Cupcake Team-ICE (Gruppe F)
Icy cupcakes fra Bornholm

Gruppemedlemmer:

Frederik Edvardsen :
E-mail: cph-fe72@stud.ek.dk GithHub:Freddyboi123

Daniel Halawi:
E-mail: cph-dh256@stud.ek.dk GithHub: DannyShayH

Emil Xavier Verdet Thorsen:
E-mail: cph-et81@stud.ek.dk GithHub: EmilXavierVT

Luke Burup Persson:
E-mail: cph-lp373@stud.ek.dk GithHub: Luke-dkk

Link til selve projektet: Gruppe F cupcakes.

Link til YouTube video af projektet: Youtube.

Anslag: (uden billeder) (med billeder):

Afleveringsdato: 31/10 2025

Indhold

Indledning

Vi vil som fire datamatiker-studerende på andet semester lave et projekt baseret på: CSS, HTML, Java, Javalin, JavaScript, Sql og Thymeleaf, som skal simulere en en ordre platform for den fiktive kunde Olsker Cupcakes, der sælger cupcakes. Dette vil vi gøre baseret på hvad vi har lært på andet semester. Med dette menes *databaser, Connectionpools, controllers, mappers* med mere.

Baggrund

I denne opgave, tager vi afsæt i en butik der har beliggenhed på Bornholm. Butikken er en cupcake-forretning, hvor der online skal kunne bestilles forskellige slags cupcakes. Det vil sige: *"kunden skal kunne bestille en cupcake hvor man kan vælge både bund og top."* (ek, 2025).

For at en kunde dog skal kunne dette, skal der være et login, hvorved bestillingen kan ses og gemmes - dertil skal prisen også gerne vises.

Ydermere skal der være et administrator-login, hvor administratorerne kan se forskellige lister med bestilte cupcakes, samt til hvornår skal det kunne laves til.

Under administrator-login, skal administratorerne kunne sætte penge ind på kundernes konto, således kunderne kan betale.

Til sidst skal man som administrator kunne slette kunders bestillinger.

Alt dette er uddybet i bilag 1 - usercases, hvor de specifikke kundekrav står.

Metode-valg

Didaktiske overvejelser

For at skabe en praktisk ramme for vores projekt, har vi valgt at bruge en didaktisk relationsmodel. I denne opgave, har vi tænkt os at tage udgangspunkt i den ældre SMTTE-model – dette står for Sammenhæng, Mål, Tegn, Tiltag, Evaluering.

De respektive emner står beskrevet i deres eget underkapitel

Sammenhæng:

Står for hvor mange mennesker, og hvilke redskaber vi vil bruge for at opnå vores mål.

I dette tilfælde er vi fire full-stack programmører, med hver vores respektive computere.

Vi har Figma, som vi bruger til design-layout, samtidig bruger vi lucid-chart til at beskrive hvordan vores hjemmeside skal interagere.

Mål:

Heri, defineres hvad vores mål er, samt de respektive delmål.

Det er defineret således at vi kan måle hvor langt vi er kommet i processen.

Vores mål er defineret ud fra vores User-Stories. (Se bilag 1 - User-Stories)

Ud fra dem alle, definerer vi dem med tre delmål.

- Minimumskrav
- Funktionelle krav
- Færdige krav.

Minimumskravene: Kravet for at det som minimum fungerer.

Funktionelle krav: Det lykkedes med det som rekvisitøren ønsker.

Færdige krav: Det som rekvisitøren ønsker, er udfærdiget, vores kode er tilpasset vores coding-conduct (se bilag 2 - coding-conduct), koden er moduleret, samt vores produkt kan overlades til kunden.

For at se de færdigarbejdede User Stories, se bilag 3 - User Stories.

Tegn:

Er den subjektivt visuelle repræsentation der indikerer om delmålene opnås, og en indikator for hvor langt man kommer mod målene.

Vores kode skal som udgangspunkt give en visuel feedback, for hvorledes vores mål opnås. Disse kan betragtes gennem gestaltlovene, som er som følge:

- Loven om nærhed. (Multimededesigneren, 2022)
- Loven om lighed. (Multimededesigneren, 2022)
- Loven om lukkethed. (Multimededesigneren, 2022)
- Loven om forbundethed. (Multimededesigneren, 2022)
- Loven om figur og forbundethed. (Multimededesigneren, 2022)

Derudover med udgangspunkt i Isabella Froes (2017) afhandling omhandlende digital literacy, tager vi udgangspunkt i *apply and transfer*.

Dette gør at hvis man har en forforståelse af et digitalt princip, som for eksempel hvordan en hjemmeside fungerer, vil man kunne forflytte det over til noget andet der i princippet er det samme, via den indre fortolkning af det ikonografiske domæne. (Froes, 2017)

Tiltag

Tiltag er de aktioner man skal tage for at sikre sig at vi kommer i mål med vores respektive delmål. Her har vi defineret tre parametre, som skal sikre vores mål bliver opnået.

- Normalformer
- Gruppeaftaler
- arbejdsfordeling

For at forstå hvordan vores normalformer er et tiltag til at vi kan opnå vores mål, er det vigtigt at kende dem:

- Første normalform: ”
  1. *Der skal være en nøgle, der entydigt identificerer den enkelte række i tabellen.*
  2. *De enkelte felter må kun indeholde en værdi (atomare værdier).*
  3. *Der må ikke være kolonner, der gentages.”* (CphBusiness, 2023, s. 1).
- Anden normalform: ”
  1. *Den opfylder alle kravene til første normalform.*
  2. *Ingen attributter/ egenskaber, der ikke selv tilhører nøglen, må afhænge af en del af nøglen.”* (CphBusiness, 2023, s. 3).
- Tredje normalform: ”
  1. *Vi har for hver tabel en nøgle, der entydigt identificerer den enkelte række i tabellen.*
  2. *De enkelte felter indeholder kun en værdi.*
  3. *Vi har ingen kolonner, der gentages.*
  4. *Ingen af vores attributter eller egenskaber, der ikke selv tilhører nøglen, afhænger af en del af nøglen.”* (CphBusiness, 2023, s. 5).

Ved at vi overholder disse normalformer, sikrer vi os at vores database er funktionel, samt vi har en struktur, som vi kan bygge vores program op omkring, som tvinger os til at holde rammerne for vores projekt.

Vi har i vores gruppe lavet aftaler, om hvordan vi arbejder sammen. At lave en gruppekontrakt er et bindende dokument er et bindende dokument med regler om hvordan vi skal opføre sig. Ved at bruge dette er risikoen for at sænke folks arbejdsmoral samt lyst til samarbejde lavere. (Kvale & Brinkmann, 2009).

Dette er også understøttet gennem Weicks teori om meningsskabelse, qua:

- ***Social****: Meningsskabelse er en social proces, der sker gennem interaktion med andre.* (Hammer & Høpner, 2025)
- ***Ongoing****: Meningsskabelse er en konstant og løbende proces.* (Hammer & Høpner, 2025)

- ***Enactment**: Mening skabes gennem handlinger og den måde, vi interagerer med verden på, snarere end gennem passiv observation.* (Hammer & Høpner, 2025)

Ved at fokusere på disse tre teser ud af de originale 7, kan vi via *og* med hinanden, skabe mening for os selv, og dermed skabe større trivsel, samarbejde og arbejdsomhed. Således vi alle kommer hurtigst og bedst i mål med hinanden.

Ved at vores måde at skabe trivsel, er blevet beskrevet, vil vi yderligere gå i dybden med hvordan vores arbejdsfordeling er blevet implementeret.

Vores udgangspunkt var at vi skulle have følgende roller:

- **Scrum master**
- **Product Owner**
- **Tech Lead**

Disse roller (beskrevet i bilag 2 - Originale Rollefordeling) har dog meget fikserede punkter, vi har dog valgt at fragmentere disse og uddelegere, set i relation til vores respektive kompetencer, samt at arbejde ud fra vores styrker samt læringsbehov. Alt dette vil blive uddybet i proces.

Evaluering

For at sikre sig *sammenhængen*, *målene, tegnene og tiltagene* bliver overholdt, eller om de skal omdefineres, er det vigtigt at vide hvornår, og hvordan vi som gruppe evaluere på det. Dette sker både ud fra de respektive rollefordelinger - med henblik på hvem der skal evaluere det respektive arbejdsområde, ydermere har vi daglige morgenmøder, hvor gårsdagens udfordringer samt arbejde blive gennemtalt - i en meningsfuld sammenhæng.

Denne proces bliver også uddybet i proces.

Teknologi-valg

- Vi har valgt at implementere følgende teknologier:
- IntelliJ Amazon Corretto 17.06.16
- Maven compiler source 20
- Maven compiler target 20
- Javalin version 6.1.3
- javalin-rendering version 6.1.3
- thymeleaf version 3.1.2.
- thymeleaf-extras.version 3.0.4.
- slf4j.version 2.0.12
- jackson.version 2.17.0-rc1

- hikariCP version 5.1.0
- junit version 5.10.2
- hamcrest version 2.2
- postgresql version 42.7.2
- Docker 28.4.0
- JavaScript - Latest version. (ECMAScript 2025)
- HTML 5

Krav

I følgende afsnit er der to krav, som er blevet udleveret

> *"Hvad er firmaets håb med dette system (hvad er deres vision for systemet eller hvilken værdi er det jeres system skal tilføre deres virksomhed).*
>
> *User stories. Disse er udleveret i opgaven(...). (ek, 2025)*

Firmaets håb

Da det er et fiktivt selskab, må vores tanke om firmaets håb være en antagelse.

Det vi dog har antaget, er at de gerne vil have et system man som firma kan arbejde videre med. Derfor har vi lavet en større database, hvorved man dog på sigt kan lægge flere cupcakes - både toppings, men også bunde, ind i - uden ny kode skal implementeres. Da firmaet ikke har specificeret dette som kundekrav, er det ikke blevet implementeret.

Yderligere er der gjort klar til at der kan lægges rabatkoder ind, så hvis butikken på sigt gerne vil have det implementeret, er det forholdsvist simpelt.

ydermere, har vi forsøgsvis gjort brug af gestaltlove og komplementærfarver, og dermed prøvet at understøtte kundens og administratorernes tilgang til hjemmesiden. Dertil har vi gjort således at man altid kan komme til de sider vi fandt relevante på alle sider gennem brug af en header.

Gennem dette, er hjemmesiden altid let at gennemskue, samt let tilgængelig.

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 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 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.

Domæne-model og ER-diagram

ER-Diagram

Det ovenstående er vores ER-diagram.

Som det ses, indeholder vores database to steder hvor databasen kunne indeholde en mange til mange relation. Derfor har vi lavet to koblingstabeller *"cupcakes_in_a_order"* beskriver hvilken ordrer cupcakes er knyttet til og *"user_defined_cupcakes"* beskriver hvilke elementer cupcaken er bygget op af.

Det har vi gjort fordi en ordre kan indeholde forskellige cupcakes og forskellige cupcakes kan optræde i mange ordre.

Samtidig kan en top have mange bunde, og en bund kan have mange toppe.

Ydermere, har vi oprettet en tabel, der ikke har en automatisk som er zip-code. Dette er gjort fordi postnumre altid er unikke i forhold til den bydel de relatere til.

Ydermere, er tredje normalform ikke overholdt i *"cupcakes_in_a_order".* Dette er gjort, da tavlen udgør to primærnøgler, dermed er en primærnøgle her ikke nødvendig. Vi kunne have refereret til tavlen ved brug af en *"composite key"*, men fandt det unødvendigt datastrukturen.

Navigationsdiagram

Vi har beskrevet et diagram, da det virkede mere overskueligt end det udvidede diagram.

Vi indleder med de "globale overgange", med det menes hvor man kan føres hen til fra alle sider.

Efterfølgende er der det, som vi definerer som placeringer. Placeringer er den hjemmeside man betragter på det pågældende tidspunkt. Dette er efterfulgt af en kort beskrivelse af selve siden, og dernæst hvilke overgange der er fra selve siden - kaldet overgange. Til sidst er der de handlinger man kan lave på siden. Hvis der er en -> betyder det at denne handling leder videre til en anden side.

Globale overgange

Vores header - navigationslinje:

1. **Fra alle sider:**
    a. Navigation via topmenu (ikon række) til: *Index(hjem-ikon)*, *Order Page(cupcake-ikon)*, *About*, *Login/logout -> login krævet, betaling (indkøbsvognsikon)*.
        i. Hvis logget ind (administrator)->
            1. Navigation via topmenu (ikon række) til *Index(hjem-ikon)*, *Order Page(cupcake-ikon)*, *About*, (visuel e-mail), Admin, profile page, Login/logout, betaling(indkøbsvognsikon).Darkmode/LightMode (halvmåne)

        ii. Hvis logget ind (almen bruger) ->

Index(hjem-ikon), Order Page(cupcake-ikon), About, (visuel e-mail), profile page, Login/logout, betaling(indkøbsvognsikon), Darkmode/LightMode (halvmåne)

Placeringer (Administrator)

1. **Administrator Index**
    a. Velkommen til administrator, dagens ord, se kommende ordre, se kunder med flest ordre samt periodens salgstal.
    b. **Overgange:**
        i. Vis Kunder
        ii. Vis Ordrer
    c. **Handlinger**
        i. Indsæt et beløb på en kundes konto
2. **Administrator ordrer**
    a. Se alle ordre med e-mail, bestilling, og hvor mange de har bestilt.
    b. **Overgange**
        i. Vis kunder.
    c. **Handlinger**
        i. Søg ordre ud fra for- og/eller efternavn.
        ii. Slette en ordre fra listen
3. **Administrator vis kunder**
    a. Se alle kunder
    b. **Overgange**
        i. Vis ordre
    c. **Handlinger**
        i. Søg kunder ud fra e-mail

Placeringer (Kunde og administrator):

1. **Index**
    a. Viser introduktion, navigation til andre sider.
    b. **Overgange:**
        i. Bestil cupcakes
        ii. Opret bruger
        iii. Om os
2. **Login**
    a. Bruger indtaster e-mail og kodeord.
    b. **Overgange:**
        i. Create User - password (hvis "Sign up")

ii. Profile page (hvis login lykkes)

c. Handlinger

i. Kan logge ind

3. Create User - password

a. Bruger vælger kodeord og bekræfter det.

b. Overgange:

i. *Create User - information (hvis det lykkedes)*

1. brugeren ikke eksistere, samt begge passwords passer.

ii. *Login*

c. *Handlinger*

i. Sætter ens e-mail og bekræfter password -> create user information

4. Create User - information

a. Bruger indtaster navn, adresse osv.

b. Overgange:

i. profile page (når registrering er fuldført)

ii. login

c. Handlinger

i. Sætter ens informationer ->profile page

5. Profile Page

a. Brugeren kan se sine tidligere ordre og hvor mange penge der er på sin konto**.**

b. Overgange

i. Bestil cupcakes

6. About

a. Information om virksomheden og teamet.

7. Vælg Cupcake

a. Bruger vælger bund og topping fra dropdowns.

b. Overgange:

i. → *Order Page* (når bekræftet)

c. Handlinger

i. Vælger respektive bunde og toppings samt antal cupcakes

8. Order Page

a. Bruger kan se sin kurv, med bestilte cupcakes, forventet leverings-/afhentningsdato samlet pris.

b. Overgange:

i. Opdater information

ii. Gå til betaling

c. Handlinger

i. Fjern en ordre - eller flere fra din bestilling.

ii. Indtast en discount-kode der gør prisen mindre

9. Opdater Bruger-information

a. Bruger kan ændre kontakt- og adresseoplysninger.

b. Overgange:

i. Gå til order page, når en eller flere oplysninger er opdateret.

c. Handlinger

i. Opdater en eller flere bruger-oplysninger - med undtagelse af e-mail.

10. Betalingsside

a. Indtast sine kort-oplysninger

b. Overgange:

i. Ordrebekræftelse

c. Handlinger

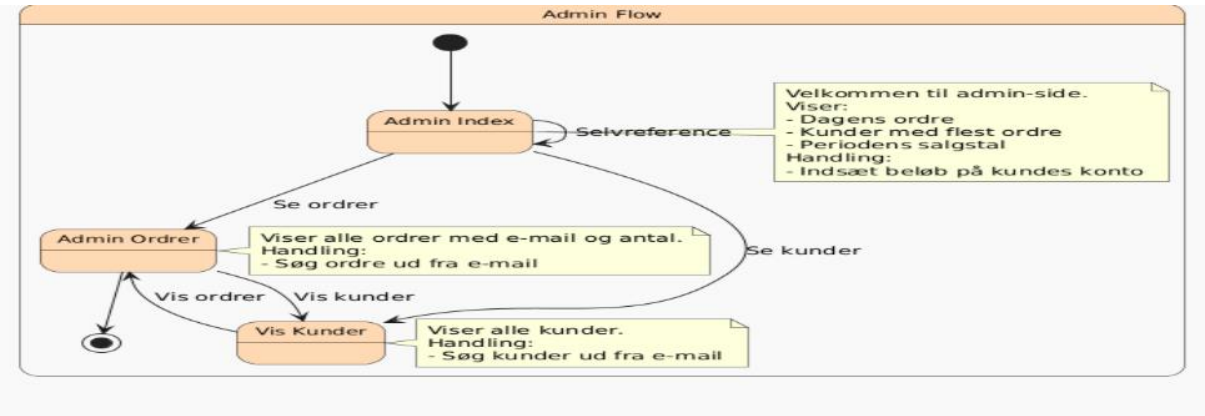
i. Brugeren indtaster kort-oplysninger - hvis korrekt-> ordrebekræftelse.

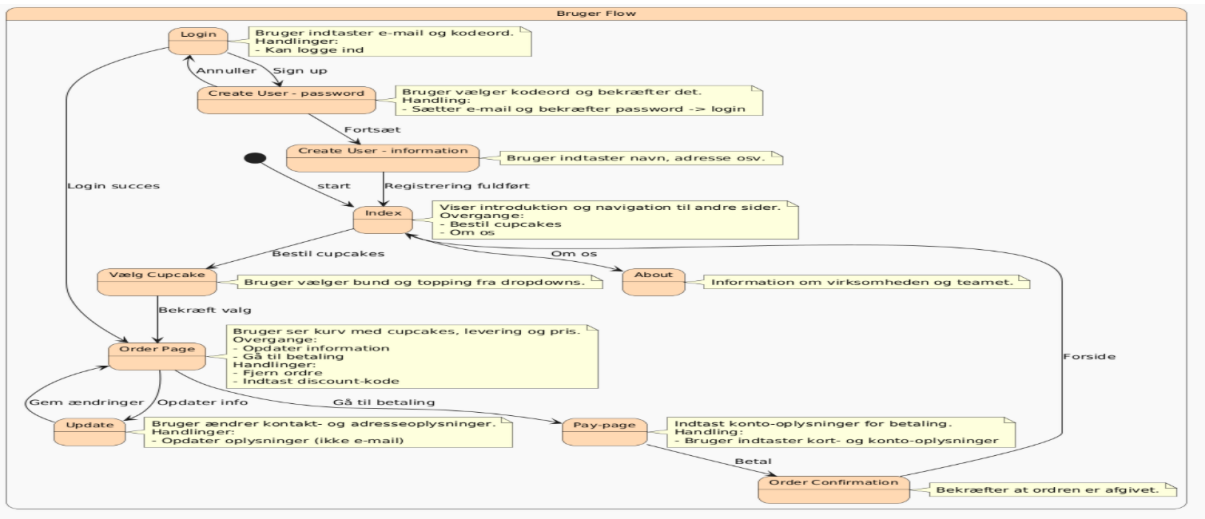
11. Ordrebekræftelse

a. Bekræfter, at ordren er afgivet.

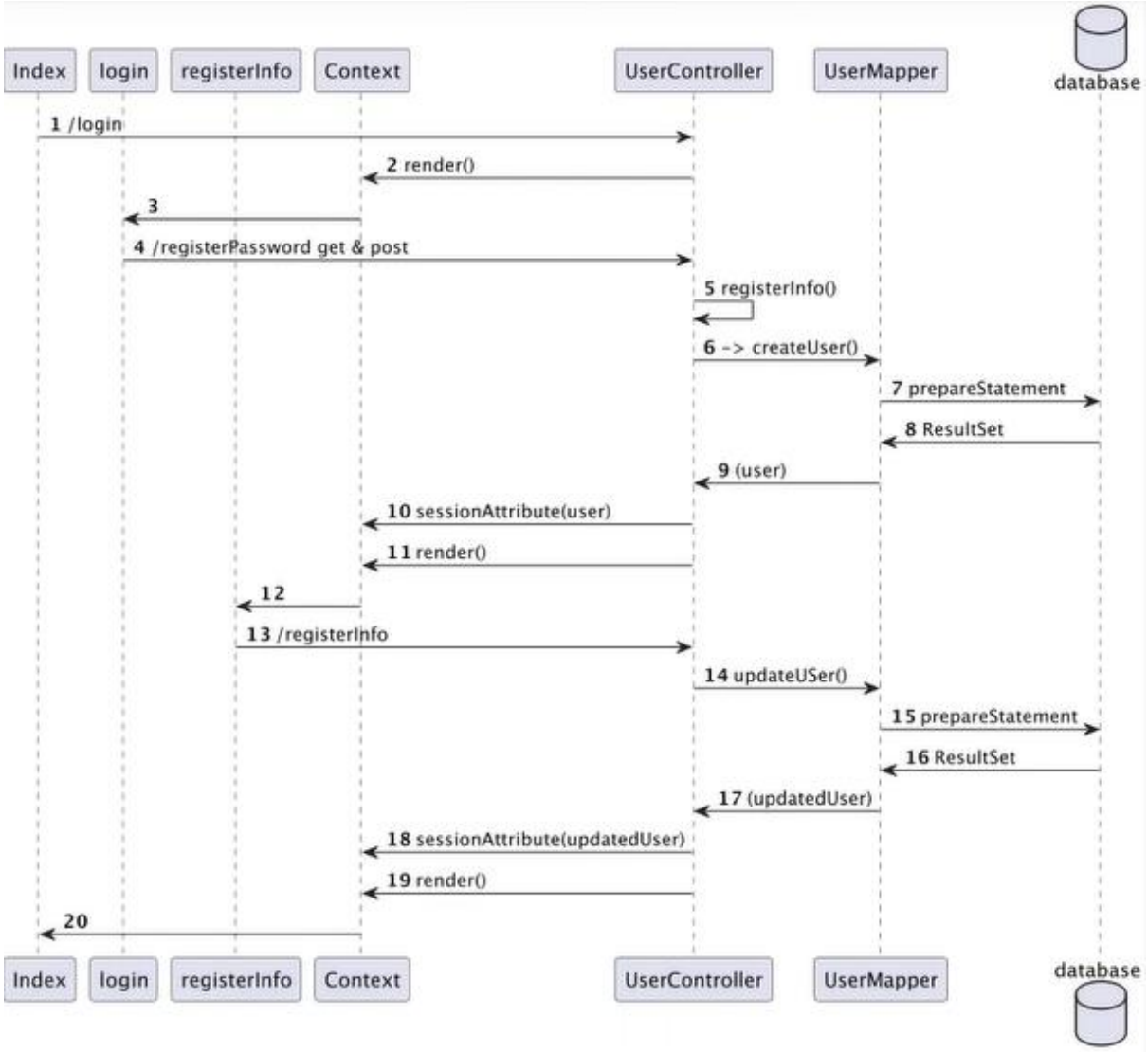
b. Overgange:

i. Index, ved tryk på forside.

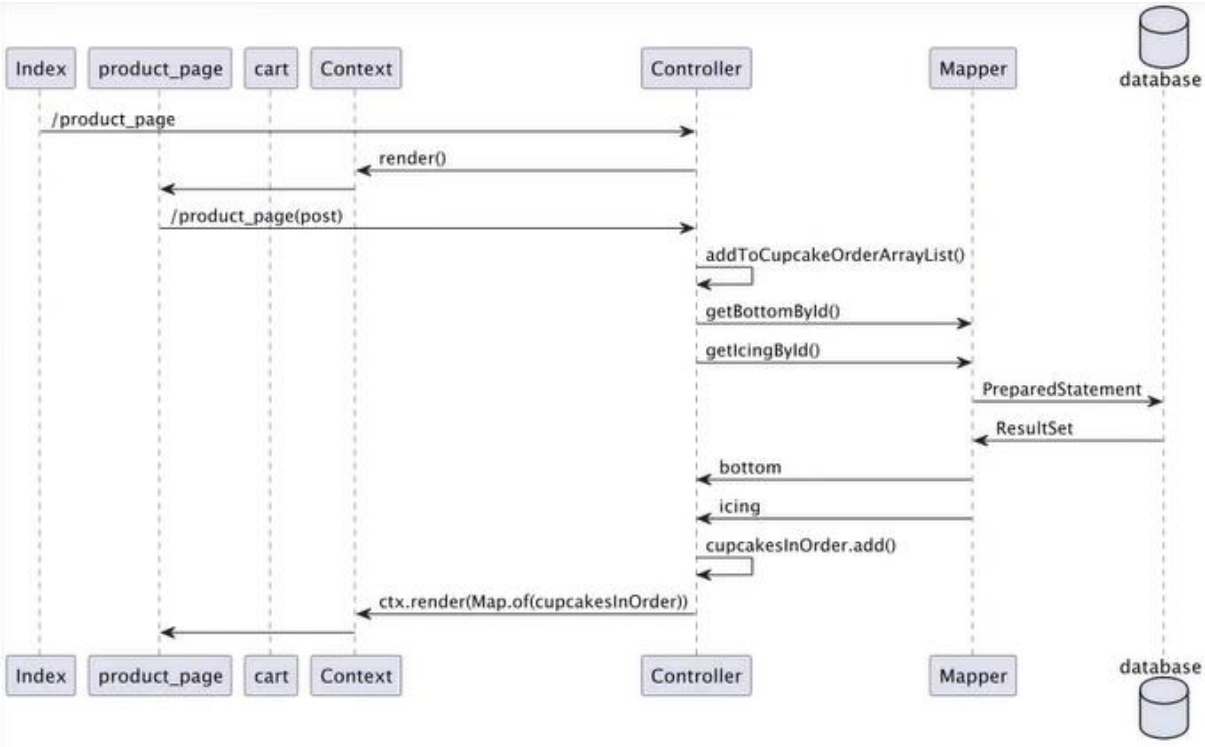




Sekvensdiagram



Opret bruger.



Bestil cupcake

Særlige forhold

- Hvordan man har valgt at lave validering af brugerinput.
    - Qua vores constraints, kan der ikke opstå ugyldige data i vores database, så felter der gerne må stå tomme, kan godt det, resten har vi brugt metoden *"required"* i vores HTML.
    - Ved vores e-mail, har vi i HTML også sørget for at "@" samt "." skal skrives i input, ved at sætte *"type="email""*.
    - Vores password behøver ikke have en vis længde, da dette er et mock-up, ellers havde vi lavet en *"required length"*.
    - Vi har beskyttet vores program ved at benytte *"prepared statements"*, hver gang vi skal ind i en database, dette sikre os mod sql-injections.
- Hvilke brugertyper, der er valgt i databasen, og hvordan de er brugt i jdbc.
    - Vi har valgt at skille vores administrator og bruger ad, således systemet tjekker en anden tabel hvorvidt brugeren er administrator eller ej.
    - Der er ydermere gjort klar til - men ikke implementeret at der kan være forskellige administratorer. i.e. en systemejer, en

superbruger, en almen medarbejder der bare kan læse ordre, vores backend støtter at denne implantation vil være let og overskuelig

    - Vi har benyttet en blanding af gode variable navne og coding conduct til at sikre at vores kode burde være let for andre at læse, og danne en naturlig rød tråd der findes igennem koden.

- Vi har brugt sessionAttributes til at gemme brugerens session når de for første gang lander på vores hjemmeside så de kan gemme cupcakes, uden at de forsvinder indtil brugeren selv lukker sessionen(hjemmesiden). Vi bruger det til:
    - En bruger der er logget ind.
    - Til vores indkøbskurv.
    - At gemme cupcakes i kurven.
    - Til diverse ”*Post*” forms og ”*Get*” forms.
- Vi er godt klar over at Javascript ikke er et krav eller forventet, derimod har vi givet os selv en udfordring til at implementere det i vores hjemmeside flere forskellige steder til at fremvise at vi har haft bedre forståelse for implantationer.
    - En af de steder hvor vores Javascript er mest tydelig er i vores product-page.html.
    - Vi har to ”<*select*>” elementer som vi giver et ID til vores Javascript, samt har vi giver vi et ID til vores ”<*img*>” elementer der sender det videre igen.
    - I vores Javascript har vi en funktion *”updateImage(select, img, previewImg)*”. Som kigger på vores valgte index i dropdown menu og sender en variable ”*data-image*” vi selv har defineret. Den siger hvis vores billeder i *”/images/*” mappen er ens med det valgte indeks, så generer billedet i mappen.

Status på implantationen

User-stories.

Som vi har beskrevet i mål, har vi kigget på de 9 user-cases, og defineret nogle forskellige delmål - ud fra hver user-story.

Vi har sammen fokuseret på samtlige af dem, og vurderet hvor langt vi er kommet med det.

- 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.*
  - **Færdige krav:** Kunden kan vælge bund/top, se ordredetaljer, kunden kan betale for sin ordre, og ordren gemmes korrekt i databasen med hentetidspunkt og status.
- US-2
  - *Som kunde kan jeg oprette en konto/profil for at kunne betale og gemme en ordre.*
  - **Færdige krav:** Kunden kan oprette/redigere profil, se tidligere ordrer.
- US-3
  - *Som administrator kan jeg indsætte beløb på en kundes konto direkte i Postgres, så en kunde kan betale for sine ordrer.*
  - **Funktionelle krav:** Administrator har en simpel grænseflade (eller SQL-kommando) til at indsætte beløb på kundens konto.
  -
- US-4
  - *Som kunde kan jeg se mine valgte ordrelinjer i en indkøbskurv, så jeg kan se den samlede pris.*
  - **Færdige krav:** Kurven opdateres dynamisk ved tilføjelse/fjernelse, viser korrekt total og kan gemmes mellem sessions.
  -
- 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.*
  - **Færdige krav:** Login fungerer, e-mail vises i topmenuen, og adgangskontrol håndhæves korrekt (kunde/administrator-adskillelse).
- US-6
  - *Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.*

    - **Funktionelle krav:** Listen viser detaljer pr. ordre (kunde, cupcakes, totalbeløb).

- 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.*
    - **Funktionelle krav:** Administrator kan se kundedetaljer inkl. ordrer for hver kunde.
- US-8
    - *Som kunde kan jeg fjerne en ordrelinje fra min indkøbskurv, så jeg kan justere min ordre.*
    - **Funktionelle krav:** Kurven opdateres, og totalbeløbet justeres automatisk.
    -
- US-9
    - *Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at vise ugyldige ordrer.*
    - **Færdige krav:** Ordrer kan slettes, databasen håndtere handlingen og UI opdateres.

Gestaltlovene

Vi har på alle siderne forsøgt efter vores fortolkning og bedste evne, prøvet at overholde gestaltlovene.

Der er dog to steder hvorved de ikke er overholdt.

Den ene er på product-page med mængden af cupcakes. Dette kommer af en HTML og CSS-udfordring, hvorved vi ikke kunne få mængde-knappen til at fungere hvis vi placerede den hvor den skulle have været.

Havde vi som gruppe haft en dag til, antager vi at det godt kunne have nået at blive implementeret.

Den næste er på administrator-index. Der har vi erfaring fra en i gruppen, der arbejder med køkken- og ordretabeller, der havde erfaring med at det var mere behageligt at sætte det således op, grundet det er to tabeller der ikke har noget tilfælles, så de på denne måde er separerede men stadig sammenhængende.

Vi har dog benyttet os af fragments til at overholde loven om lighed på alle sider på vores header og footer.

Databasen

Vi har til selve databasen implementeret alle CRUD-metoder som får systemet til at fungere i forhold til de respektive user-stories.

Det der på sigt godt kunne blive lavet, var CRUD-metoder til resten. Havde dette været et kundekrav, ville det godt have kunne være implementeret.

Proces

Hvordan antog vi processen skulle forløbe

Vores umiddelbare plan var at dele rollerne op ift. vores respektive kompetencer qua, vi alle har nogle baggrunde der styrker forskellige aspekter set i rollernes ansvarsområder - *Scrum master, Product owner* og *tech lead.*

Vi valgte at omfordele rollerne i fire andre formater, men alle tangerende til de tre originale.

Årsagen til dette var at vi alle skulle have lov til at forsøge sig med det hele. Da det er vores første større opgave, fandt vi som gruppe det at finde sin respektive spidskompetence og fokusere på det, kontra at uddelegere ud fra noget man ikke nødvendigvis har kompetencerne til. Derfra antog vi at vi bedre kunne skabe mening i relation til opgaven.

Så taget ud fra vores givne rollefordelinger, blev vores umiddelbare rollefordeling blev således:

- **GitMaster - Emil**
    - Holde overblikket over hele projektet. (selv tilføjet)
    - I kigger på jeres Git Project eller hvad I ellers bruger til at holde styr på opgaverne.
    - kigge på pull request og acceptere eller afvise dem.
    - tage en beslutning ved diskussioner om design af koden. Husk at alle skal høres. Husk at notere diskussionen og beslutningen i log og få det med i rapporten.
- **Backend Tech-lead - Frederik**
    - Holde fokus - samt have styr på back-end. (selv tilføjet)
    - agere kunde, hvis I er i tvivl om noget, der har med kravene at gøre.
    - sikre at kodestandarder er overholdt.
- **Front-end Tech-lead - Daniel**
    - Holde fokus på HTML og CSS samt controllers. (selv tilføjet)

    - prioritere jeres opgaver. Hvad vil kunden have fokus på?
    - sikre at DoD er overholdt.

- **Scrum-Master - Luke**

    - Tovholder på dokumentationen. (selv tilføjet)
    - sikre at I afholder stand-up møder. Møderne skal være korte.
    - Alle kommer med en kort status af hvad de laver og om der er nogle forhindringer.
    - holde styr på planen for ugen og vide hvem der arbejder hvor. I tilfælde af sygdom melder I ind til scrum master.
    - tage fat i vejlederen, hvis der opstår problemer, I ikke selv kan klare.
    - Scrum masterens opgaver er at sikre, at processen kører, at teamet fungerer godt sammen, og at der ikke opstår blokeringer i arbejdsflowet.

- **Fælles**

    - sikre at alle høres når I diskuterer løsninger.
    - tage en beslutning i kraft af sin kunde-rolle. Husk at notere i jeres log hvilke beslutninger, der bliver taget og hvilke diskussioner I havde. Tag det med i rapporten.

Processen i praksis

I ren praksis, kom det til at ligne det meget.

Den første uge, kørte vi de forskellige roller på skift, så alle kom til at forsøge sig med det hele, på trods af at det stadig var den respektive person, som havde ansvaret for den rolle.

Samtidig ses det også at Luke havde mange roller ift. de andre, men de roller var forholdsvis små, og primært relevante ift. morgenmødet. Luke kodede dog med der hvor der var behov i relation til opgaven.

Som det er nævnt i vores Team-Manifest, skulle alle høres.

Derfor holdt vi daglige morgenmøder.

Vores morgenmøder var struktureret på følgende måde:

- Vi startede med at sige pænt goddag, og havde det sjovt med hinanden, inden vi skulle i gang med arbejdet, for at starte dagen på en god måde.

    - I særdeleshed for at bruge vores tre udgangspunkter i sensemaking.
- Morgenmødet startede med at dialog om hvad der var blevet lavet dagen forinden. Havde man lavet det som aftalt eventuelt mere eller mindre? - eventuelt andet.
    - Hver person havde cirka 5 min. Til at forklare hvad personen havde lavet, og næste skridt i processen. Var der behov for mere tid, så fik man selvfølgelig det.
- Der blev merget fra GitHub - med det som skulle merges ind.
- Use-casene blev gennemgået, for at fokusere på det næste skridt i processen.
    - Derfra kunne vi sammen have en praktisk dialog om hvad der så skulle laves.
- Efter dette tog mødelederen i samråd med de andre, beslutningen om hvad der skulle prioriteres, og hvorfor. Der var dog alle gangene fælles konsensus.

Folk brød derefter ud i grupper, for at arbejde med det de skulle arbejde med.

Der var altid mulighed for at spørge hinanden til råds, og få det man havde lavet gennemgået.

Kl. 13, holdt vi et status-møde, for at holde styr på hvor langt vi hver især var kommet, og om dagens mål kunne nås.

Kl.15:30, holdt vi et afsluttende status-møde, for hvad dagen derpå skulle indeholde, og hvad dagen derpå skulle indeholde. Derudover var der frivillig aften-kodning for dem der havde lyst.

Det skal hertil siges at møde-dokumentationen blev mindre desto længere vi kom i projektet. (se bilag 5 - møde-planlægning)

Læringsprocessen

Det som gik rigtig godt, var vores umiddelbare kommunikation. Som det er forventeligt ved gruppearbejde, var der nogle gange uenighed omkring hvordan man skulle løse et specifikt problem. Qua vores team-manifest, blev det dog hurtigt løst igen, uden nødvendighed for at tale det igennem.

Ved at vi havde brugt Figma til at sætte hele projektet op, som det første, samt at vi havde sat vores database op, havde vi alle et fælles mål.

Vi var alle enige om hvad vi skulle lave, på det pågældende tidspunkt, og hvis man manglede noget at lave, var der altid arbejde at dele ud af.

Samtidig havde vi gennem vores uddybede user-stories, noget vi kunne alle kunne måle hvor langt vi var nået i projektet. Dette kunne også mærkes i vores daglige morgenmøder, hvor vi italesatte hvor langt vi var nået.

Vi brugte samtidig rigtig meget *code with me*, som havde fordelen af at vi alle vidste hvad alle lavede på samme tid. Derfor har alle en god grundlæggende forståelse af koden.

En anden måde vi dog kunne have sat det op på, var at lave generelle mål, således vi alle havde et delprojekt, man havde et antaget tidspunkt det skulle kunne nås til.

Derfor kunne en læringsproces være at vi næste gang arbejde med fragmenter af koden, implementerede dette, for så at arbejde videre med den næste del.

Dette skal forstås at arbejde med branches, med en specifik opgave hver.

Dette vil også hjælpe på at kunne arbejde mere med gitHub, og kunne push og commit oftere - hvilket vi som gruppe ikke har været så gode til i denne omgang.

Bibliografi

CphBusiness. (09. 02 2023). Normalisering.

ek. (20. 10 2025). *https://2semfall2025.kursusmaterialer.dk/projects/cupcake/rapportskabelon/*. Hentet fra https://2semfall2025.kursusmaterialer.dk/projects/cupcake/rapportskabelon/

Froes, I. (2017). *Playful Literacy: A Thesis on Young Children's Play Practices on Digital Tablets.* IT-universitet i København. Hentet fra https://research.cbs.dk/da/publications/playful-literacy-a-thesis-on-young-childrens-play-practices-on-di

Hammer, S., & Høpner, J. (2025). *Meningsskanbelse, organisering og ledelse En introduktion til Weicks univers.* Samfundslitteratur og forfatterne.

Kvale, S., & Brinkmann, S. (2009). *Interview: Introduktion til et håndværk.* Hans Reitzels Forlag.

Multimededesigneren. (2022). *Multimediedesigneren.dk*. Hentet 20. 10 2025 fra Multimediedesigneren.dk: https://multimediedesigneren.dk/gestaltlovene/

Ryberg, J. (12. 08 2025). *lex.dk*. Hentet 22. 10 2025 fra https://lex.dk/det_kategoriske_imperativ

Bilag

Bilag 1 user-stories. - User-stories defineret som US.

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 orddelinger 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 ordrelinie 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 udgyldige ordrer. F.eks. hvis kunden aldrig har betalt.

Bilag 2: Coding-conduct

I følgende bilag, vil vi sætte præmissen for hvordan vi vil sætte vores kode op.

Det basale:

- Al kode, kommentarer og commit-beskeder bliver skrevet på engelsk.
- Vi vil så vidt det er muligt sørge for at alle commit-beskeder er saglige og tydelige.
- Kommentarer I kode, skal altid være relateret til ”special case” - det vil sige, forklare utydelig kode, eller hvorfor noget der virker overflødigt, er relevant.
- Vi overskriver og sletter ikke bare kode, der tales altid med den relevant person, for at sikre sig essentiel kode ikke bliver slettet.

Syntax:

Variabel navne:

Det vi ikke gør:

```
public int addN(int æ, int w)
{
    return w + æ;
}
```

Ovenfor ses hvordan at to variabler hedder æ og w, som ikke er tydelige eller forklarende, så derfor bliver koden forvirrende.

Det vi gør:

```
public int addTwoNumbers(int numberOne, int numberTwo)
{
    return numberOne + numberTwo;
}
```

Vi sørger for at variabelnavnene er tydelige og forklarende, så andre der skal læse vores kode, kan se hvad vi gør og hvorfor.

Metode-struktur:

Det vi ikke gør:

```
public int addTwoNumbers(int numberOne, int numberTwo){
        return numberOne + numberTwo;
    }
```

Som vi ser, så starter selve metodekroppen på selve metodehovedet.

Det kan for nogle godt virke forvirrende at have metodehovedet og metodekroppen sammen.

Det vi gør:

```
public int addTwoNumbers(int numberOne, int numberTwo)
    {
        return numberOne + numberTwo;
    }
```

Holder selve metodekroppen for sig, og metodehovedet for sig.

If statements:

Det samme gør vi for if-statements.

Så vi gør ikke følgende:

```
if(text.equals("w")){

};
```

I stedet gør vi således:

```
if ( text.equals( "w" ))
{

};
```

Bilag 3 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.*
  - **Minimumskrav:** Kunden kan vælge bund og top og tilføje en cupcake til en ordre.
  - **Funktionelle krav:** Kunden kan vælge bund og top, se samlet pris og gennemføre betaling.
  - **Færdige krav:** Kunden kan vælge bund/top, se ordredetaljer, kunden kan betale for sin ordre, og ordren gemmes korrekt i databasen med hentetidspunkt og status.

- **US-2**
  - *Som kunde kan jeg oprette en konto/profil for at kunne betale og gemme en ordre.*
  - **Minimumskrav:** Kunden kan indtaste e-mail og kodeord og oprette en bruger i databasen.
  - **Funktionelle krav:** Kunden kan logge ind, gemme ordrer og foretage betaling via sin profil.
  - **Færdige krav:** Kunden kan oprette/redigere profil, se tidligere ordrer.

- **US-3**
  - *Som administrator kan jeg indsætte beløb på en kundes konto direkte i Postgres, så en kunde kan betale for sine ordrer.*
  - **Minimumskrav:** Administrator kan tilgå kundetabellen og ændre balance manuelt.
  - **Funktionelle krav:** Administrator har en simpel grænseflade (eller SQL-kommando) til at indsætte beløb på kundens konto.
  - **Færdige krav:** Administrator kan sikkert og nemt ændre saldo via administrator-brugerinterface.

- **US-4**
  - *Som kunde kan jeg se mine valgte ordrelinjer i en indkøbskurv, så jeg kan se den samlede pris.*
  - **Minimumskrav:** Kurven viser en liste over valgte cupcakes.
  - **Funktionelle krav:** Kurven viser hver cupcake med pris og samlet total.
  - **Færdige krav:** Kurven opdateres dynamisk ved tilføjelse/fjernelse, viser korrekt total og kan gemmes mellem sessions.

- **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.*
  - **Minimumskrav:** Loginformular accepterer e-mail og kodeord og tjekker mod databasen.
  - **Funktionelle krav:** Login giver adgang til brugerspecifikke sider, og e-mail vises på hver side.
  - **Færdige krav:** Login fungerer, e-mail vises i topmenuen, og adgangskontrol håndhæves korrekt (kunde/administrator-adskillelse).

- **US-6**
  - *Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.*
  - **Minimumskrav:** Administrator kan se en liste med alle ordrer i databasen.
  - **Funktionelle krav:** Listen viser detaljer pr. ordre (kunde, cupcakes, totalbeløb).
  - **Færdige krav:** Listen kan sorteres/filtreres, og ordredetaljer kan åbnes individuelt.

- **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.*
  - **Minimumskrav:** Administrator kan se en liste over kunder.
  - Funktionelle krav: Administrator kan se kundedetaljer inkl. ordrer for hver kunde.
  - **Færdige krav:** Systemet viser kundeoversigt med søge- og filtreringsmuligheder, ordrer er linkbare, og data præsenteres overskueligt.

- **US-8**
  - *Som kunde kan jeg fjerne en ordrelinje fra min indkøbskurv, så jeg kan justere min ordre.*
  - **Minimumskrav:** Kunden kan slette en ordrelinje fra databasen eller kurven.
  - **Funktionelle krav:** Kurven opdateres, og totalbeløbet justeres automatisk.

    - **Færdige krav:** Brugeren kan fjerne linjer med bekræftelsesdialog, kurven opdateres dynamisk, og ændringer gemmes.

- **US-9**
    - *Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at vise ugyldige ordrer.*
    - **Minimumskrav:** Administrator kan slette en ordre fra databasen
    - **Funktionelle krav:** Administrator kan slette ordrer via administrator-interface med visuel feedback.
    - **Færdige krav:** Ordrer kan slettes, databasen håndtere handlingen og UI opdateres.

Bilag 4 - Originale rollebeskrivelser

Team roller

Ansvarsområder i teamet

I jeres team skal I dele ansvarsområderne ud. I skal skifte ansvar hver uge. I trækker lod om hvem, der først tager hvilken rolle. I må gerne være to om samme rolle, hvis I har lyst.

Scrum master

Scrum masterens opgaver er at sikre, at processen kører, at teamet fungerer godt sammen, og at der ikke opstår blokeringer i arbejdsflowet.

Scrum masters konkrete opgaver er at

- sikre at I afholder stand-up møder. Møderne skal være korte. I kigger på jeres Git Project eller hvad I ellers bruger til at holde styr på opgaverne. Alle kommer med en kort status af hvad de laver og om der er nogle forhindringer.
- sikre at gruppekontrakten overholdes. Hvis der er noget, der ikke fungerer, indkalder scrum master til et møde, hvor I får talt om hvad I skal gøre.
- sikre at alle høres når I diskuterer løsninger.
- sikre at I er klar til vejledning og ved hvad I gerne vil have ud af vejledningen. Det er ikke scrum master, som skal formulere spørgsmål, men vedkommende skal sikre, at teamet har talt om hvad der skal ske til vejledning, hvem der præsenterer for vejleder mv.
- holde styr på planen for ugen og vide hvem der arbejder hvor. I tilfælde af sygdom melder I ind til scrum master.
- tage fat i vejlederen, hvis der opstår problemer, I ikke selv kan klare.

Product Owner

Product owner er kundens forlængede arm. Det vil i praksis sige, at product owner taler med kunden og formidler kravene videre til teamet.

Product owners konkrete opgaver er at

- agere kunde, hvis I er i tvivl om noget, der har med kravene at gøre (fx vil Martin mon kunne se alle ordrer eller bare de 10 første?).
- prioritere jeres opgaver. Hvad vil kunden have fokus på?
- tage en beslutning i kraft af sin kunde-rolle. Husk at notere i jeres log hvilke beslutninger, der bliver taget og hvilke diskussioner I havde. Tag det med i rapporten.

- tage fat i vejlederen, hvis det er en større beslutning, I ikke selv kan tage.

Tech lead

Tech lead har sidste ord når det kommer til det tekniske. Tech lead er ansvarlig for kvaliteten af jeres kode.

Tech leads konkrete opgaver er at

- kigge på pull request og acceptere eller afvise dem.
- sikre at kodestandarder er overholdt.
- sikre at DoD er overholdt.
- sikre at der dokumenteres og testes som aftalt.
- tage en beslutning ved diskussioner om design af koden. Husk at alle skal høres. Husk at notere diskussionen og beslutningen i log og få det med i rapporten.
- tage fat i vejlederen, hvis I har brug for input til det tekniske.

Bilag 5 - Team Manifest

"Handl kun efter den maksime, om hvilken du tillige kan ville, at den bliver en almen lov" (Ryberg, 2025).

1. ALTID sørge for, at der er et ordentligt sprog, når der bliver kommunikeret.

2. ALDRIG være bange for at sige din mening.

3. VÆR MED til at skabe et fællesrum, hvor der er respekt for medstuderende.

4. KOM til forudbestemte møder, arrangementer og andre aktiviteter.
    a. HVIS du ikke kan komme/bliver forsinket, så meld det til en fra gruppen.

5. HOLD FOKUS så længe der er behov for det.
    a. Blive enige i fællesskabet om, hvor længe vi vil bruge på opgaven.
        i. Bliv enige med dig selv om, hvor længe du vil bruge, og rapporter det til gruppen.
    b. GIV DIT BEDSTE og ikke en andens.
    c. RESPEKTER andre arbejdsmåder.
    d. VÆR GOD til at gå på kompromis, når der er behov for det.

6. HJÆLP HINANDEN i presset situationer.

7. LAV EN EVALUERING efter hvert gruppearbejde, så vi kan forbedre os til næste opgave

Bilag - 6 Møde-planlægning

Dag 1

(Luke)

Mødested:?

(Discord? Kl. ?)

Personligt ville jeg gerne mødes personligt, men gider ikke tage hele vejen ud til skolen, og da Daniel bor for langt væk fra København, tænker jeg Discord er optimalt.

Vi er samtidig kommet så langt at vi godt kan lave en forholdsvis kort dag med en optional bagefter. Tænker mødetid mellem kl. 9 og 10.

Jeg laver gerne kaffe hvis folk vil mødes her, tager også gerne ud til andre hvis meet-up er en (delvis)ting.

Plan: Mødes og sige goddag. Evt. Smalltalk 15 min.

Gennemgå rapportskabelon: -Luke, 10 min. Efterfølgende spørgsmål og dialog 20 min.

1.1 gennemgå US:1-9, har vi alle forstået det på samme måde. 45-75 min.

1.2 IFK: Hvor langt er vi kommet? Hvem har lavet hvad - forstår alle det? Lucid Figma ERD Database HTML CSS Er vi færdige 30-60 min.

Kort pause15 min.

1.3. Gruppekontrakt: vi er alle voksne, så gider ikke sætte tid af til andet end at vi skal være sammen om at skrive den - hvis folk har kommentarer til det.

15 min Forventningsafstemning: Hvor meget koder vi sammen hvor meget koder vi hver for sig - hvornår og hvordan melder man sig til og fra? 30 min.

1.5 roller i teamet: Hvem er ansvarlige for hvad? Luke laver indspark til dialog 5 min. 30-75 min.

Planlægning mandag-mandag: 45 min. Tak for I dag, kod godt videre.

Dag 7

(Luke)

Hvad vi mangler pr. d. 28/10 2025:,

ORDRE,

Se den samlede pris

HTML/CSS/NAVIGATION,

Alle Navigationslinjer skal være ens

Lav about us

Fix index Evt.

scratch dagens ord.

Media Query - size