Cupcake Projekt

Navn	Studie-mail	GitHub
Lasse Timm Hansen	cph-lh546@cphbusiness.dk	Maximitas
Johan Nikolaj Poulsen	cph-jp471@cphbusiness.dk	ash2213
Asim Kadir Kilic	cph-ak576@cphbusiness.dk	AerrowV
Zana Baran	cph-zb40@cphbusiness.dk	ShadowB0X

Klasse A: Gruppe G

Projektfase: 21. oktober 2024 – 31. oktober 2024

Udarbejdelse af Rapport: 28. oktober 2024 – 31. oktober 2024

https://github.com/ash2213/Cupcake_Project

Video link: https://youtu.be/Lzuu4NqFxpw

Baggrund

Projektet omhandler en virksomhed ved navn Olsker Cupcakes. en virksomhed som har beliggenhed på Bornholm og som gerne vil sælge deres cupcakes til resten af øen ved hjælp af en hjemmeside.

Kunden skal kunne bestille en cupcake, hvor de selv vælger en bund og en topping ud fra butikkens valgmuligheder. Denne cupcake skal kunne købes på hjemmesiden, og derefter skal man kunne se på sin ordre, og hvornår den er klar til afhentning.

Dernæst skal man kunne oprette en profil og logge ind. Når man logger ind på sin profil, skal man kunne se sine ordrer, også dem man har lavet på en anden session.

Til sidst skal det være muligt at logge ind på en admin konto, der kan se alle ordrer og redigere dem som ‘in progress’ eller ‘completed’, og at kunne fjerne færdige ordrer.

Teknologivalg

- JDBC
- Postgresql ver. 16.2
- Javalin ver. 6.1.3
- IntelliJ IDEA 2024.2.4 ultimate edition
- Java 17.0.13 Amazon Corretto
- Docker Desktop
- pgAdmin 4
- HTML
- CSS
- Thymeleaf
- PlantUML

Olskers krav

Olskers vision for systemet er at udvide deres kundebase ved at tilbyde kunderne mulighed for at bestille cupcakes online og hente dem direkte i butikken.

Hjemmesiden skal give mulighed for, at man kan bestille cupcakes hjemmefra eller hvor som helst via en enhed med internetadgang og få adgang til hele udvalget af kombinationer. Derudover skal man kunne se, hvornår ordren er klar, samt få et overblik over bestillingen og den samlede pris.

Dette system vil også kunne reducere spild, da kunderne bestiller præcis de cupcakes, de ønsker, online, i stedet for at risikere, at cupcakes må kasseres i butikken på grund af overproduktion.

User-stories

US-1:
Som kunde kan jeg bestille og betale cupcakes med en valgfri bund og topping, 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 et 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, og dermed se den samlede pris.

US-5:
Som kunde eller administrator kan jeg logge på systemet med e-mail og password. Når jeg er logget på, skal jeg kunne se min e-mail på hver webside (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, så 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.

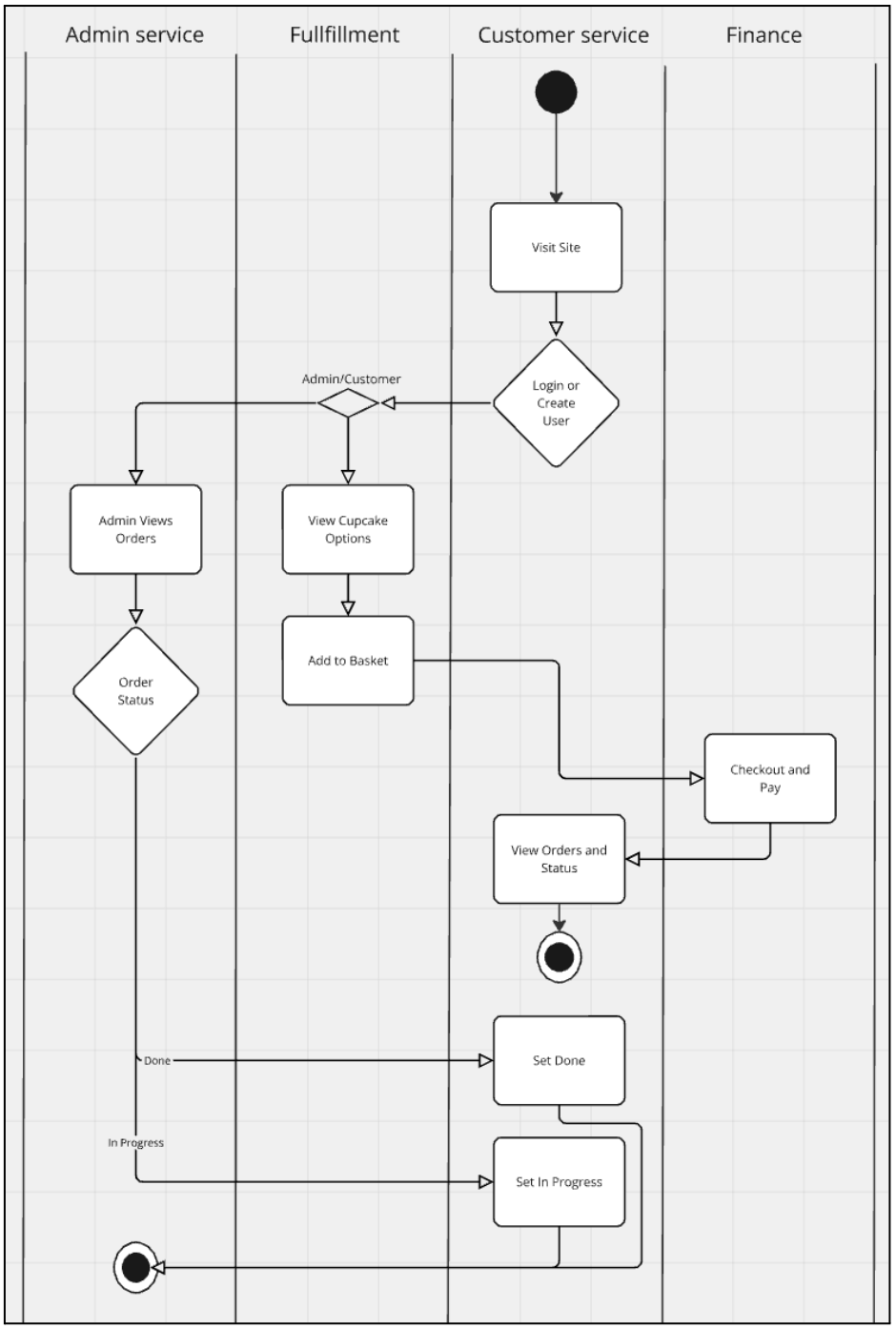
Aktivitetsdiagram

Aktivitetsdiagrammet består af fire swimlanes hvor vi starter i customer service. Når man kommer ind på hjemmesiden, kan man lave en konto eller hvis man allerede har en, logge ind.

Hvis der bliver logget ind med en admin konto, vil man være i stand til at se alle ordrer der er blevet lavet på hjemmesiden. Administratoren vil også være i stand til at sætte kundernes ordrer som at være i gang eller at være færdige.

Når man logger ind som en customer/kunde, kan man se mulighederne for at bygge

en cupcake. Den cupcake man har bygget kan dernæst lægges i sin kurv. Når cupcaken er lavet og puttet i sin kurv, vil man kunne se sin kurv med sit valg. Den samlede pris på sine valg vil blive vist og man kan nu betale for sine cupcakes. Efter betaling vil der være en kvittering på din ordre og en statusbesked på ordren.

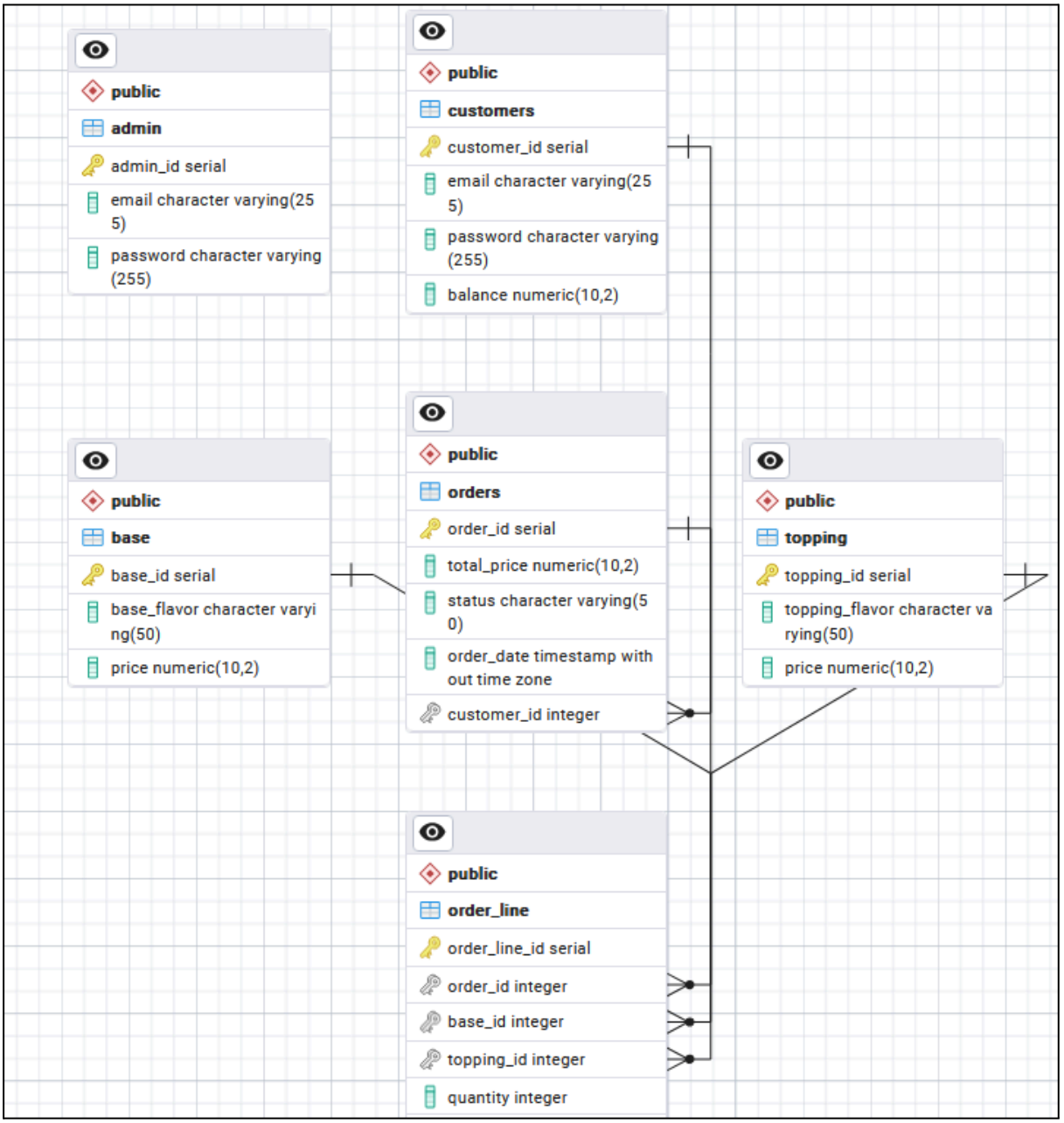


Aktivitetsdiagrams sidste udgave.

ER-diagram

Udgangspunkt for ER-diagram

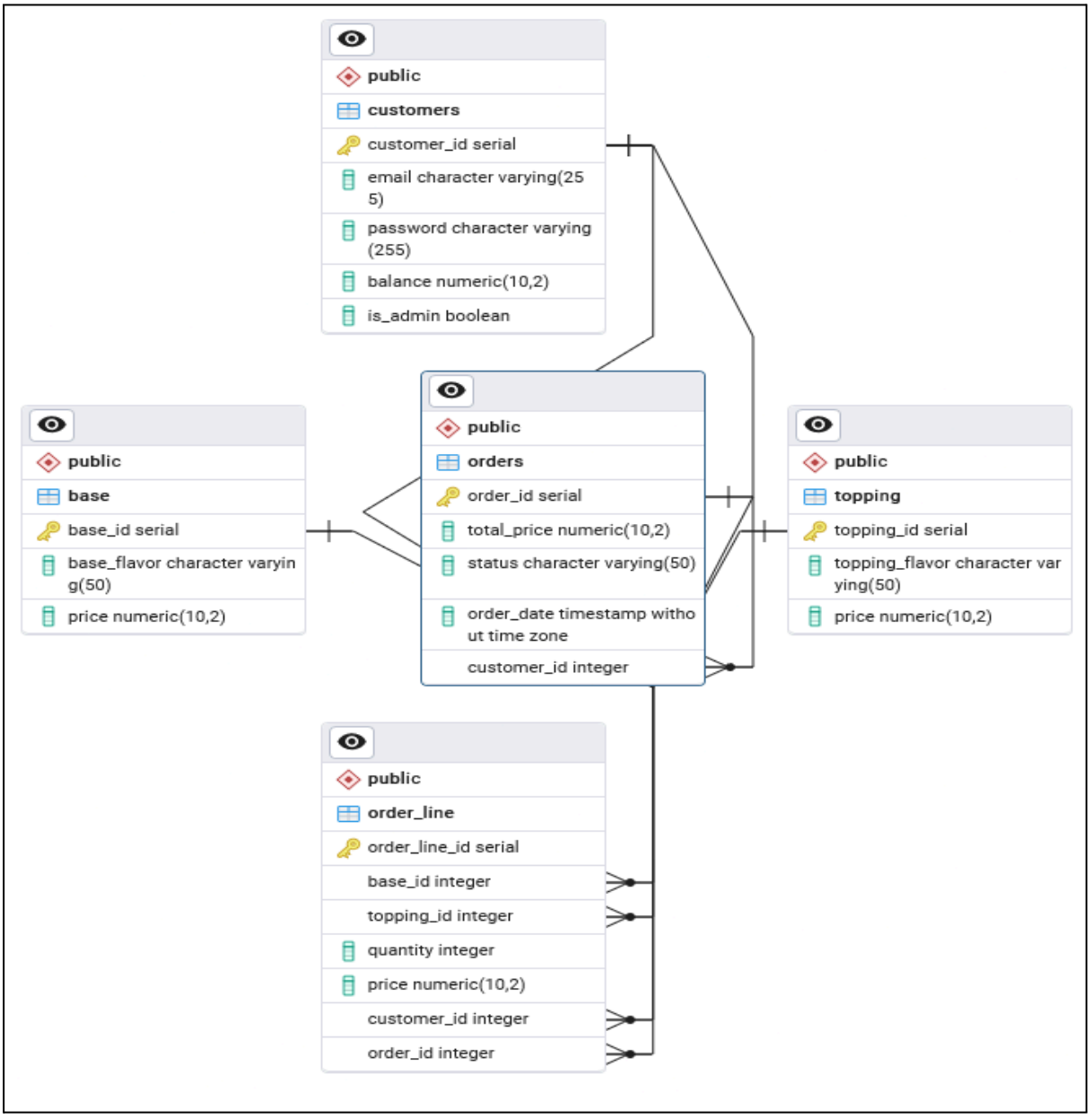
I vores start diagram havde vi oprindeligt en separat admin-tabel, men i løbet af processen fandt vi ud af, at vi i stedet kunne tilføje en *boolean* (is_admin) til vores customers-tabel. På den måde kan vi tjekke, om en kunde er admin, og i vores loginmetode vise en administrator side, hvis kunden har administratorrettigheder.



ER-diagrams første udgave.

Slutpunkt for ER-diagram

I vores endelige diagram ser systemet lidt anderledes ud. Vi har nu fjernet admin-tabellen, da den ikke blev anvendt. I stedet har vi tilføjet en is_admin boolean til at tjekke, om kunden, der logger ind, er admin.



ER-diagrams sidste udgave.

Om ER-diagrammet

- **Normalisering og 3. normalform**
    - Diagrammet er designet til at overholde 3. normalform, idet hver tabel har en primærnøgle, og alle kolonner i tabellerne afhænger direkte af denne uden indirekte afhængigheder. Dette sikrer, at dataene ikke er redundante, hvilket gør databasen lettere at vedligeholde.

- **en-til-mange relationer**
    - customers og orders: Der er en en-til-mange relation mellem customers og orders, hvor en kunde (customer_id) kan have flere ordrer, men hver ordre tilhører en enkelt kunde. Denne struktur forhindrer gentagelse af kundeoplysninger og knytter dem i stedet til flere ordrer.
    - customers og order_line: Der er også en en-til-mange relation mellem customers og order_line, hvilket betyder, at en kunde kan have flere ordrelinjer. Dette gør det muligt at se, hvilke specifikke produkter eller cupcakes en kunde har bestilt i en enkelt ordre.
    - orders og order_line: Her er der en en-til-mange relation, hvor en ordre kan indeholde flere linjer med forskellige kombinationer af bund (base) og topping. Dette gør det muligt for en ordre at bestå af forskellige produkter.
    - base og order_line samt topping og order_line: Base og topping har en en-til-mange relation til order_line, hvilket gør det muligt at anvende samme base og topping i flere ordrelinjer.

- **Begrænsninger med fremmednøgler**
    - Alle tabeller anvender automatisk genererede nøgler som primærnøgler (serial-type), hvilket sikrer unikke ID'er uden manuel indtastning. Vores fremmednøgler sikrer, at relationerne mellem tabellerne opretholdes, forhindrer forældreløse data og sikrer konsistens i databasen.

Navigationsdiagram

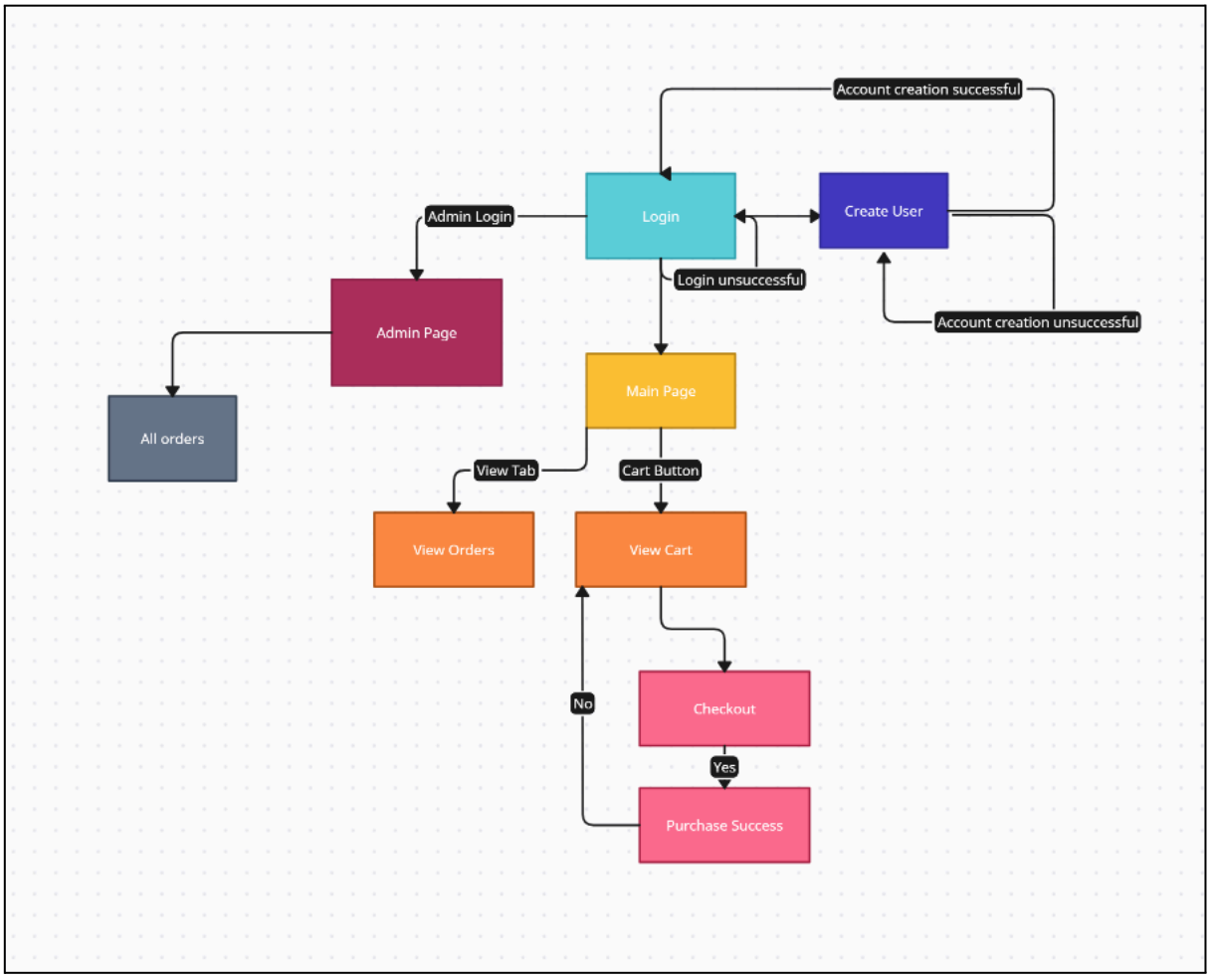
- **Login og brugeroprettelse:**
    - Brugere starter på login-siden.
    - Hvis en bruger ikke har en konto, kan de oprette en ved at vælge create user-siden.
    - Ved succesfuld kontooprettelse kan brugeren vende tilbage til login-siden, mens en mislykket oprettelse holder brugeren på create user-siden.
    - Efter et mislykket login forbliver brugeren på login-siden, mens et succesfuldt login sender dem videre til homepage.

- **Hovedside (Homepage):**
    - Homepage fungerer som det centrale navigationspunkt, hvor brugere kan vælge mellem at se deres bestillinger (view orders) eller kurvens indhold (view cart).
    - Administratorer har også mulighed for at tilgå admin page fra homepage.

- **Administrator-sider:**
    - Administratorer kan logge ind via admin login for at få adgang til admin page.
    - Fra admin page kan du se alle ordrer (all orders), hvilket giver en samlet oversigt over alle ordrer i systemet.

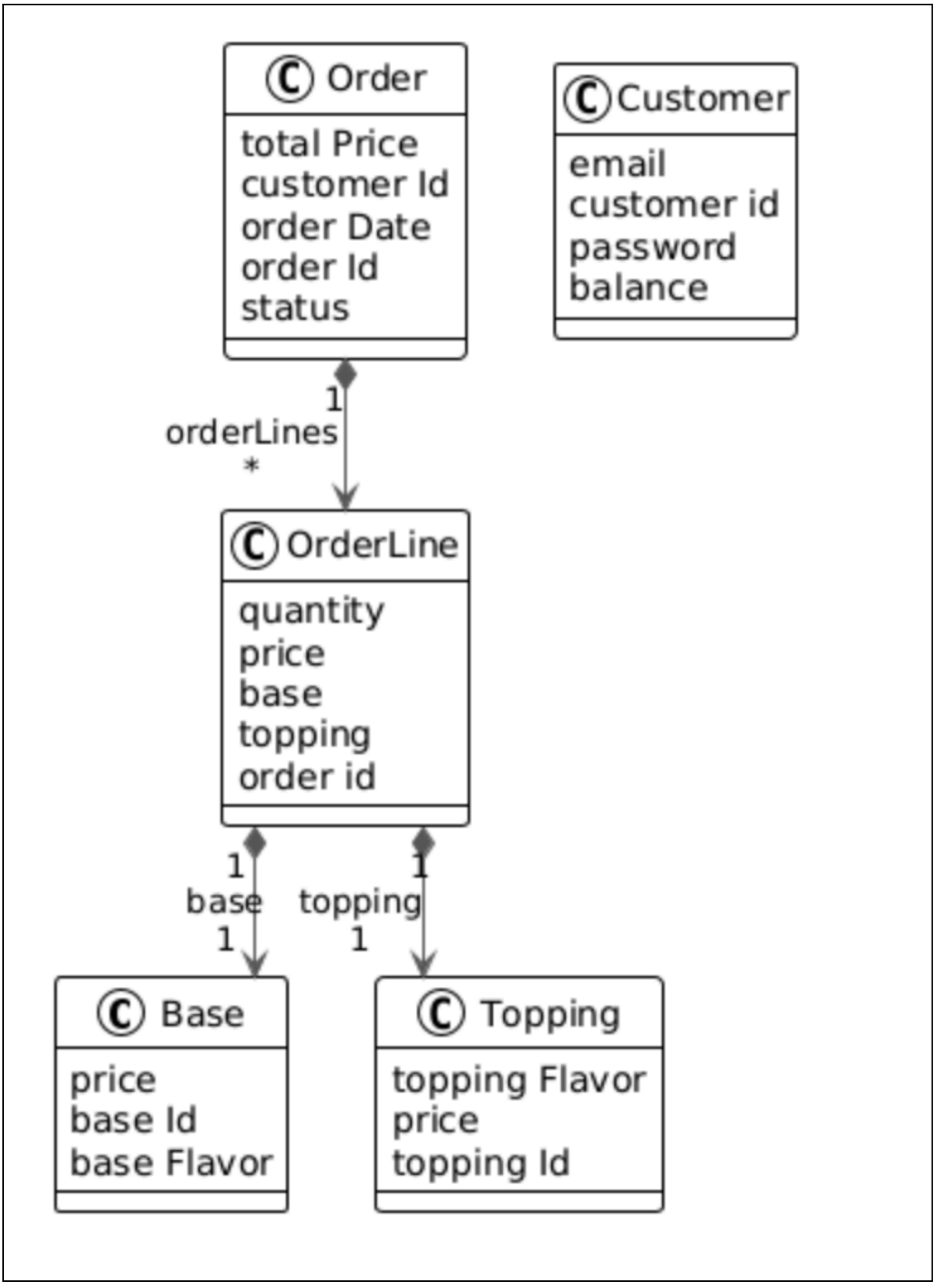
- **Kurv og betaling:**
    - Brugere kan fra homepage klikke på cart button for at se indholdet af deres kurv (view cart).
    - Hvis de vælger at gå til kassen, bliver de sendt til checkout.
    - Ved et succesfuldt køb lander de på purchase success-siden, som bekræfter, at ordren er afgivet.

- **Se bestillinger (View orders):**
    - Fra homepage kan brugeren vælge at gå til view orders, hvor de kan se en oversigt over deres tidligere ordrer.



Navigationsdiagram.

Domænemodel



Domænemodel.

Dette billede viser et domæne-diagram for et kunde bestillingssystem. Diagrammet illustrerer klasserne og deres relationer, der repræsenterer forretningslogikken i kundens ordre proces. Diagrammets struktur afspejler, hvordan data omkring ordrer, produkter og kunder håndteres i systemet.

- **Customer:**
  Klassen repræsenterer kunden.

- **Order:**
  Denne klasse repræsenterer en specifik ordre foretaget af en kunde. Order har en komposition i forhold til OrderLine, hvilket betyder, at en ordre består af en eller flere ordrelinjer.

- **OrderLine:**
  Denne klasse repræsenterer en enkelt linje i en ordre, der kan være et produkt med en specifik base og topping.
    - OrderLine har en 1:* relation til Order, hvilket betyder, at en ordre kan have flere ordrelinjer.

- **Base:**
  Denne klasse repræsenterer den grundlæggende base i et produkt, som kunden kan vælge.
    - OrderLine refererer til Base med en 1:1 forbindelse, hvilket betyder, at hver ordrelinje har præcis én base.

- **Topping:**
  Denne klasse repræsenterer toppings, der kan tilføjes til produktet.
    - OrderLine refererer også til Topping med en 1:1 forbindelse, hvilket betyder, at hver ordrelinje kan have én topping.

Relationer og Sammenhæng:

- Composition: Order har en komposition relation til OrderLine, hvilket betyder, at når en ordre slettes, bliver alle tilknyttede ordrelinjer også slettet.

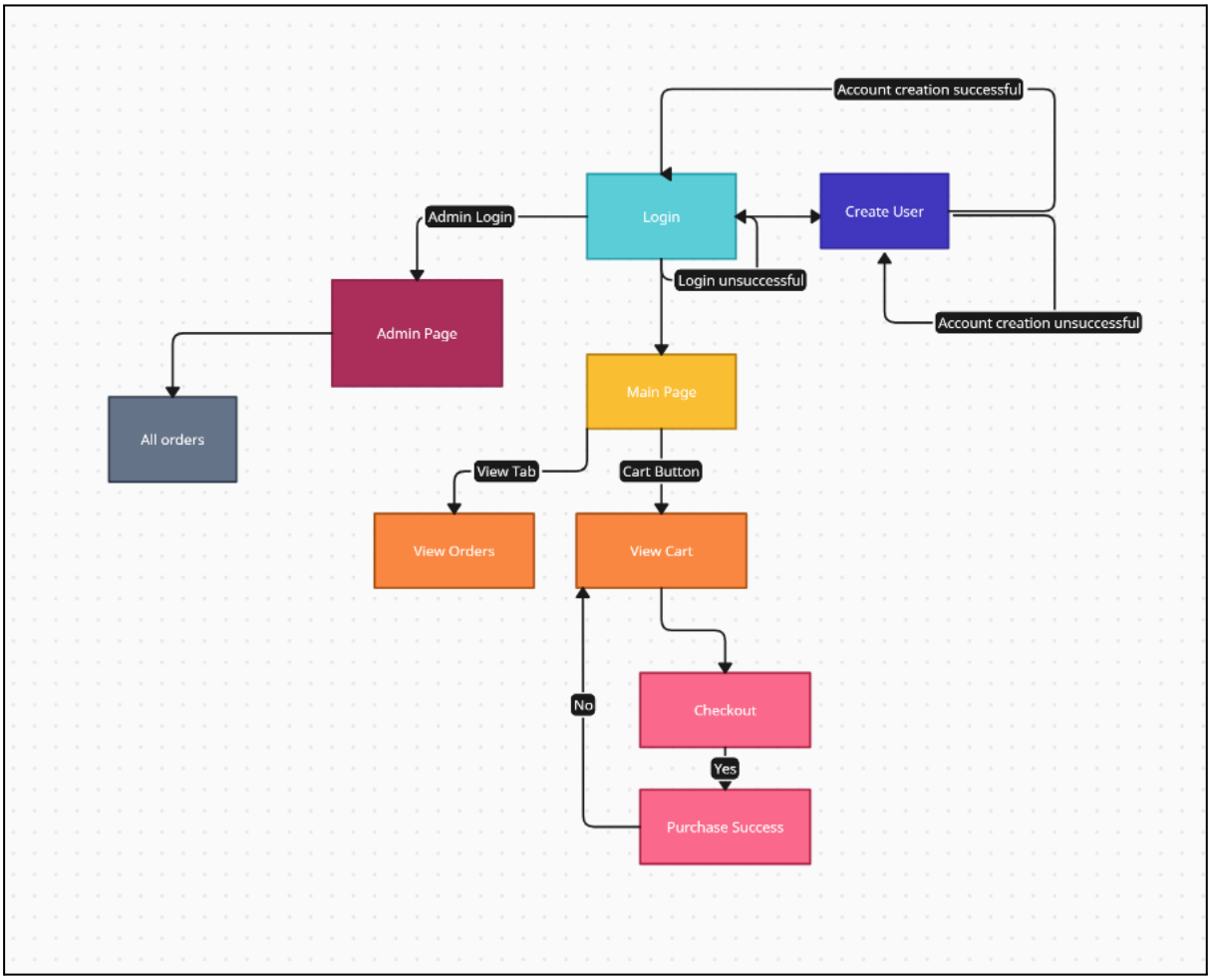
- Aggregation: OrderLine afhænger af både Base og Topping, da disse klasser definerer produktets smag og pris.

Dette domæne-diagram giver et overblik over, hvordan forskellige elementer (kunde, ordre, base og topping) hænger sammen og interagerer i systemet. Det sikrer, at kundens bestillingsproces er struktureret og gør det muligt at gemme detaljer om hver ordre, inklusive kundens valg af base og topping samt den samlede pris for ordren.

Flow og brugeroplevelse

Vores navigationsdiagram præsenterer et klart og struktureret brugerflow, hvor brugerne starter med login og derefter navigerer til hovedsiden. Herfra kan de administrere deres bestillinger, se kurvens indhold og gennemføre købet. Administratorer har desuden adgang til at se alle ordrer i systemet.

Diagrammet gør navigationen let at forstå og sikrer, at alle brugerscenarier er tilgængelige gennem logiske veje.



Særlige forhold

- **Hvilke informationer gemmes i session:**
    - Vi gemmer kundens e-mail for at vise den i hjørnet af hjemmesiden, så brugeren kan se, at de er logget ind på deres profil. Derudover gemmer vi den samlede pris i kurven, så brugeren kan se den totale pris for alle cupcakes i kurven.

- **Hvordan håndterer man exceptions. Det kommer vi tilbage til senere i semesteret:**
    - Vi implementerer en exception handler, som håndterer fejlmeddelelser ved at sende en brugervenlig besked i stedet for tekniske fejlmeddelelser som "Error 404," som brugeren ikke kan bruge til noget.

- **Hvordan man har valgt at lave validering af brugerinput:**
    - Ved login kræves det, at E-mail og Password-felterne udfyldes med en e-mail og et password, der findes i databasen, og e-mailen skal indeholde “@”.
    - Ved registrering af en konto skal e-mailen igen indeholde “@”. Der er to felter til password, som skal matche. Alle felter skal være udfyldt, og e-mailen skal være unik i databasen.
    - Ved oprettelse af en cupcake skal brugeren vælge en option i drop-down boksen for både topping og bund. Hvis der ikke vælges noget, kan man ikke gå videre.

- **Hvordan man har valgt at lave sikkerhed i forbindelse med login:**
    - Vi benytter *POST* frem for *GET* for at sikre, at brugernavn (e-mail) og password ikke vises i URL’en på den næste side. E-mailen er desuden sat til at være en unik nøgle i databasen, så hver e-mail kun kan bruges én gang.

- **Hvilke brugertyper, der er valgt i databasen, og hvordan de er brugt i jdbc:**
    - Vi har kunder, der opretter en konto og bliver brugere. En bruger kan i databasen ændres til at være en admin-konto, hvilket giver adgang til ekstra funktioner, når de logger ind som administrator.

Status på implementation

- **Man har ikke nået at lave alle de websider man har med i navigations diagrammet:**
    - Vi har opnået at skabe alle de ønskede websider i overensstemmelse med navigations diagrammet.

- **Man har ikke nået at lave alle CRUD metoderne til alle tabellerne:**
    - Vi har ikke implementeret fulde CRUD-metoder til alle tabeller, men har i stedet fokuseret på de basale funktioner, som vores program kræver.

- **Man har fundet en fejl "i sidste øjeblik", men har ikke haft tid til at rette det. - F.eks. at man har brugt session forkert, sådan at man på en af siderne kan komme ind uden at være logget ind:**
    - Vi har ikke oplevet nogen fejl i sidste øjeblik, såsom forkert session håndtering, hvor man eksempelvis kunne tilgå sider uden at være logget ind.

- **Øvrige mangler:**
    - Vi har ikke færdiggjort User Stories 7 og 9.

Proces

- **Hvad var jeres planer for teamets arbejdsform og projektforløbet?**
    - Vores plan var at fordele user stories mellem gruppens medlemmer, så hvert medlem kunne arbejde på sin egen branch. Vi ønskede at bruge pull-requests så meget som muligt for at integrere arbejdet løbende og

sikre samarbejde.

- **Hvordan kom det til at forløbe i praksis?**
    - Resultatet blev, at der var for mange user stories, der overlappede hinanden, så vi endte med at arbejde i grupper af to. Herefter fordelte vi de individuelle opgaver, så hver person kunne finpudse sin del selvstændigt.

- **Hvad gik godt og hvad kunne have været bedre?**
    - Arbejdsmoralen og samarbejdsforholdet var godt. Vi kunne dog have haft større fordel af pull-requests i forhold til vores arbejdsproces og brugt dem som en måde at bede om hjælp og feedback på.

- **Hvad har I lært af processen og hvad vil I evt. gøre anderledes næste gang?**
    - Det ville have været smart at lave integrationstest for at kunne teste koden hurtigere og mere effektivt. Vi kunne også have brugt mere tid på at nedbryde vores user stories i mindre opgaver, i stedet for den arbejdsform, vi endte med at anvende.