Datamatiker 2. semester

Cupcake

CPH Business Lyngby

Deltagere

Navn: Abbas M. Badreddine

CPH e-mail: cph-ab632@cphbusiness.dk

Github navn: AbbasMB

Navn: Frederik Bastiansen

CPH e-mail: cph-fb157@cphbusiness.dk

Github navn: Frederik-BipBop

Navn: Emil Wolfert Schmidt Andersen

CPH e-mail: cph-ea178@chpbusiness.dk

Github navn: iirne

Navn: Thomas Atchapero

CPH e-mail: cph-ta241@cphbusiness.dk

Github navn: Tatchapero

Marts/April 2025

Indholdsfortegnelse

Indledning

Olsker Cupcakes-projektet omhandler en frontend- eller backend-hjemmeside, hvor man kan bestille sin egen custom cupcake. Når en bruger har lagt en ordre i kurven, bliver den gemt i databasen og hentet ned til hjemmesiden igen.

En bruger har en valuta, som de betaler med på hjemmesiden. For at en bruger kan få tilføjet valuta, skal en admin indsætte deres valuta manuelt via et script. Admin kan se alle kunder, slette ordrer og indsætte valuta.

Baggrund

Olsker Cupcakes er et bageri, der er startet som iværksætter-virksomhed, og er lokaliseret på Bornholm. Virksomheden ønsker en hjemmeside hvor deres kunder kan bygge deres egne cupcakes og placere en ordre. Hjemmesiden skal også understøtte administrator funktionalitet, således at virksomheden kan holde styr på ordrerne. Virksomheden har selv udarbejdet et mock-up, som de ønsker at deres hjemmeside skal ligne. Derudover har virksomheden stillet både nogle funktionelle krav og nogle ikke-funktionelle krav, som står nærmere beskrevet i afsnittet Krav.

Teknologi valg

- Postgresql 16.2
- Pgadmin 4 (v.8.13)
- HTML
- CSS
- IntelliJ IDEA 2024.2.3
- Figma
- Java 17
- JUnit 5.12.0
- Javalin 6.5.0
- Javalin Rendering 6.5.0
- Thymeleaf 3.1.3

- Thymeleaf Extras 3.0.4
- Jetty
- HikariCP 6.2.1

Krav

Virksomhedens mål og vision med hjemmesiden er

- At gøre det nemmere for virksomhedens kunder at bestille cupcakes
- At gøre det nemmere for virksomheden at holde styr på deres bogholderi
- At give virksomheden mulighed for at kunne se deres kunder
- At gøre det nemmere for virksomheden med, at skabe et overblik over fuldførte og ikke fuldførte ordre

Funktionelle krav

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.
2. Som kunde kan jeg oprette en konto/profil for at kunne betale og gemme en en ordre.
3. Som administrator kan jeg indsætte beløb på en kundes konto direkte i Postgres, så en kunde kan betale for sine ordrer.
4. Som kunde kan jeg se mine valgte ordrelinjer i en indkøbskurv, så jeg kan se den samlede pris.
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).
6. Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.
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.
8. Som kunde kan jeg fjerne en ordrelinje fra min indkøbskurv, så jeg kan justere min ordre.

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

Ikke-funktionelle krav

1. Der laves en mock-up i Figma eller lignende, som viser de websider den færdige løsning kommer til at bestå af.
2. Ordrer, kunder og øvrige data skal gemmes i en database.
3. Databasen skal normaliseres på 3. normalform med mindre andet giver bedre mening.
4. Kildekoden skal deles på GitHub.
5. Det færdige produkt skal udvikles i Java 17, Javalin, Thymeleaf template engine, PostgreSQL Database, HTML og CSS.
6. Websitet skal helst kunne fungere tilfredsstillende både på en almindelig skærm og på en mobiltelefon (iPhone 12 og lignende). Hvis det volder problemer, så lav kun jeres løsning til en laptop.

Aktivitetsdiagram

Figur 1 viser det workflow, som hjemmesiden kommer til at afspejle. Diagrammet viser i grove træk, hvordan en kunde kan købe cupcakes, og hvilke scenarier som kunder kan ende ud i, hvis de eksempelvis ikke har nok penge på kontoen.

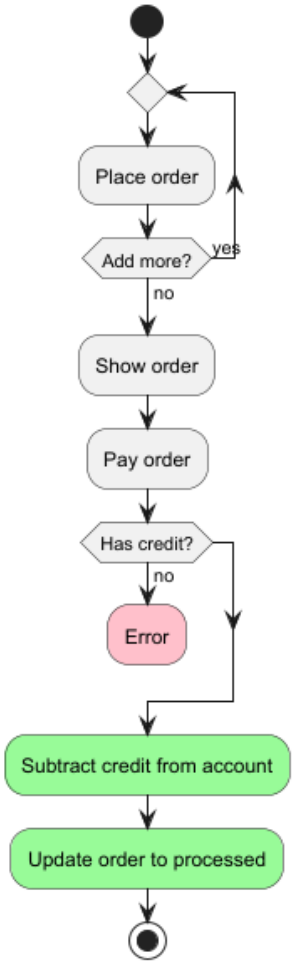

Fig. 1: Aktivitetsdiagram TO-BE af workflowet

Domæne model og ER diagram

For at skabe enighed omkring hvordan programmet skal være opbygget før man begyndte på kodningen, har der været dannet en domænemodel samt ERD for at vide hvilke entiteter der ville være i systemet samt deres relationer til hinanden. Dette sørger for at der ikke ender med at lave entiteter som indeholder data som bliver dækket af andet, samt at holde systemet simpelt.

Domænemodel

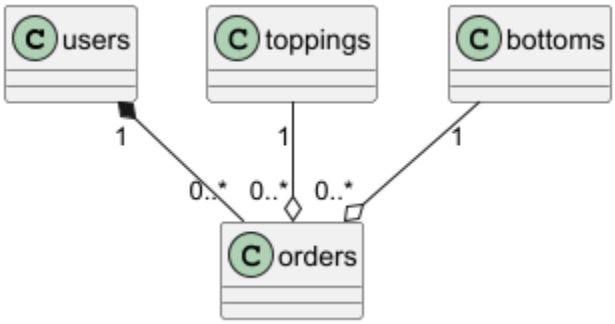

Fig. 2: Domænemodellen viser domæne objekter, samt deres relationer og multiplicitet

Relationen mellem brugere og ordrer er, at en bruger kan lave flere ordrer, men hver ordre hører til én bestemt bruger, dvs. 1-til-mange relation.

Relationen mellem ordrer & toppings/bottoms er at en ordre består af én type af topping og én type af bottom, men bottoms & toppings kan bruges i mange ordrer - Igen en 1-til-mange relation.

ER diagram

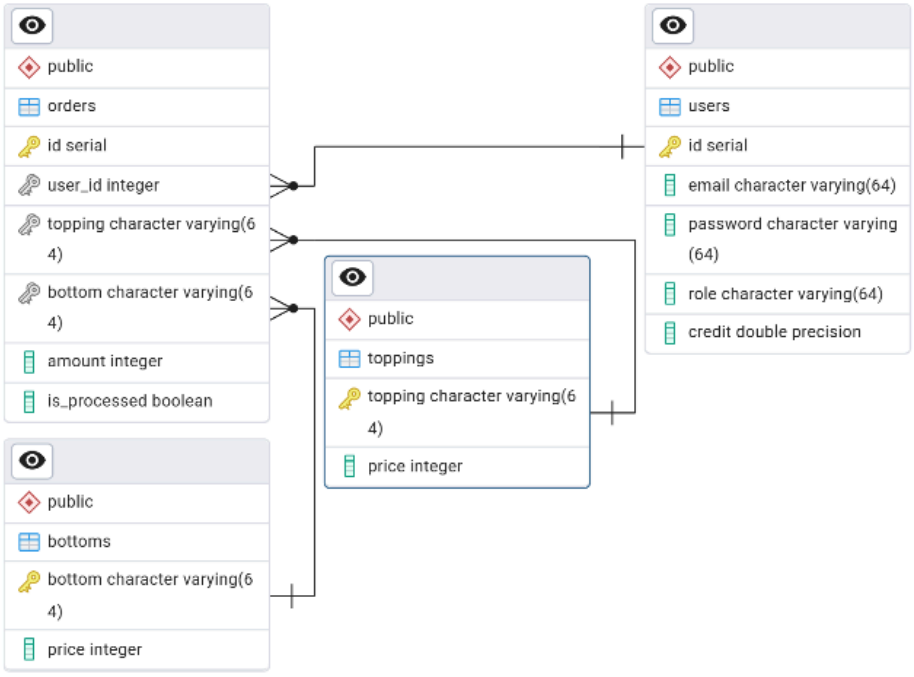

Fig. 3: Entity Relation Diagram viser entiteter, samt deres attributter og relationer

I ERD diagrammet er der ikke blevet brugt autogenererede ID'er i alle tabeller, der er specifikt ikke brugt i bottom og toppings. Det har de ikke, da der menes at navnene er unikke og beskrivende og de forventes ikke at de ændres ofte og dermed fungerer godt som stabile primære nøgler.

Der er blevet aktivt brugt foreign keys for at sikre at vores data hænger sammen korrekt. Her er hvordan der bliver sikret det:

- orders.user_id refererer til users.id
- orders.topping refererer til toppings.topping
- orders.bottom refererer til bottoms.bottom

Disse relationer gør, at der kan oprettes ordrer med toppings og bunde, der eksisterer databasen, og at der ikke kan refereres til en bruger, som ikke findes. Det hjælper os med at sikre sammenhæng i vores data.

Navigation Diagram

Figur 4. viser hvordan en bruger trykker sig sin vej igennem vores hjemmeside. Den viser hvad du kan, som bruger og hvad en admin har af ekstra muligheder i forhold til en bruger.

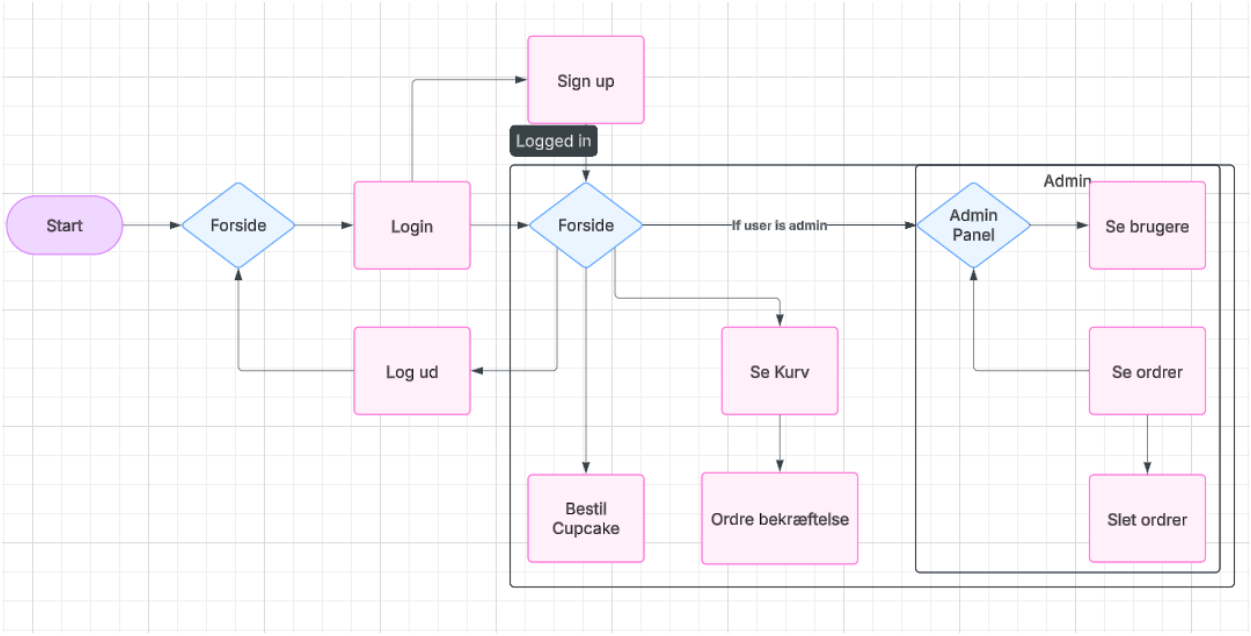

Fig. 4: Navigations diagrammet viser de sider man kan navigere til og deres composite state

Særlige forhold

Session attributes

For at sørge for at brugeren ikke skal logge ind har der været taget brug af et session attribut som gemmer et snapshot af brugeren og tillader dem at handle ud fra dette snapshot. Dette kan dog skabe problemer med at hvis der sker ændringer i brugeren ude fra sessionen, ville det ikke påvirke brugeren i sessionen. Det ville betyde at en bruger kan potentielt betale med penge de ikke har, hvis deres admin tilladelser bliver taget fra dem så ville det ikke påvirke dem indtil de logger ind igen og andre lignende problemer.

Exceptions

Ved alle vores handlinger hvori man skal tilgå databasen sørges der for at kaste en ny type exception for at sørge for at det kan håndteres af programmet hvad der skal ske i tilfælde af at databasen fejler.

Sikkerhed

For at undgå SQL injection har der været taget brug af prepared statements. Dette er fordi at prepared statements tillader brugeren at kunne stoppe sql tidligt og kører deres eget kodet, men det ville i stedet blive behandlet som tekstinput.

Samt for at sørge for at input er som forventet, har man i html gjort at indtastning af brugernavn kan kun være i form af en valid email både i signup og login for at minimere dårlig data.

I vores backend for at sørge for at en admin ikke kan se passwords, har der også været taget brug af at hashe alle passwords således at de ikke er menneskeligt læseligt. I programmet hasher vi det indtastede password og tjekker om det passer med det som ligger i databasen. På den måde har vi undgået af at i tilfælde af en databrud eller misbrug at passwords ikke bliver lettere komprimeret.

Brugere

Vores program har 2 typer af brugere, de er sat op som enten "null" eller “admin” i databasen. Dette bliver så sendt til front-end i bruger-entiteten som så skjuler og viser elementer angående hvilke bruger-type de er.

Status på implementation

Til refleksion over systemet, er det ikke gjort det muligt at kunne se en ordre som en gæstebruger. Man skal have en oprettet bruger for at kunne se og betale for ens ordre.

Det havde været til fordel for gæstebrugere at give dem mulighed for at lave en ordre og dernæst enten udfylde relevante informationer for at kunne lægge en ordre, hvis man ikke

ønsker at oprette en bruger.

Derudover er det ikke muligt for brugeren at se, hvor mange varer de har sat i indkøbskurven. Brugeren skal aktivt selv gå ind på indkøbskurven for at se hvor mange varer de har i indkøbskurven. Det havde været en fordel, hvis man oppe ved kurv ikonet satte en funktion op der talte op for hver cupcake man satte ned i indkøbskurven.

Demo video

Her bliver der vist hvordan applikationen ser ud.

https://youtu.be/Ve9DbCoy0H0

Proces

Plan

Ansvarsområder

Teamet har været inddelt i roller, således at hver person har haft et ansvarsområde for projektet.

- **Scrummaster**
  - Har planlagt og faciliteret Daily Stand-up, Planning og Retrospektiv
  - Har registreret og delt arbejdsskema og fravær
- **Product Owner**
  - Har designet Figma prototypen
  - Har prioriteret User Stories
- **Tech Lead**
  - Har været påkrævet Reviewer på Pull Requests

Rollerne har været fordelt således:

Navn	Sprint 1	Sprint 2

Abbas	Tech Lead	Scrum Master
Emil	Product Owner	Tech Lead
Frederik	Tech Lead	Product Owner
Thomas	Scrum Master	Tech Lead

Road map

Et sprint har forløbet sig over én uge, og i perioderne:

- **Sprint 1**: Mandag d. 24. Marts 2025 - Søndag. 30. Marts 2025
- **Sprint 2**: Mandag d. 31. Marts 2025 - Fredag d. 4. April 2025

Planen har været at færdiggøre webapplikationen i Sprint 1, således at Sprint 2 kan udnyttes til at skrive rapport.

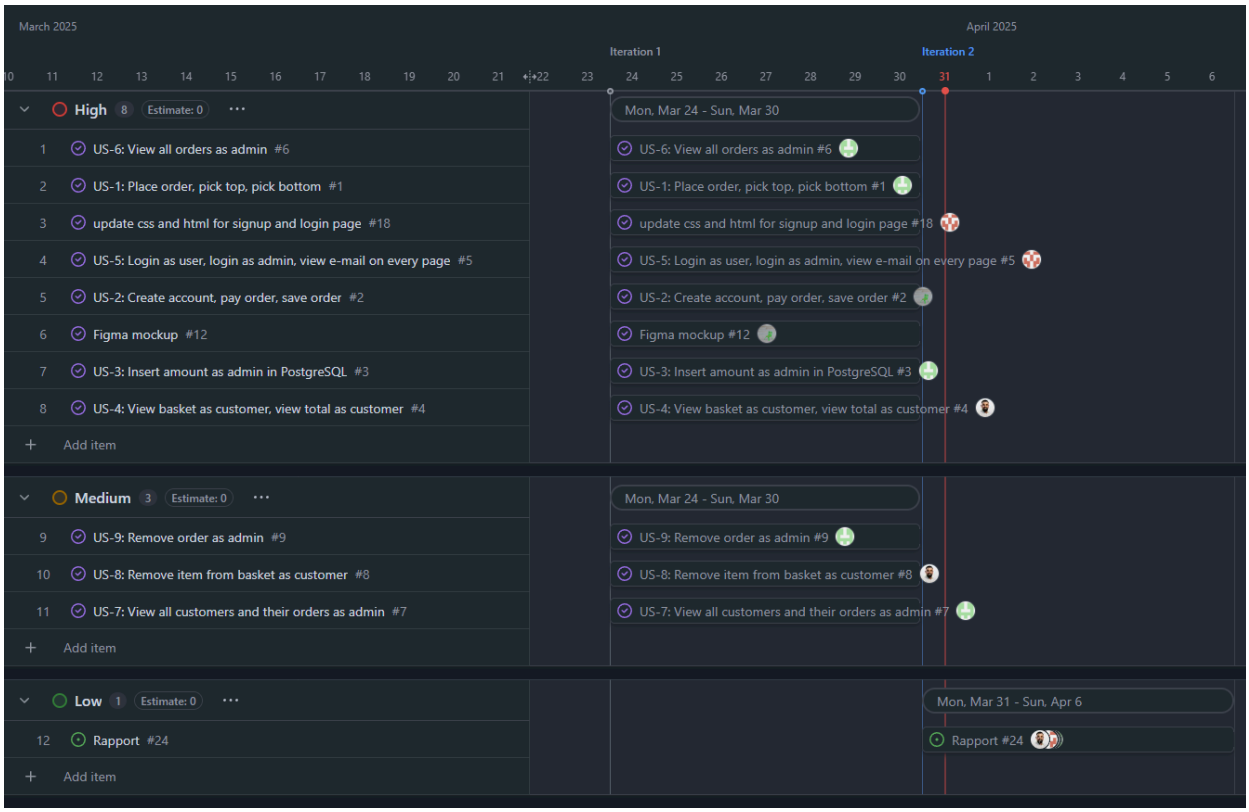

Fig. 5: Roadmappet viser hvordan opgaverne er fordelt over sprint iterationer

Team kapacitet

Alle opgaver har i fællesskab været estimeret i T-shirt størrelser. Estimering er sket via Planning Poker, som er foregået som følg.:

1. En User Story bliver læst op
2. Hvis der er usikkerhed om opgaven, stilles der spørgsmål
3. Alle vurderer hver især deres estimering i skjul
4. Når alle har vurderet, bliver alles estimering vist på samme tid
5. Hvis der er uenigheder om størrelsen, så skal de afvigende estimeringer argumentere for, hvorfor opgaven er større/mindre end hvad flertallet har vurderet
6. Når der er total, enighed, bliver estimeringen skrevet ind i opgaven

Figur 6 viser i **Size** kolonnen, hvad de enkelte opgaver er blevet vurderet til.

	Title	Priority	Status	Size	Iteration	Type	Assignees
	High 8 Estimate: 0						
1	US-6: View all orders as admin #6	High	Done	S	Iteration 1	Feature	Tatchapero
2	US-1: Place order, pick top, pick bottom #1	High	Done	L	Iteration 1	Feature	Tatchapero
3	update css and html for signup and login page #18	High	Done	S	Iteration 1	Bug	Frederik-BipBop
4	US-5: Login as user, login as admin, view e-mail on every page #5	High	Done	M	Iteration 1	Feature	Frederik-BipBop
5	US-2: Create account, pay order, save order #2	High	Done	L	Iteration 1	Feature	Iirne
6	Figma mockup #12	High	Done	M	Iteration 1	Task	Iirne
7	US-3: Insert amount as admin in PostgreSQL #3	High	Done	XS	Iteration 1	Feature	Tatchapero
8	US-4: View basket as customer, view total as customer #4	High	Done	M	Iteration 1	Feature	AbbasMB
+	Add item						
	Medium 3 Estimate: 0						
9	US-9: Remove order as admin #9	Medium	Done	S	Iteration 1	Feature	Tatchapero
10	US-8: Remove item from basket as customer #8	Medium	Done	S	Iteration 1	Feature	AbbasMB
11	US-7: View all customers and their orders as admin #7	Medium	Done	M	Iteration 1	Feature	Tatchapero
+	Add item						
	Low 1 Estimate: 0						
12	Rapport #24	Low	In Progress	L	Iteration 2	Task	AbbasMB, Frederik...
+	Add item						

Fig. 6: Billedet viser oversigt over arbejdsfordelingen af opgaverne

Praksis

Teamet har i fællesskab lavet:

- ERD
- Definition of Done
- Kode standarder
- Estimering af User Stories
- Rapport

Hvert teammedlem er blevet tildelt en User Story ved start af Sprint 1. Når en opgave blev løst, skulle man tage den øverste ledige opgave i backloggen, såfremt at den øverste opgave ikke var afhængig af en anden opgave, der var i gang. På den måde sikres det, at de højeste prioriterede opgaver bliver løst først.

Burn up chart

Burn up chart viser progressionen i projektet over tid, og viser hvor meget arbejde der er færdiggjort samt hvor meget arbejde der mangler.

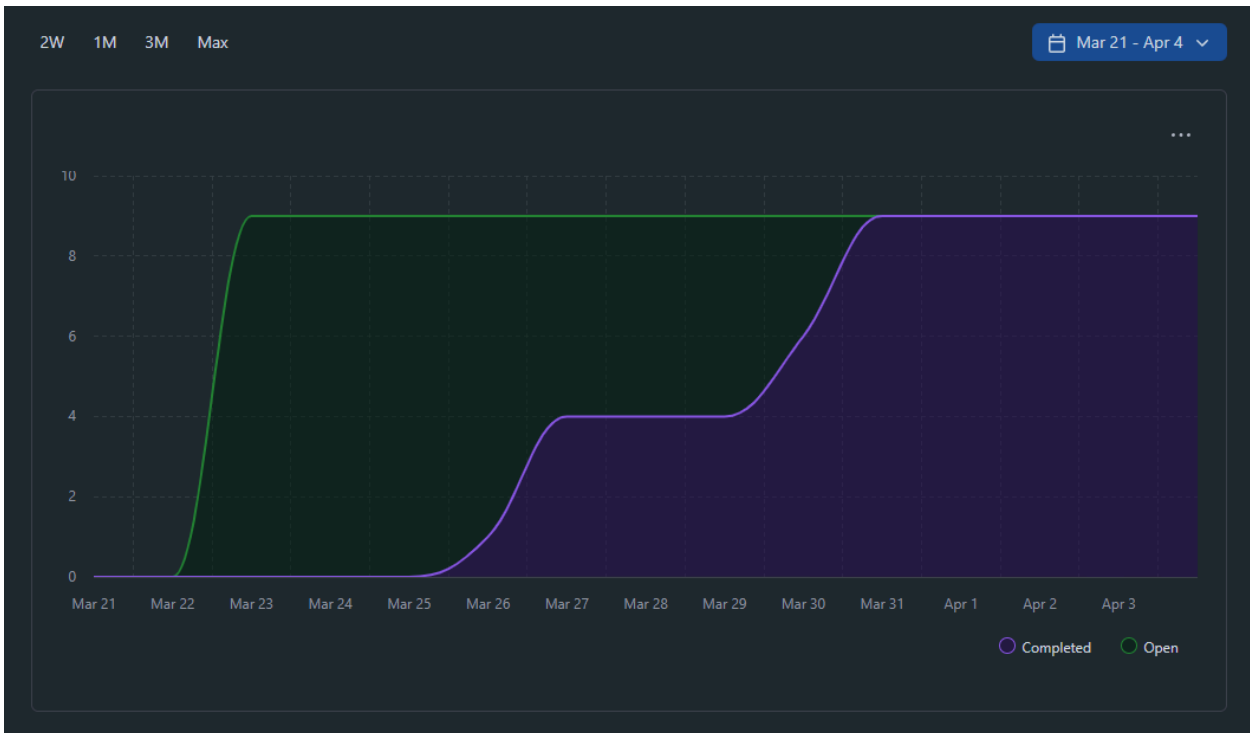

Fig. 7: Burn up chart viser oprettet og gennemførte opgaver over tid

Retrospektiv

Der er blevet afholdt retrospektiv-møde i slutningen af projektet, hvor der i fællesskab er blevet diskuteret hvordan processen kunne forbedres. Mødet har forløbet sig ved, at alle har haft 5 minutter til at skrive kommentarer til følgende kategorier:

- Hvad gik godt?
- Hvad kan forbedres?
- Hvad skal vi begynde at gøre næste gang?

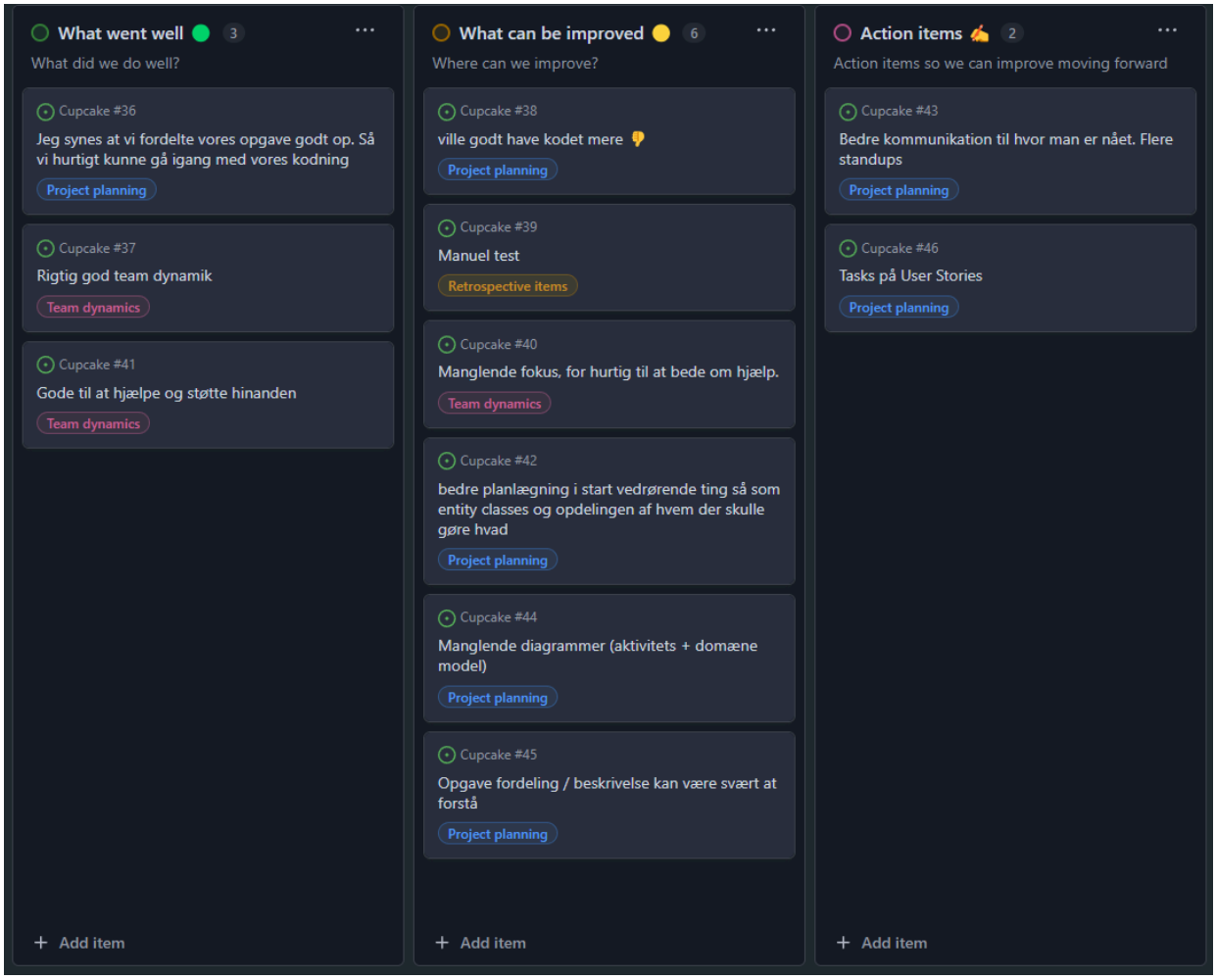

Fig. 8: Billedet viser retrospektive boardet

Efter de 5 minutter er gået, har mødefacilitator gennemgået hvert kommentar i fællesskab, og givet hvert medlem mulighed for at uddybe deres kommentar, som har skabt diskussion og idéer til hvordan processen kan forbedres fremadrettet.