *orderlines.bottom_id* refererer til *bottoms.bottom_id*

Disse relationer sikrer, at der ikke kan indsættes data, som ikke relaterer sig til eksisterende poster, f.eks. at en ordre altid tilhører en gyldig bruger.

Datatyper

Primærnøgler anvender datatypen serial for automatisk nummerering. Tekstfelter er defineret som character varying for fleksibilitet, mens timestamp with time zone bruges til datofelter for at sikre korrekt håndtering af tidszoner. Felter som paid, admin og is_guest er boolean, da de repræsenterer binære værdier. Priser er angivet som double precision for at muliggøre beregninger med decimaler.

Constraints

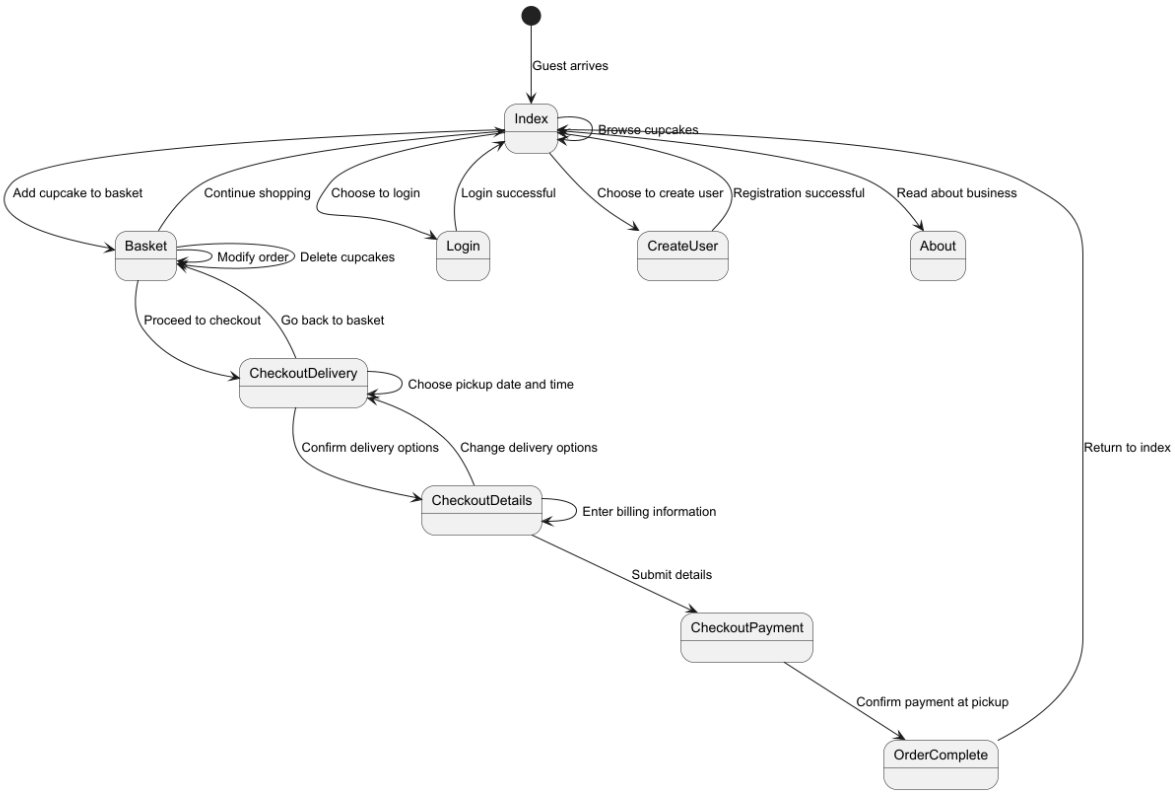
Modellen bygger på anvendelsen af Primary Key og Foreign Key constraints for at sikre unikke og konsistente relationer. Derudover kan felter som email i users med fordel have en UNIQUE constraint for at forhindre dubletter. NOT NULL constraints bør anvendes på felter, der er nødvendige for forretningslogikken, eksempelvis firstname, order_date og price_total.

Forretningslogiske overvejelser

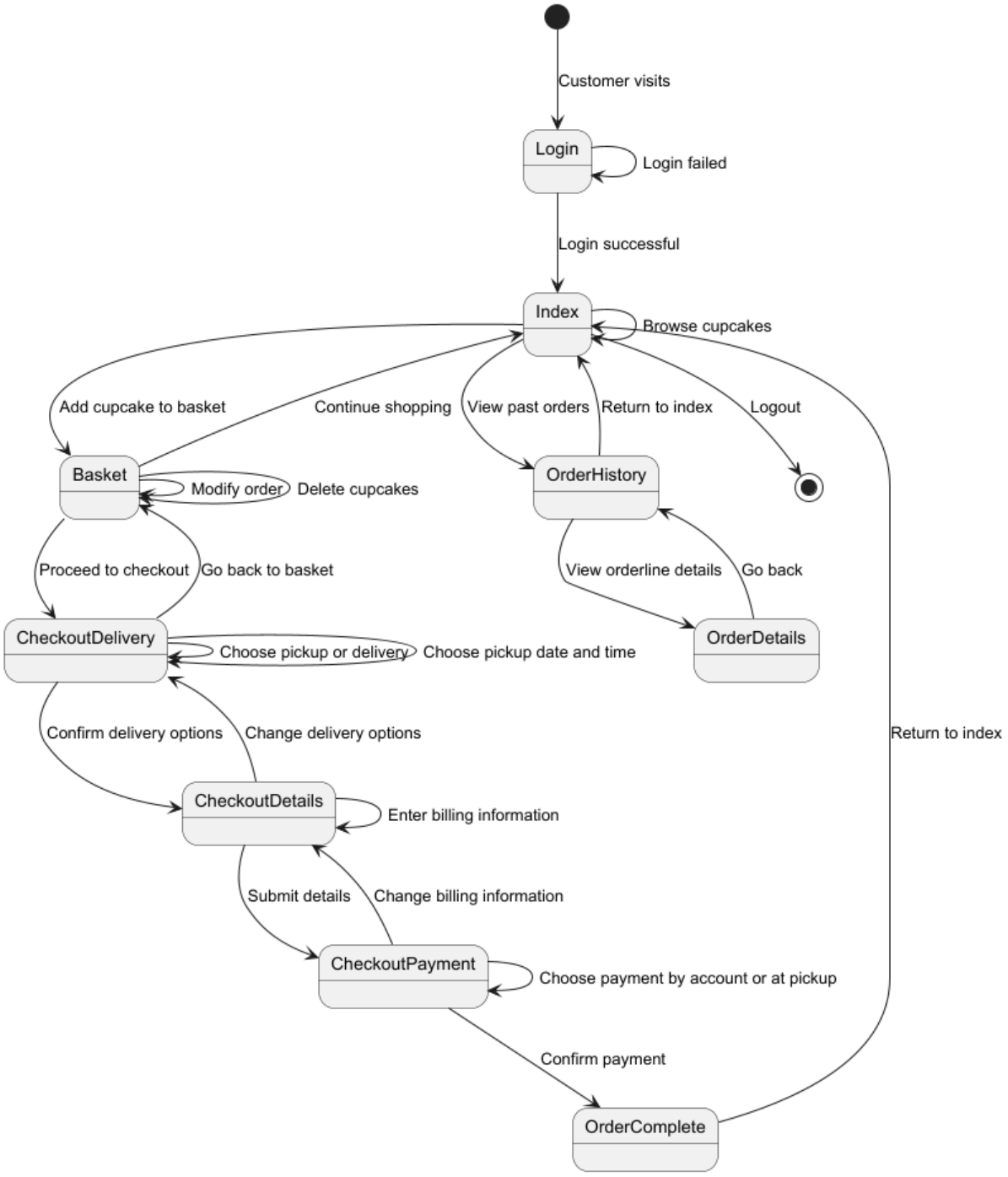
Adskillelsen af orders og orderlines giver mulighed for flere produkter pr. ordre samt individuel prisberegning pr. linje. Felterne price_total og orderline_price anvendes for at gemme beregnede værdier direkte i databasen, hvilket forbedrer performance. Felterne is_guest og admin gør det muligt at håndtere både gæstekunder og administratorer i samme system.

Navigationsdiagram

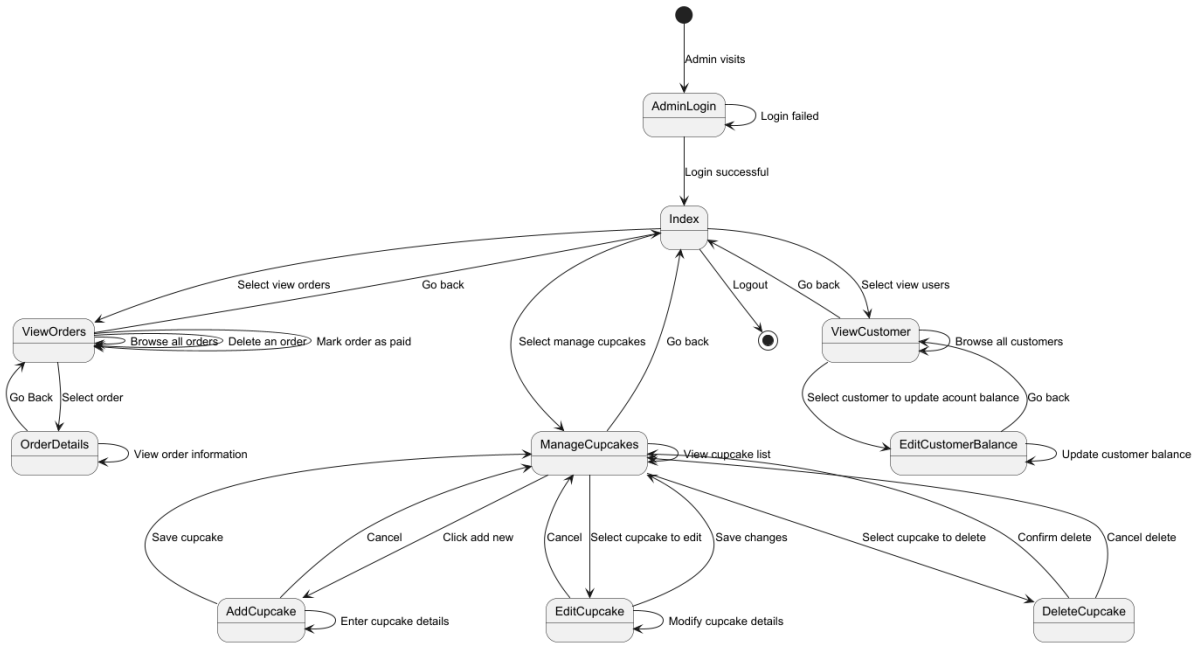
Guest



Customer


Customer visits
Login
Login failed
Login successful
Index
Browse cupcakes
Add cupcake to basket
Continue shopping
View past orders
Return to index
Logout
Basket
Modify order
Delete cupcakes
OrderHistory
Proceed to checkout
Go back to basket
View orderline details
Go back
CheckoutDelivery
Choose pickup or delivery
Choose pickup date and time
OrderDetails
Confirm delivery options
Change delivery options
Return to index
CheckoutDetails
Enter billing information
Submit details
Change billing information
CheckoutPayment
Choose payment by account or at pickup
Confirm payment
OrderComplete

Admin



Særlige forhold

Hvilke informationer gemmes i session

I Session gemmes en bruger (CurrentUser) såfremt at brugeren er logget ind. Ligeledes gemmes en ”cart” Der indeholder den aktive bestilling.

Hvordan håndterer man exceptions.

Exceptions optræder generelt som DatabaseExceptions, samt SQLexceptions i Mapper klasserne. Dette er for at fange databasefejl så tidligt som muligt.

Der er ligeledes anvendt IllegalArgumentExceptions som et valideringslag i Service klasserne

Hvordan man har valgt at lave validering af brugerinput.

Validering af brugerinput sker på flere niveauer. I front end sker valideringen i HTML, hvor der bl.a. er anvendt ”required” i inputfelter.
I back end sker valideringen i Controller- samt Service klasser, såsom UserController UserServiceImpl der holder metoder til validering af eksempelvis e-mail, postnummer og password – Hvis brugeren laver fejl bliver dette håndteret via Exceptions, hvor en passende besked bliver vist til brugeren i viewet. I HTML er der på denne måde anvendt error- og success messages til at guide brugeren igennem sitet.

Hvordan man har valgt at lave sikkerhed i forbindelse med login

Som sikkerhed ved login er der anvendt BCrypt til at hashe passwords. Ligeledes er der lavet et krav om indhold og længde af en brugers passwords. Dette valideres i UserController.

Hvilke brugertyper, der er valgt i databasen, og hvordan de er brugt i JDBC

I databasen er der anvendt en type af bruger: en User. Denne entitet kan, ved hjælp af boolske attributter, optræde som enten ”Guest”, ”Customer” eller ”Admin”. I UserMapper har vi derfor valgt at lave både en createUser- og en createGuestUser metode, hvor vi dermed kan håndtere staten af en users is_guest boolean. Guest er indsat i datalaget for at kunne give en bruger mulighed for at købe fra applikationen uden at være logget ind. En Admin kan kun sættes som Admin direkte i databasen.

Status på implementation

- Alle user stories er mødt og implementeret i koden.
- Alle crud metoder er implementeret i de forskellige mappers
- Der er lavet integrationstest på tre ud af fem mappers – Alle tests er grønne på afleveringstidspunktet.
- Alle websider er stylet. Siderne er også mobil venlige - undtagen Admin’s se alle ordre og kunder i systemet.
- Programmet indeholder et servicelag samt diverse klasser til at håndtere statistik (såsom total revenue, monthly revenue og average revenue) – Dette er ikke implementeret i views samt controllers ved afleveringstidspunktet.

Proces

- Hvad var jeres planer for teamets arbejdsform og projektforløbet?

Planen for teamet var at deles i to og to for her at anvende pair programming. Til dette skulle der være en tydelig opgavefordeling hvor teamet løbende ville anvende sig af GitHub Projects til at skabe et overblik over projektets fremgang. For at dette skulle kunne fungere, skulle der løbende afholdes code reviews på tværs af de to par hvilket blev løst ved at anvende pull requests på GitHub.

- Hvordan kom det til at forløbe i praksis?

Som udgangspunkt har planen fungeret som forventet. Der har igennem hele projektet været en fælles forståelse for målet. Der er løbende blevet opdateret på GitHub Projects samt der er løbende blevet afholdt code reviews i forbindelse med pull requests.

- Hvad gik godt og hvad kunne have været bedre?

- Kommunikationen har ikke været optimal. Det ene par har arbejdet remote og det andet har arbejdet på skolen. Det har gjort at det til tider har føltes som om der blev arbejdet på to individuelle projekter. Begge par har siddet med denne følelse.

+ Der blev fra starten skabt en solid ide og fællesforståelse af selve projektet og dets end-state. Dette har givet en god arbejdsfordeling og det har været nemt at holde fokus på opgaven og minimere konflikter ved eksempelvis merge og implementering.

- Hvad har I lært af processen og hvad vil I evt. gøre anderledes næste gang?

I forbindelse med processen har vi lært at der er fordele ved forskellige arbejdsformer – dette skal tilgodeses og der skal arbejdes på en endnu bedre kommunikation. Dette kan løses ved eksempelvis ikke at arbejde i faste par gennem et helt projekt, men skifte partnere ved forskellige punkter i processen som f.eks. ved en ny feature.