Svømmeklubben Delfinen

ADMINISTRATIONSSYSTEM

Gruppe 1
Gustav, Mads, Sofie R, Tobias & Victor | Rapport | 13/12

Github brugernavne:

Gustav: *gustavtjac*
Sofie: *sobro002*
Mads: mcmadsv
Victor: *viictv*
Tobias: *mf-gua*

Github repo: https://github.com/gustavtjac/Sv-mmeklubDelfinen

Indledning

Vi skal udvikle et administrationssystem til svømmeklubben Delfinen. Systemet skal bruges af formanden, en kasserer samt en svømmetræner. Formanden skal kunne oprette nye medlemmer, kassereren tager sig kontingentbetalinger og svømmetræneren skal bruge en oversigt over de bedste konkurrencesvømmere i de forskellige svømmediscipliner. Klubben består af både motionister og konkurrencesvømmere inddelt i junior- og seniorkategori, som kontingentets pris afhænger af. Data skal opbevares sikkert og indlæses ved softwarestart.

METODE

Vi har brugt følgende modeller og diagrammer til at udvikle et administrationssystem til svømmeklubben.

I forbindelse med udviklingen af systemet, *kan* der opstå forskellige risici, som vi vha. en udvidet risikoanalyse har identificeret og forsøgt at håndtere, såfremt de realiseres. Interessenterne af systemet kan også udgøre en risiko, såfremt de har indflydelse på den strategiske proces eller er nødvendige for gennemførelsen af denne.

For at illustrere koncepterne i forretningen, har vi brugt en domænemodel.

Til selve udviklingsarbejdet af systemet, har vi brugt scrum som metode. Vi har udarbejdet to system-sekvensdiagrammer for at vise interaktionen mellem aktørerne og systemet, hvor klassediagrammet illustrerer designet og strukturen af vores kode. Med et sekvensdiagram viser vi hvordan objekterne interagerer med hinanden i tidsbestemt rækkefølge. Dette giver et overblik over hvorvidt GRASP-principperne opfyldes.

Analyse af svømmeklubben Delfinen

Vi har brugt følgende analyseværktøjer til at opnå en fyldestgørende forretningsforståelse, således det endelige produkt opfylder kundens forventninger.

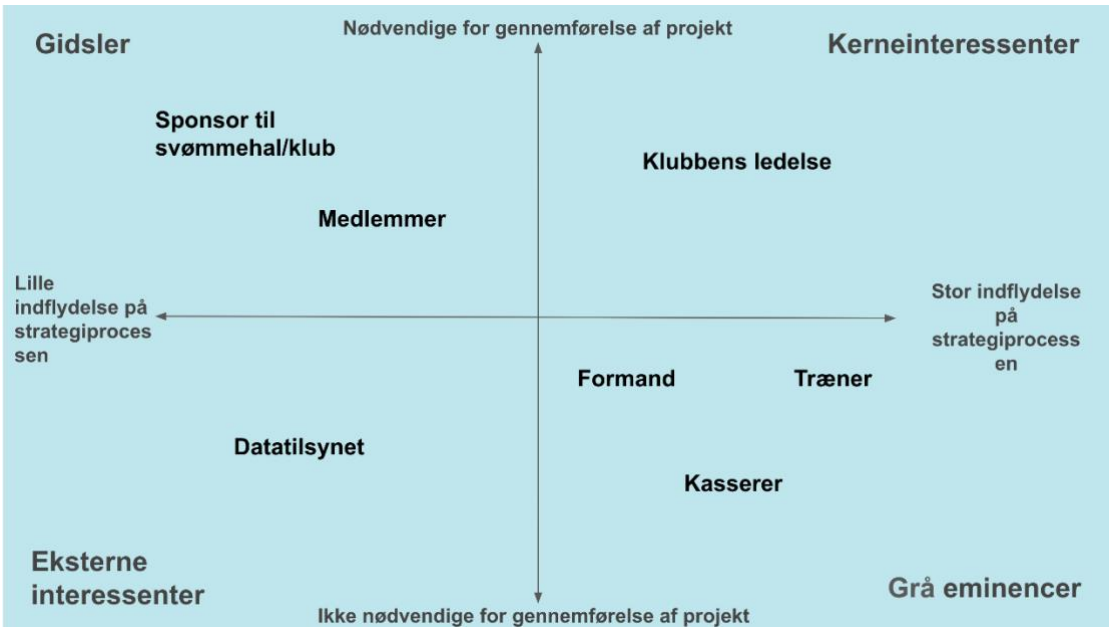
Forretningsforståelsen består først og fremmest i at analysere involverede interessenter vha. en interessentanalyse. Dernæst hvilke risici der kan opstå i forbindelse med projektet. Der skal foreligge en dokumentation for håndteringen af begge disse elementer.

Anden del af analysen består i at identificere, hvilke krav svømmeklubben har til deres kommende administrationssystem, som også visuelt illustreres i en domænemodel. Således søger vi at dokumentere, at vi forstår kundens forretning og krav.

Når kravene er identificeret, vil vi som sidste led i vores analyse udarbejde to use cases samt dertilhørende system-sekvensdiagrammer for at vise, hvad systemet skal kunne for at opfylde de respektive krav.

INTERESSENTANALYSE

For at analysere svømmeklubbens interessenter har vi lavet en interessentanalyse. Vi har først identificeret alle interessenter og opdelt dem i 4 grupper, som vist i diagrammet nedenfor. De vigtigste er kerneinteressenterne som både er nødvendige for gennemførelse af projektet og har stor indflydelse på strategiprocessen. Herefter kommer de "grå eminencer" som ikke er nødvendige for gennemførelsen af strategien, men som har stor indflydelse på strategien i og med de har interesse i den. Næstsidst kommer Datatilsynet som er dem der kontrollerer at reglerne for GDPR bliver overholdt, hvilket Delfinen vil være underlagt jævnfør alle de personlige oplysninger de vil komme til at håndtere. Datatilsynet er en "ekstern interessent" fordi de har en lille indflydelse på strategiprocessen og heller ikke er nødvendige for gennemførelsen. Til sidst er der "gidsler", som har en lille indflydelse på strategien, men som stadig er nødvendige for gennemførelsen. Hertil har vi valgt sponsorer og medlemmer.



Nedenunder er selve interessentanalysen hvor vi kigger nærmere på hvordan de forskellige interessenter både drager fordel af og potentielt kan opleve ulemper ved projektets gennemførelse. Formålet med interessentanalysen er at skabe en forståelse af interessenternes positioner, deres forventninger og de mulige konsekvenser, som projektet kan have for dem. Vi vurderer også igen nødvendigheden for hver interessent samt deres direkte indflydelse. Til sidst kommer vi med forslag til hvordan vi håndterer både fordele og ulemper bedst muligt i forhold til den enkelte interessent.

Interessent	Fordele ved projektet	Ulemper ved projektet	Vurdering af interessent	Håndtering
Ledelse	Forhåbentlig mere tilfredse medlemmer. Forhåbentligt mere tilfredse medarbejder. Effektivisering af arbejdsprocesser. Bedre overblik over økonomien og medlemmer	Ledelsen investerer et relativt stort beløb. Tilvænding til nyt system	Dem der har bestilt systemet. Deres krav vi skal opnå. Har en interesse i projektet fordi det forhåbentligt skal løse deres problemer. **- Høj nødvendighed** **-Høj indflydelse**	Hold ledelse opdateret regelmæssgt. Evt i mellem sprints. Hav shippable kode klar til fremvisning
Formand	Mere effektiv arbejdsproces	Skal lære det nye IT-system	Har direkte stillet krav. Ham der opretter medlemmer. **- Lav nødvendighed** **- Høj indflydelse**	Bringe ham ind til sprint-review specifikt når vi arbejder på at lave medlems-oprettelsesdelen
Træner	Mere effektiv arbejdsproces, Bedre visning af top svømmeelever Mulighed for at se en top 5 via systemet	Skal lære det nye IT-system	Har direkte stillet krav. Skal interagere med systemet. **- Lav nødvendighed** **- Høj indflydelse**	Bringe ham ind til sprint-review specifikt når vi arbejder på at designe konkurrencesvømmerdelen af projektet
Kasserer	Mere effektiv arbejdsproces, Automatiseret visning af forventet kontigent Automatiseret visning af forventet restance	Skal lære det nye IT-system	Har stillet direkte krav og dermed haft stor indflydelse på hvad vi skal kode. Skal direkte interagere med vores system. **- Lav nødvendighed** **- Høj indflydelse**	Bringe ham ind til sprint-review specifikt når vi arbejder på at designe økonomidelen af projektet
Medlemmer	Nemmere oprettelse Bedre overblik over svømmetider samt stævner.	Sværere ved at undgå restance betaling	Betaler kontingent for at bruge svømmehallen. Funderer ledelsen. **- Høj nødvendighed** **- Lav indflydelse**	Vi videregiver vores opdatering til ledelsen så de kan melde ud til medlemmerne
Sponsor	Måske flere kunder til svømmehallen som resultat af systemet, hvilket resulterer i mere eksponering til sponsoren. Svømmehal mere konkurrencedygtig og derfor et bedre sted at være sponsor	Højere forventning fra svømmehallen i det de er blevet "Opgraderet"	Supplerer med penge som gør det mulig for ledelsen at betale for systemet **- Høj nødvendighed** **- Lav indflydelse**	Videregive nødvendig information til ledelse, som derefter videregiver til eventuelle sponsorer.
Datatilsynet	Nemmere at følge med i om reglerne for GDPR bliver overholdt.		GDPR lovgining i forhold til data persistence. **- Lav nødvendighed** **- Lav indflydelse**	Vi videregiver vores opdatering til ledelsen så de kan melde ud til medlemmerne

RISIKOANALYSE

For at kunne få et bedre billede af de risici der kan forekomme under projektet, har vi lavet en SWOT-analyse der viser teamets interne styrker og svagheder samt de eksterne trusler og muligheder.

Interne Forhold	
Styrker	**Svagheder**
Forskellige mennesker (Forskellige egenskaber)	Potentielt dårlig fremmøde
Forskellige måder at arbejde på	Forskellige måder at arbejde på
Klar på at lærer(motiverede)	Pga. forskelligheder, er der mulighed for folk får uenigheder
Stort team til den givne opgave.	Ikke meget erfaring med at estimere tasks
	Måske Team er for stort til givne opgave.
	Lille erfaring med at kode større projekter
Eksterne Forhold	
Muligheder	**Trusler**
Undervisere	Mulig sygdom
CodeLab	Computerfejl(eks. Github nede, kode ikke opdateret)
ChatGPT / Copilot	
Medstuderende	

I vores udvidet risikotabel under har vi inddraget risikomomenter baseret på teamets egne svagheder samt i forhold til vores interessenter. Vi har vurderet sandsynlighed, konsekvens og produkt ud fra skemaet i ser under dette tekststykke. Det vigtigste at forstå er at jo højere produktet er, jo mere risikabelt er momentet. Vi har lavet præventive tiltag til hvert enkelt risikomoment der indeholder hvordan vi bedst muligt kan undgå at denne risiko hænder. Hvert præventive tiltag har nogle tilhørende ansvarlige som er de personer der skal sørge for at vi får udlevet disse tiltag. Hvis vi nu skulle være så uheldige at risikomomentet blev reelt har vi også lavet et løsningsforslag som beskriver hvordan vi har tænkt os at løse problemet. Løsningsforslagene har også en ansvarlig tilknyttet.

Sandsynlighed	Konsekvens					
	1	3	7	10		Produkt
1						1-5
2						6-10
3						11-14
4						15
5						
	Ubetydelig	Mindre	Større	Katastrofalt		
Risikograd	Mindre	Moderat	Høj	Kritisk		

Udvidet risikoanalyse over It-Projektet							
Risikomoment	**Sandsynlighed**	**Konsekvens**	**Produkt**	**Præventive tiltag**	**Ansvarlig**	**Løsningsforslag**	**Ansvarlig**
Deltager i teamet deltager ikke tilstrækkeligt	5	3	15	Sørge for et godt arbejdsmiljø, hvor vii alle har samme mål og alle er inkluderet.	Victor - SCRUM MASTER	Genovervejer arbejdsfordeling og opdaterer den overordnet plan. Eventuelt overarbejde hvis det resulterer i at tidsplanen skrider	Victor - SCRUM MASTER
Teamet miskommunikere og koder i forskellige retninger	3	1	3	God anvendelse af systemudviklingsmetoder med tilstrækkelig analyse (SCRUM) så alle i teamet kan forstå hvad vi skal til at kode. Hvis man er i tvil om det man laver er rigtigt så spørg ind til andre i teamet	Teamet	Kigge på hvad der er gået galt og kigge på den overordnet plan igen og sørge for der er en fælles forståelse.	Gustav - Ansvarlig for intern kvalitet
Fejlanalyse af fremstillede krav	2	7	14	Læs, forstå, løs. Tilstrækkelig analyse af det givne tekstykke, Fordel eventuelt de fundne krav i funktionelle og ikke-funktionelle og byg derefter videre til et FURPS+. Domæne-model til at visualisere krav.	Teamet	Finde ud af hvad der er gået galt samt hvorfor det er gået galt. Bruge samme værktøjer med ny viden til at opnå bedre forståelse af kravene.	Tobias - Product Owner
Fejlestimering af task, Burndown-chart passer ikke	5	1	5	Brug forskellige metoder til at komme frem til estimering fx. Planning Poker og Three Point Estimation	Teamet + SCRUM MASTER til at facilitere værktøjerne	For hver Sprint opnår vi mere viden i forhold til hvor lang tid ting tager. Brug denne viden til at reestimere med samme metoder	Teamet - SCRUM MASTER
Risiko for social-loafing	5	3	15	Giv alle mulighed for at arbejde på projektet. Hjælp eventuelt teammedlem med at komme i gang hvis de har svært ved det. Højt humør	Teamet + SCRUM MASTER	Hvis loaferen selv er villig vil et teammedlem eller SCRUM MASTER høre loaferens forklaring på hvorfor de loafer. De kan bruge denne iformation til at imødekomme deres behov og	Teamet
Risiko for at Medarbejdere ikke har nok teknisk erfaring til at benytte sig af systemet	1	3	3	Designer systemet til at være brugervenligt og meget ligetil	Team + Product Owner	Allerede imellem sprints kan vi inviterer medarbejderne til Sprint reviews og der viser og forklarer hvordan systemet virker	SCRUM MASTER til facilitering + Teamet til at forklarer
Risiko for at Formand ikke er tilfreds med systemet	1	10	10	Bringer Formand ind til Sprint review, så han kan følge med i hvordan vi designer. Udføre ordenligt analyse af krav for at forstå hvad han beder om	Tobias - Product Owner	Hivs Formanden er utilfreds må vi høre ham ud og på baggrund af den information ligge en plan på hvordan vi kan fixe problemet	SCRUM MASTER + Team

KRAVSPECIFIKATION

For at kunne indfri kundens forventninger til det kommende administrationssystem, skal kravene identificeres og specificeres. Vi har identificeret kravene vha. FURPS+ og udformning af user stories. Alle user stories er anført i product backlog under afsnittet Udviklingsfasen.

FURPS+

Vi har identificeret kundens krav ved at bruge FURPS+ som tjekliste. Følgende systemkrav: Funktionalitet (F), brugervenlighed (U), pålidelighed (R), ydeevne (P) og supportability (S) ses nedenfor. Da kunden ikke har specificeret krav til ydeevne eller krav herudover (+) er disse ikke anført.

	Krav
F	- Oprette medlemmer. - Medlem består af alder, aktivitetsform og kontingent. - Aktivitetsformer: aktiv/passiv, junior/senior, motionist/konkurrencesvømmer. - Kontingentets størrelse afhænger af aktivitetsformen, herunder aktiv/passiv og junior/senior/pensionist. - Svømmediscipliner for konkurrencesvømmere: butterfly, crawl, rygcrawl og brystsvømning. - Data for konkurrencesvømmere skal gemmes og opdateres ved bedre resultater. Dataene består af stævne, placering og tid samt træningsresultat. - Oversigt over top fem bedste konkurrencesvømmere i hver svømmedisciplin inddelt i junior og senior. - Oversigt over forventede totale indbetaling af kontingent og medlemmer i restance.
U	- Tekstbaseret system der kører i terminalen.
R	- Følgende oplysninger skal kunne tilgås: medlemsoplysninger, samlede kontingentindbetaling, medlemmer i restance samt bedste resultater for konkurrencesvømmere.
P	
S	- Skalerbar, herunder genbrugelig, vedligeholdelsesvenlig samt robust kode.

Domænemodel

For at illustrere koncepterne til det administrative system, har vi udarbejdet en domænemodel. Ved brug af konceptuelle klasser, kan vi nemmere få et visuelt overblik over hvordan entiteterne associeres med hinanden samt deres antal.

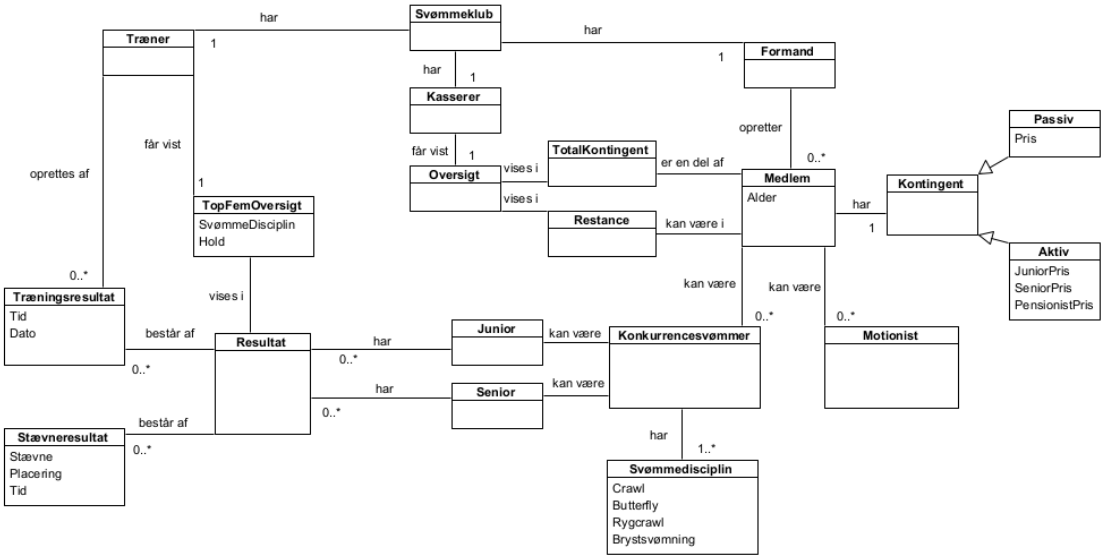
Domænemodellen viser at forretningen svømmeklubben Delfinen har tre aktører, der direkte interagerer med systemet.

Den første aktør er en formand, som opretter medlemmer. Et medlem har en alder og et kontingent, som afhænger af følgende forhold: hvorledes medlemmet er aktivt eller passivt; hvis medlemmet er aktivt, beregnes kontingentet ud fra alder inddelt i junior,

senior eller pensionist. Derudover er medlemmet enten konkurrencesvømmer eller motionist.

Konkurrencesvømmere er tilknyttet en eller flere svømmediscipliner: crawl, butterfly, rygcrawl eller brystsvømning. De er ydermere inddelt i enten junior- eller senior hold med et dertilhørende resultat.

Resultatet kan være et stævneresultat og/eller træningsresultat. Stævneresultatet har egenskaberne stævne, placering og tid. Træningsresultat har en tid og dato. Det er systemets anden aktør, *træneren*, som opretter træningsresultaterne. *Begge* typer resultater vises i en top-fem oversigt over bedste svømmere inddelt i hold samt svømmediscipliner.



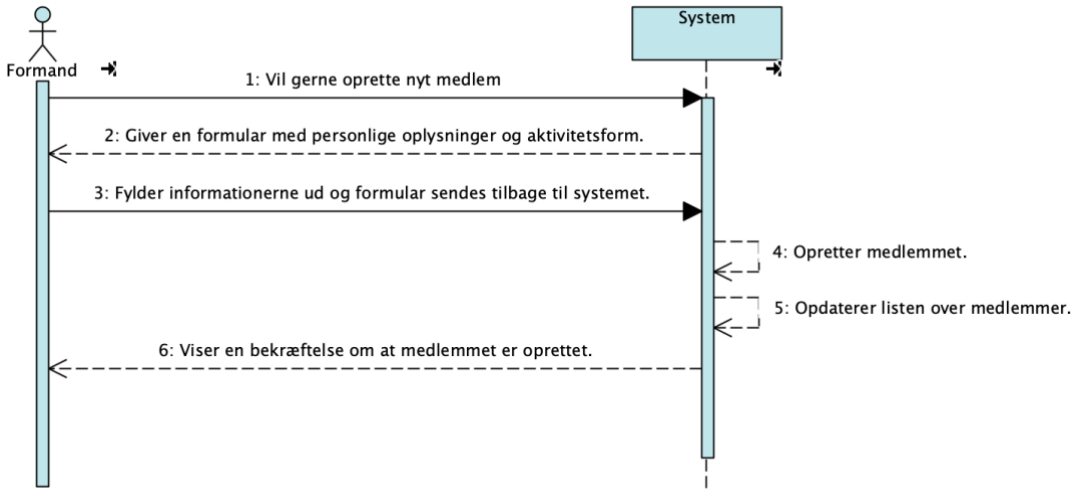
HVAD SKAL SYSTEMET KUNNE?

For at beskrive hvad systemet gør ud fra aktørernes perspektiv, har vi udarbejdet to fully-dressed use cases, som hver illustreres i et dertilhørende system-sekvensdiagram.

Use Case 1

	Opretter medlem
Primær aktør	- Formand.
Interessenter	- Medlemmer. - Træner, såfremt medlemmet er en konkurrencesvømmer. - Kassereren, da medlemmet har indflydelse på det samlede kontingent
Forudsætninger	- Kunde vil gerne være medlem.
Hovedsucces scenarie	- Formand vil gerne oprette et nyt medlem. - Systemet giver en formular med personlige oplysninger og aktivitetsform. - Formand indtaster information. - Systemet opretter medlemmet. - Systemet sender en kvittering for oprettelse af medlem
Alternativt scenarie	- Medlem ombestemmer sig. - Formanden udfylder formularen forkert.
Efterbetingelser	- Systemet opdaterer listen over medlemmer.
Specielle krav	- Computer.

System-sekvensdiagram 1

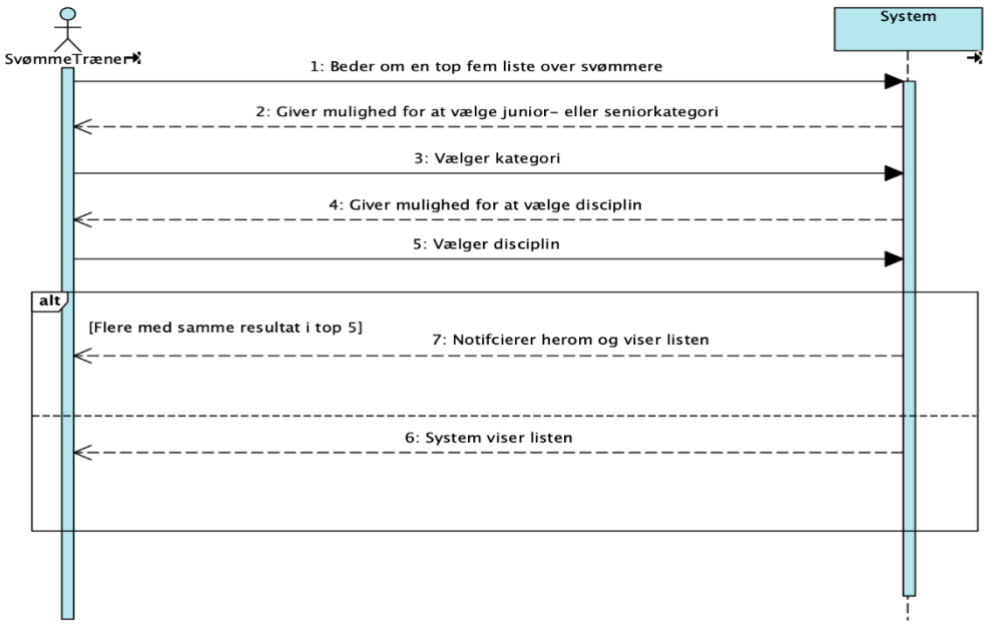


Use Case 2

	Vis top fem bedste konkurrencesvømmere
Primær aktør	- **Træner.**
Interessenter	- **Konkurrencesvømmere**: Vil gerne kunne se deres resultater. - **Sponsorere**, **agenter**, **investorer**.
Forudsætninger	- Konkurrencesvømmere skal være oprettet i systemet samt have et stævneresultat
Hovedsucces scenarie	- Træneren beder system om top fem liste over bedste konkurrencesvømmere. - Systemet giver mulighed for at vælge junior- eller seniorkategori. - Træneren vælger kategori. - Systemet giver mulighed for at vælge svømmedisciplin - Træneren vælger svømmedisciplin. - System viser listen baseret på kategori og svømmedisciplin. - Systemet viser listen.
Alternativt scenarie	- Der er to eller flere med samme resultat som kvalificerer sig til top fem listen.
Efterbetingelser	- Top fem listerne gemmes korrekt i systemet så træneren kan tilgå dem.
Specielle krav	- Computer.

System-sekvensdiagram 2

Systemsekvensdiagrammet viser interaktionen mellem svømmetræneren og administrationssystemet. Hovedsuccesen er at systemet viser en top fem liste over bedste konkurrencesvømmere. Såfremt der er mere end fem konkurrencesvømmere med samme resultat, notificerer systemet herom og viser fortsat listen (alternative flow).



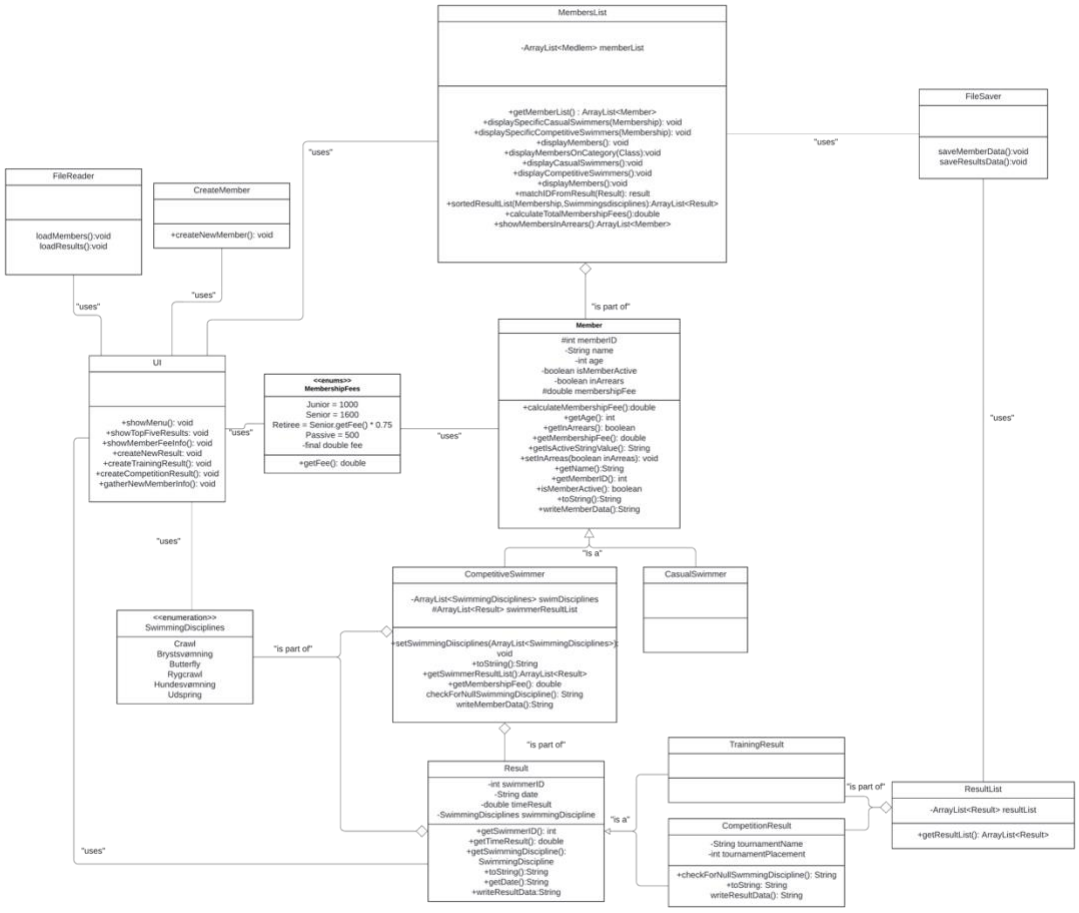
Design af system

Vi har designet administrationssystemet ved at modellere de klasser samt deres indbyrdes relationer i et klassediagram, som systemet skal indeholde. Vi har udarbejdet den overordnede struktur vha. diagrammet før første sprint. Den er efterfølgende blevet opdateret løbende under hvert sprint efterhånden, som vi påbegyndte en ny opgave.

For at skabe en fælles forståelse af, hvorledes systemet fungerer, har vi udarbejdet to sekvensdiagrammer. Diagrammerne illustrerer hvorledes en bruger oprettes i systemet samt hvordan en top fem liste over bedste konkurrencesvømmere vises.

KLASSEDIAGRAM

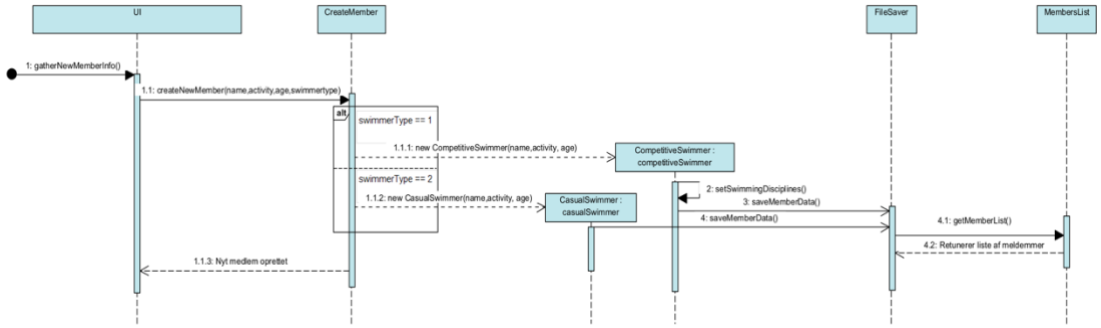
Dette klassediagram viser strukturen i systemet ved at illustrere klasserne, deres attributter, metoder og forhold. Klassediagrammet er en vigtig del af systemets design, da det giver et overblik over, hvordan data og funktionalitet er organiseret.



SEKVENSDIAGRAM 1

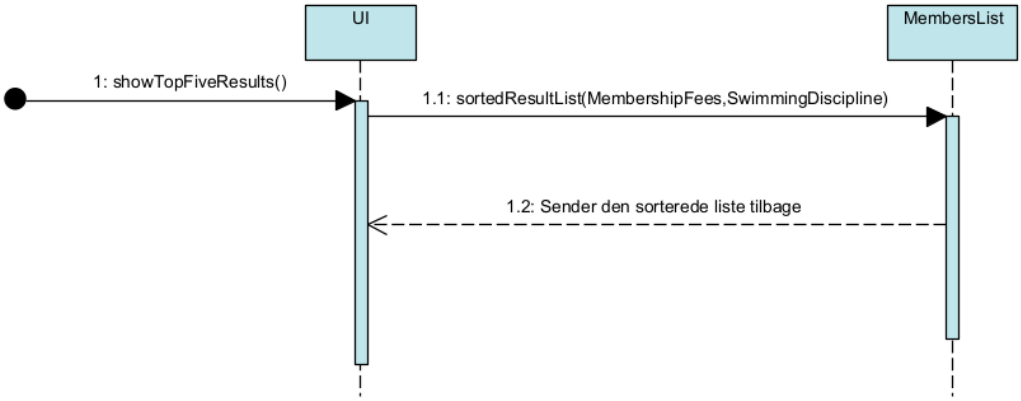
Brugeren ønsker at oprette et nyt medlem. Klassen "UI" kalder derfor metoden "gatherNewMemberInfo" fra "UI"-klassen. Efter den har indsamlet data, kalder den metoden "createNewMember(data 1,data 2,data 3,data 4)" som ligger i

"createMember"-klassen. Alt efter hvad swimmerType er lig med oprettets enten et "competitiveSwimmer"-objekt eller "casualSwimmer"-objekt. Hvis objektet er en "competitiveSwimmer" kaldes metoden "setSwimmingDisciplines()" som ligger inde i "CompetitiveSwimmer"-klassen. Derefter kaldes "saveMemberData()" metoden fra "FileSaver"-klassen som så kalder "getMemberList()" metoden fra "MembersList"-klassen. Til sidst retunerer "CreateMember"-klasssen til "UI"-klassen at medlemmet er oprettet.



SEKVENSDIAGRAM 2

Træner ønsker at se de fem bedste resultater inden for en nested kategori, f.eks. "Crawl -> Junior". ShowTopFiveResults kaldes fra hovedmenuen, hvorefter systemet beder brugeren indtaste den ønskede svømmedisciplin efterfulgt af medlemstypen. Disse to svar, sætter sortedResultList(input1, input2) igang fra MembersList klassen som præsenterer en tidssorteret liste for brugeren i den specificerede kategori.



Udviklingsfasen

Vi har brugt scrum som udviklingsmetode med følgende rollefordeling:

Product owner: Tobias
Scrum master: Victor

Den ansvarlige for ekstern kvalitet: Sofie
Den ansvarlige for intern kvalitet: Gustav

Produktmålet er at udvikle et administrativt system der skal imødekomme svømmeklubbens behov og opfylde de fastlagte krav. I det efterfølgende afsnit er disse krav fremsat i en product backlog. Product backlog bruges til at definere, hvad der vil opfylde produktmålet ud fra user stories.

Sprint backloggen består af målet for det pågældende sprint samt de product backlog items denne indeholder. Disse er nedbrudt i mindre opgaver med et tidsestimat for, hvor lang tid det tager, at løse den enkelte opgave. Når sprintmålet er nået, har vi opnået "definition of done", og systemet er derfor i inkrementel udvikling.

Summen af den estimerede tid vises i et burndown chart under det enkelte sprint.

PRODUCT BACKLOG

Nedenstående kursiverede user stories er prioriteret i rækkefølge ud fra, hvornår hver user story skal programmeres. Dvs. den første user story er af højeste prioritet og så fremdeles.

1. *Som formand i svømmeklubben vil jeg gerne kunne oprette nye medlemmer i systemet med tilknyttet alder, ønsket aktivitetsform og medlemspris, så jeg kan få et overblik over den enkelte kunde.*

2. *Som kasserer i svømmeklubben vil jeg gerne have at medlemmerne, som bliver oprettet, automatisk får udregnet den rigtige kontingentpris ud fra medlemmets oplysninger, herunder aktivt/passivt medlemskab og alder.*

3. *Som træner vil jeg gerne kunne oprette konkurrencesvømmernes resultater så som "stævne", "tid" og "placering" efter de har været til stævne.*

4. *Som træner i svømmeklubben vil jeg kunne se en top fem liste over de bedste konkurrencesvømmere (inddelt i junior og senior samt svømmedisciplin) for at kunne udtage svømmere til deltagelse i konkurrencer.*

5. *Som kasserer vil jeg kunne se en oversigt over forventede kontingentbetaling og regnskab for at kunne danne et overblik.*

7. *Som kasserer ønsker jeg en oversigt over medlemmerne for at kunne se, hvilke medlemmer, der er i restance.*

8. *Som bruger af systemet vil jeg have at systemet skriver og gemmer data sikkert, så i tilfælde af sammenbrud vil data ikke blive tabt.*

SPRINT BACKLOG

Sprint backloggen er opbygget i en tabel for hvert sprint. Tabellen indeholder sprintmålet for de respektive user stories med dertilhørende opgaver. Tidsestimatet er angivet uden for den enkelte opgave angivet i story points. Hvert story point svarer til 15 min. Summen af tidsestimaterne er vist i et burndown chart under tabellen. Da vi igangsætter et nyt sprint hver dag, afholdes daily standup inden. Her planlægger vi, hvordan vi har til hensigt at gennemføre det pågældende sprint.

Efter hvert sprint har vi afholdt sprint review for at gennemgå det pågældende arbejde. Selve refleksionen over arbejdet er sket under sprint retrospektiv for at identificere styrker og svagheder i processen, således denne kan blive forbedret ved at tilpasse product backlog til det efterfølgende sprint.

Første sprint

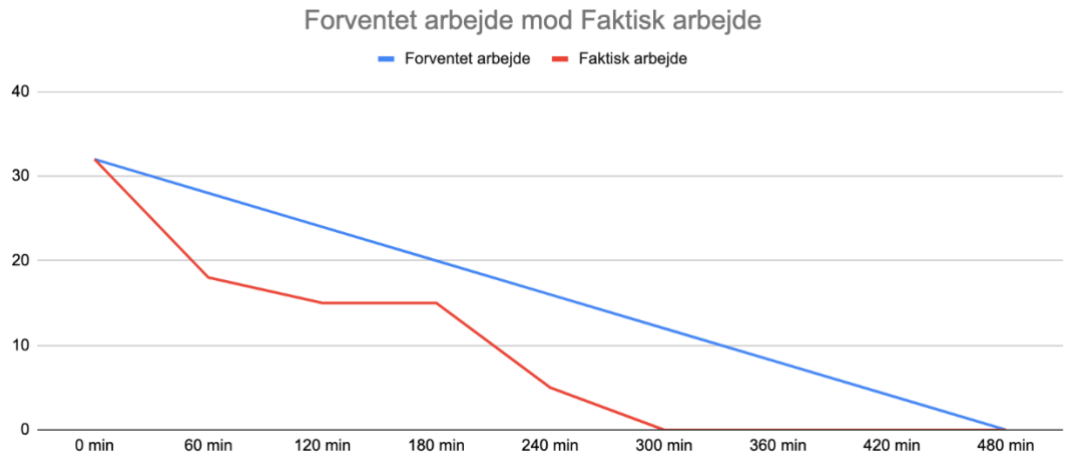
09/12-24

Sprintmål	Opgaver	Tidsestimat
Oprettelse af medlemmer	1. Opret superklassen "Member" med attributterne: ID, name, age, isMemberActive, inArrears, membershipFee. 2. Lav konstruktør med parametrerne: name, age og isMemberActive. 3. Opret subklasserne "CompetitiveSwimmer" og "CasualSwimmer" til "Member"-superklassen. "CompetitiveSwimmer" skal indeholde en arrayList af "swimmingDisciplines" samt en parametriseret konstruktør med name, isMemberActive og age. 4. Opret enum "MembershipFees" med værdierne junior(1000), senior(1600), retiree(1600*0.75) og passive(500) samt en getter-metode til at hente enum-værdierne. 5. Opret en klasse "CreateMember" med en metode, således brugeren kan indtaste medlemsoplysninger for et nyt medlem. Medlemsoplysningerne er navn, alder, aktiv/passiv, konkurrencesvømmer/motionist. Opret en metode "setSwimmingDisciplines" i "CompetitiveSwimmer"-klassen for at kunne tildele svømmedisciplin til konkurrencesvømmere. Håndtér også ugyldigt input.	2 1 3 5 5
Håndtering af medlemskontingent	1. I klassen "Member": opret metode "calculateMembershipFee" til at beregne kontingent ud fra alder samt aktivitetsformen fra enummen "MembershipFees". Tildel objektet "membershipFee" returneringsværdien af metoden.	5
Oprettelse af konkurrencesvøm-mernes resultater	1. Opret en klasse "CompetitionResult" med attributterne swimmerID, tournamentName, date, timeResult, tournamentPlacement og swimmingDisciplines. Klassen indeholder en parametriseret konstruktør med tournamentName, date, timeResult, tournamentPlace, swimmingDisciplines, swimmerID). 2. Opret klassen "CreateResult" med en metode således brugeren kan oprette et nyt resultat inkl. brugerinteraktion. 3. Tilføj en toString metode til et medlem og competitiveSwimmer der viser dem visuelt.	5 5 1

Daily Standup

Vi aftaler at opgaverne i første sprint kodes sammen, da attribut- og metodenavne ikke er kortlagt. Scrum master flytter opgaverne i sprint backlog i Trello. Hele udviklerteamet opdaterer klassediagrammet i forhold til klasser, attributter og metoder med GRASPs principper in mente.

Burndown Chart



Sprint Review

Vi er blevet færdige med første sprint, hvor basisfundamentet af funktionaliteten er lagt til næstkommende sprint. Alle tasks er testet og de respektive sprintmål er opnået. Vi har fulgt GRASP's principper ved at uddele ansvaret på forskellige klasser for at opnå *high cohesion* og *low coupling*.

Sprint Retrospektiv

Vores opgaver var ikke tilstrækkeligt nedbrudt. Det resulterede i, at vi undervejs skulle finde løsninger, således opgaverne kunne udføres. Derudover var det svært for alle gruppemedlemmer at følge koden, fordi planen ikke var nedskrevet.

I forhold til estimering af opgaverne, endte vores burndown chart ikke med at passe. Summen af tidsestimaterne var tre timer højere end den tid, vi reelt endte med at bruge på at kode.

Til næste sprint skal de respektive user stories nedbrydes yderligere, således opgaverne dækker sprintmålet fyldestgørende. Derudover skal nuværende opgaver re-estimeres og nye opgaver tidsestimeres.

Andet sprint

10/12-24

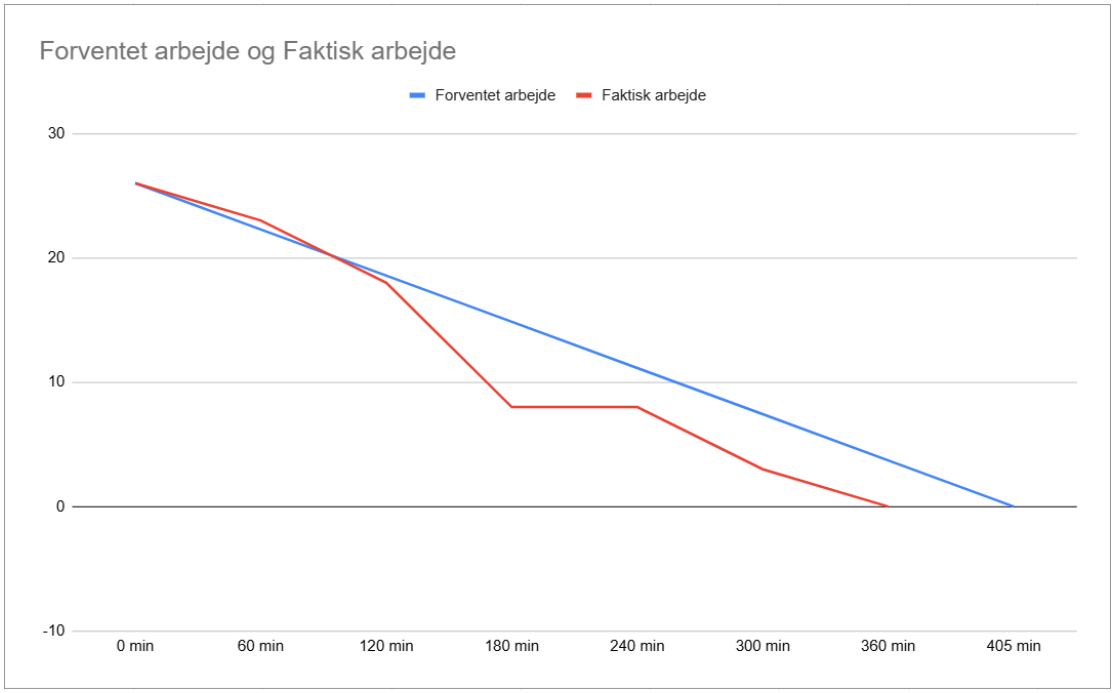
Sprintmål	Opgaver	Tidsestimat
Top fem liste over bedste konkurrencesvømmere	1. Opret "MembersList"-klasse med en statisk ArrayListe attribut som tager medlemsobjekter ind. 2. Tilføj i "Member"-klasse konstruktøren så hver instans af et medlem automatisk bliver tilføjet til den statiske MedlemsListe ArrayList. 3. Opret klassen ResultList der indeholder en statisk arraylist af CompetetionResults. 4. Opret i konstruktøren af Result At de automatisk skal tilføjes til ResultList. 5. Lav en metode som beregner og viser de top 5 svømmere indenfor alle 4 kategorier og både for senior og junior. 6. Lav i "Result"-klassen en showTop5Results som bliver kaldt i UI klassen når menuen "Vis top 5 konkurrenceresultater" bliver valgt. I metoden kan man specificere aldersgruppe og disciplin for at få vist en mere specifik top 5 resultatliste.	2 2 5 5 1 1
Oprettelse af brugergrænseflade	1. Opret klassen "UI" med hovedmenu som bliver kaldt ved programstart med submenuerne: "Opret medlem", "Opret resultat", "Vis medlemmer", "Vis top 5 konkurrence resultater", "Vis kontingentinformation", "Luk program". 2. Opret en metode i MemberList klassen: "displayMembers" som herefter kaldes inde i UI klassen når "Vis medlemmer" bliver valgt. Når denne bliver kaldt skal følgende valgmuligheder vises og følgende sorteringsmetoder laves: "Alle medlemmer -> Aldersgruppe/Alle", "Konkurrencesvømmere -> Aldersgruppe/Alle", "Motionist -> Aldersgruppe/Alle".	2 7

Daily Standup

Vi starter dagen ud med at tilføje opgaver til dagens sprint samt tidsestimere dem. Derudover skal vi planlægge, hvordan de næste programmeringsopgaver skal uddelegeres til det enkelte teammedlem vha. Github.

Hvert teammedlem påbegynder opgaver og flytter dem selv i Trello på vores scrum board. Forventede kontingentindbetaling skal unit testes.

Burndown Chart



Sprint Review

Vi har opnået alle sprintmål, således at vores MemberList klasse er klargjort, samt funktionalitet som visning af de fem bedste konkurrencesvømmere. Det var svært at adskille listen over top fem konkurrencesvømmere fra deres respektive kategorier, da vi var nødt til at benytte os af et nyt værktøj "instanceof " operatoren for at tilgå "CompetitiveSwimmer" i vores MedlemsListe bestående af superklassen "Member".

Sprint Retrospektiv

Det lykkedes os at bryde sprintmålene ned i flere opgaver, men det viste sig ikke at være tilstrækkeligt, da klasserne løbende opnåede flere koblinger. Vi var derfor nødt til at gentænke vores kodestruktur og opnåede derfor ikke *low coupling*, som først antaget. Product backlog opdateres, således implementeringen af metode til at lagre og indlæse data i programmet er sidstprioriteret. Tredje og sidste sprint bliver løst på samme vis, da vi i andet sprint fortsatte ud fra struktur.

Tredje sprint

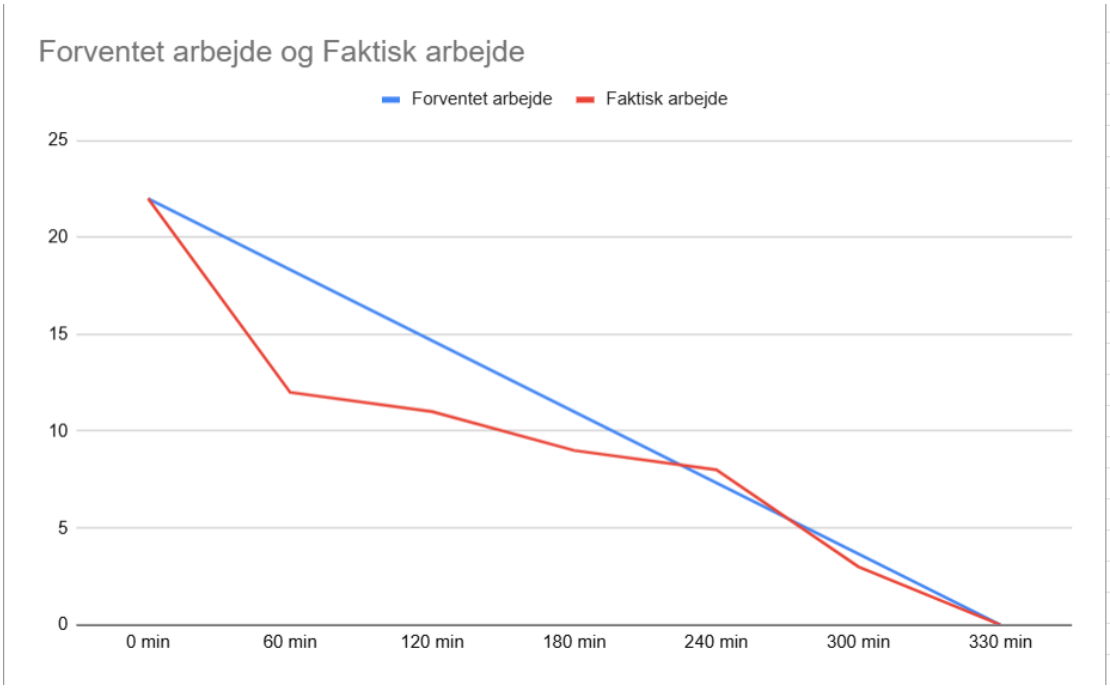
11/12-24

Sprintmål	Opgaver	Tidsestimat
Oversigt over medlemmer i restance samt samlede kontingent	1. Lav en metode displayExpectedTotalFees som looper gennem MedlemsListe ArrayListen og lægger alle medlemsPriserne sammen i en sum og returnerer denne. 2. Opret en metode der looper igennem medlemlisten i "MedlemsListe"-klassen og viser hvilke medlemmer der er i restance.	5 3
Implementering af datalagring og indlæsning	1. Opret en klasse der skal være ansvarlig for at gemme alle medlemsobjekter til en fil på computeren. Sørg for at køre denne metode inden programmet lukkes. 2. Opret klasse der skal indlæse alle medlemsobjekter og resultatobjekter fra filen og på den måde instantiere dem igen ved programstart.	7 7

Daily Standup

Tredje og sidste sprint kodes i fællesskab vha. "Code With Me", således vi kan arbejde i samme klasser og følge hinandens kode, da opgaverne overlapper hinanden. Scrum master flytter opgaverne i Trello på vores scrum board.

Burndown Chart



Sprint Review

I 3. sprint får vi gjort systemet helt færdigt. Det der tog længst tid i tredje sprint var at sørge for at alle medlemsobjekter bliver gemt på en fil, i tilfælde af systemnedbrud eller bare slukning. Til sidst lavede vi også en klasse som indlæser medlems- og resultatobjekter fra filen ved programstart.

Sprint Retrospektiv

Det første sprint mål "Oversigt over medlemmer i restance samt samlede kontingent" viste sig at være lige til og blev overestimeret en smule. Denne overestimering viste sig dog at være sund for den samlede estimering, da næste sprintmål krævede mere problemløsning end vi først havde antaget.

Konklusion

Systemet lever op til FURPS+-modellen og GRASP-principperne gennem en række nøglefunktioner og designvalg. Funktionalitetsmæssigt understøtter systemet oprettelse og vedligeholdelse af medlemsoplysninger, beregning af kontingent, registrering af svømmeresultater og visning af de fem bedste svømmere. Brugervenligheden sikres med en tekstbaseret brugergrænseflade, som er enkel og nem at navigere, med en konsistent menuopbygning og inputmetoder gennem hele systemet.

For at sikre pålidelighed er der implementeret fejlhåndtering i inputmetoderne, hvilket forhindrer, at systemet crasher ved fejlinput. Systemet er desuden forbedret med FileReader og FileSaver, der gemmer medlems- og resultatoplysninger i tilfælde af systemnedbrud. Ydeevnen er optimal på grund af den simple tekstbaserede arkitektur, som reagerer hurtigt på brugerinput og håndterer mange medlems- og resultatoplysninger uden væsentlige problemer.

Supportabiliteten er sikret gennem struktureret og kommenteret kode, som gør vedligeholdelse og opdatering lettere. Systemet er designet med klasser og metoder, der kan udvides, og encapsulation sikrer, at data kun kan ændres via definerede metoder, hvilket forbedrer datasikkerheden. Systemet er også designet med en klar opdeling af ansvar mellem klasser og metoder, hvilket gør implementeringen let at forstå og i overensstemmelse med almindelig Java-praksis.

I forhold til GRASP-principperne demonstrerer systemet generalitet ved at håndtere forskellige typer medlemmer og resultater, mens pålideligheden styrkes gennem robust fejlhåndtering. Tilpasningsdygtigheden fremmes af en modulær struktur med separate klasser for forskellige funktioner, som gør det nemt at tilføje eller ændre funktionalitet. Skalerbarheden er designet til at håndtere stigende datamængder, men systemets konsolapplikation kan begrænse det, hvis datamængden bliver markant større. Systemets præstationer forbliver dog stabile med hurtige svartider og minimalt ressourceforbrug.

I projektets udviklingsfase blev SCRUM anvendt som metode, hvilket inkluderede daglige stand-up møder, der fremmede kommunikationen, skabte overblik over fremdrift og identificerede potentielle hindringer. For at overvåge fremdriften i sprintene blev der anvendt burndown charts, som i de første to sprints afslørede

fejlestimater, men i det sidste sprint stemte chartet overens med det endelige tidsestimat.

Samlet set lever systemet op til FURPS+-modellen og GRASP-principperne ved at tilbyde en funktionel, brugervenlig, pålidelig og supportabel løsning. Udviklingsmetoden SCRUM og brugen af burndown charts har yderligere styrket projektets struktur og fremdrift. Dog er der plads til forbedring i skalerbarheden, hvilket kan blive et problem når datamængden vokser betydeligt.

Glossary

Sprint: En sprint er en fast tidsperiode, typisk på 1-4 uger, hvor et Scrum-team arbejder på en veldefineret mængde opgaver med målet om at levere et konkret og brugbart resultat. Sprinten afsluttes altid med en leverance, som kan være en funktion, et produkt, eller en del af en løsning, som er klar til brug og eller evaluering.

Domænemodel: Formålet med en domænemodel er at skabe en fælles forståelse blandt udviklere og andre interessenter af det problemområde, som systemet skal dække.

Sekvensdiagram: Et sekvensdiagram er en type diagram, som bruges i softwareudvikling. Den bruges til at beskrive, hvordan objekter, aktører eller systemkomponenter interagerer med hinanden over tid. Den viser rækkefølgen, hvori beskeder eller handlinger sendes mellem forskellige klasser eller objekter og systemet i et bestemt scenarie.