Delfinen Rapport

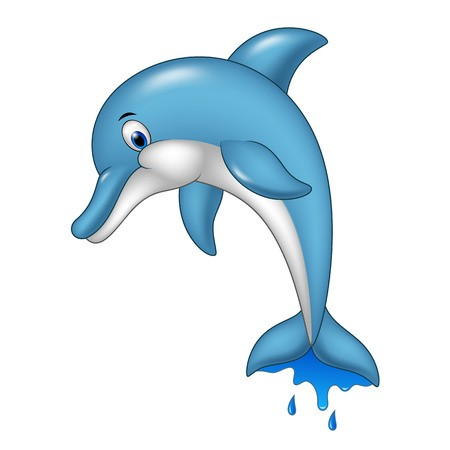
Gruppe:  
Jannick Lund-Pedersen

Mikkel Kophmann Palm

Anton Hulbæk Haastrup

Joakim Thorum Poulsen

# 



[**Resume:**](#_dmvo44pyfwnd) **3**

[**Vision:**](#_u372ti1f4eqq) **3**

[**Gantt:**](#_dvfemd94mxti) **4**

[**Kravspecifikation:**](#_g1yceg79825m) **5**

[1.0 Funktionelle krav:](#_yrwv3qytcem6) 5

[2.0 Anvendelighedskrav:](#_cq767os8naol) 5

[3.0 Supportability krav:](#_dw9j0z5mrqwc) 6

[4.0 Yderligere ikke-funktionelle krav:](#_so7ltxslfgv2) 6

[**Design**](#_uw0il58li8l1) **7**

[Noun Verb analyse:](#_et3f2do86xm3) 7

[Brief Use Case:](#_2gp3nx56jbm5) 8

[Casual Use Case:](#_w91me980ppf2) 9

[Use Case Diagram:](#_1jj67zlhqmfe) 11

[Chairman:](#_769widwgqr1l) 11

[Cashier:](#_kfn5vf1wdark) 11

[Coach:](#_qq2ectshxtuz) 12

[Domænemodel:](#_m0eviop0ns83) 12

[Klassediagram:](#_et0s6rbehfu8) 13

[System Sequence Diagram:](#_29hzeiyafo4w) 14

[**Konstruktion**](#_ekxux6nwh3w4) **15**

[Kode Dokumentation:](#_bc7hx06kjv4u) 15

[Delfinen:](#_zfvo4txp9fyn) 15

[Member:](#_my1a0y5ru0xw) 15

[CompSwimmer:](#_b8cohqilp19w) 15

[Rules:](#_khbtqu4yk86e) 15

[Files:](#_9d1aiwkasuo7) 15

[**ITO**](#_dv1idrbbp3ax) **16**

[SWOT Analysis:](#_qycef71957nc) 16

[Interessentanalyse:](#_t096lle7cowe) 18

[Risikoanalyse:](#_oiqg5h7otr34) 18

# **Resume:**

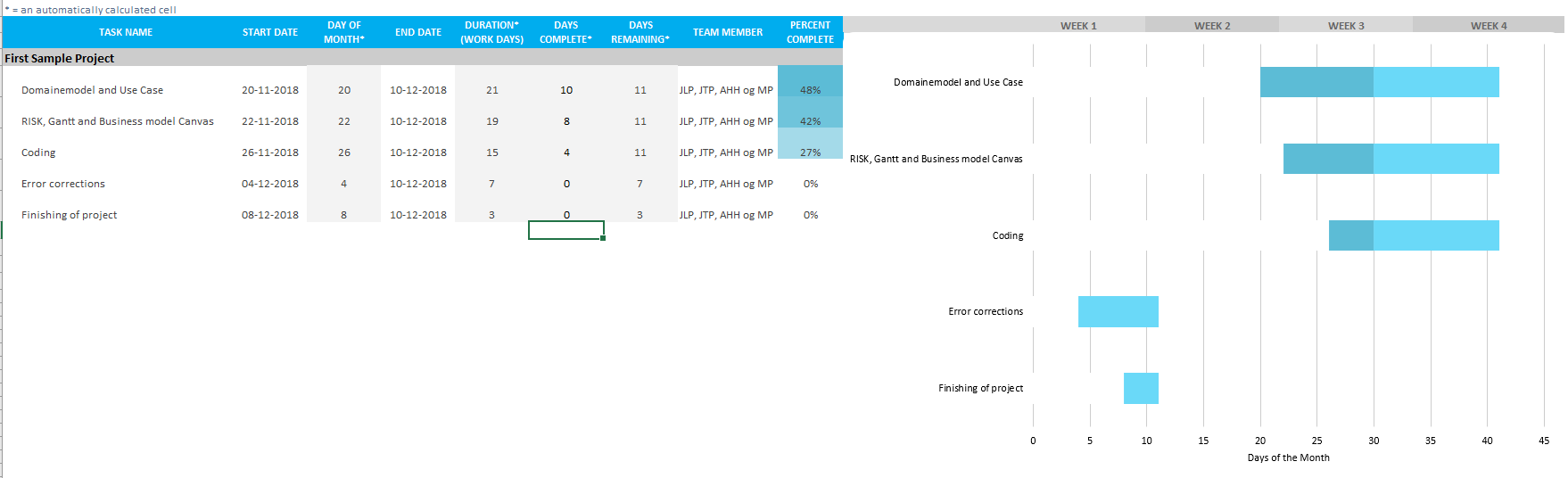
Målet med denne opgave er at lave et program til svømmeklubben Delfinen, der kan holde styr på klubbens medlemmer. Formålet med denne rapport er at give den nødvendige dokumentation for at forstå tankeprocessen bag softwarens design og implementation.

# **Vision:**

Lav et program der skal kunne give klubbens administration mulighed for at oprette medlemmer, slette medlemmer, ændre medlemskaber opkræve kontingent samt sammenligne trænings- og konkurrenceresultater.

Da der er en tidsbegrænsning på omkring 2 uger til at lave dette program, er det formentligt ikke muligt at lave et fuldt færdigt program, det forventes at der udvælges centrale usecases og designes og programmere disse

# Gantt:



# **Kravspecifikation:**

#### 1.0 Funktionelle krav:

|  |  |
| --- | --- |
| 1.1 | Programmet skal kunne håndtere informationer omkring medlemmer. (navn, alder etc.) |
| 1.2 | Medlemmer skal kunne opdeles som værende passive eller aktive. |
| 1.3 | Medlemmer skal kunne opdeles som værende motionist eller konkurrencesvømmer. |
| 1.4 | Medlemmer skal kunne opdeles som værende junior, eller senior. |
| 1.5 | Klubbens kasserer skal kunne tage sig af kontingentbetaling. |
| 1.6 | Kontingentet for juniorsvømmere (under 18 år) skal være 1000 kr. årligt |
| 1.7 | Kontingentet for seniorsvømmere (over 18 år) skal være 1600 kr. årligt. |
| 1.8 | Seniorsvømmere over 60 år skal have 25% rabat på seniorkontingentet. |
| 1.9 | Kontingentet for et passivt medlemskab skal være 500 kr. årligt. |
| 1.10 | Kassereren skal kunne se en oversigt over medlemmer der er i restance. |
| 1.11 | Konkurrencesvømmere skal inddeles i to hold, junior (under 18) og senior (over 18). |
| 1.12 | Hver konkurrencesvømmer skal registreres i forhold til svømmedisciplin. |
| 1.13 | Bedste træningsresultat skal løbende registreres inden for hver svømmedisciplin. |
| 1.14 | Bedste konkurrence resultat skal løbende registreres inden for hver svømmedisciplin |
| 1.15 | Træneren skal kunne se en top 5 over klubbens bedste svømmere inden for hver disciplin. |

#### 2.0 Anvendelighedskrav:

|  |  |
| --- | --- |
| 2.1 | Programmet skal kunne benyttes af klubbens administration og andre superbrugere. |

#### 3.0 Supportability krav:

|  |  |
| --- | --- |
| 3.1 | Programmet skal kunne benyttes på svømmeklubbens computerer. |

#### 4.0 Yderligere ikke-funktionelle krav:

|  |  |
| --- | --- |
| 4.1 | Programmet skal være skrevet i programmeringssproget Java. |
| 4.2 | Programmet skal benytte filer og objekter. |
| 4.3 | Programmet skal anvende nedarvning og arrylister. |

# Design

## Noun Verb analyse:

Svømmeklubben Delfinen er en mindre klub, der er i vækst.

Klubbens ledelse ønsker derfor at få udviklet et administrativt system til at styre medlemsoplysninger, kontingenter og svømmeresultater. Det er klubbens formand, der tager sig af nye medlemmer. Ved indmeldelse i klubben registreres diverse stamoplysninger om personen herunder alder. Desuden registreres oplysninger om personens ønskede aktivitetsform, det vil sige aktivt eller passivt medlemskab, junior eller senior svømmer, motionist eller konkurrencesvømmer. Klubbens kasserer tager sig af alt vedrørende kontingentbetaling. Kontingentets størrelse er betinget af flere forhold. For aktive medlemmer er kontingentet for ungdoms svømmere (under 18 år) 1000 kr årligt, for seniorsvømmere (18 år og over) 1600 kr. årligt. For medlemmer over 60 år gives der 25 % rabat af senior taksten. For passivt medlemskab er taksten 500 kr. årligt. Kassereren har ønsket, at systemet kan vise en oversigt over medlemmer der er i restance.

Konkurrencesvømmerne har tilknyttet en træner.

Konkurrencesvømmerne er inddelt i 2 hold efter alder.

Ungdomsholdet er for svømmere under 18 år. Seniorholdet er for svømmere på 18 og over. Hver konkurrencesvømmer er desuden registreret i forhold til hvilke svømmediscipliner, han er aktiv i. Inden for hver svømmedisciplin registreres den enkelte svømmers bedste træningsresultat og dato løbende. For de svømmere, der har deltaget i konkurrencer, registreres stævne, placering og tid. Det er på baggrund af de enkelte svømmers resultater, at træneren udtager svømmere til deltagelse i konkurrencer. Træneren ønsker derfor en oversigt, der kan vise klubbens top 5 svømmere inden for hver svømmedisciplin.

**Navneord**.

**Udsagnsord.**

**ledelse system medlemsoplysninger**

**svømmeresultater medlemmer stamoplysninger kasserer**

**kontingentbetaling ungdomssvømmere seniorsvømmere seniortaksten**

**taksten oversigt Konkurrencesvømmerne træner**

**Ungdomsholdet Seniorholdet svømmediscipliner kontingenter**

**træningsresultat dato oversigt**

**styre indmeldelse registreres vise tilknyttet**

**inddelt registreret deltaget udtager**

## Brief Use Case:

Disse use cases er hvad vi kom frem til ud fra Noun Verb analysen.

**UC #1: Registration**

**Primary Actor: Admin**

**Main Success Scenario:**

1. The management opens the system
2. The management chooses to add a member.
3. The management enters the new information about the member.
4. The management exits the system.

**UC #2: Subscription**

**Primary Actor: Cashier**

**Main Success Scenario:**

1. The cashier opens the system
2. The cashier chooses to edit subscriptions for a user.
3. The cashier enters a user for payment of subscription.
4. The cashier enters the payment.
5. The system accepts the payment and prints a date for next payment.
6. The cashier exits the system.

**UC #3: Arrears overview**

**Primary Actor: Cashier**

**Main Success Scenario:**

1. The cashier opens the system.
2. The cashier chooses to see an overview over members in arrears.
3. The system prints a list over members and how much arrear they’re in.
4. The cashier exits the system.

**UC #4: Register swimmer**

**Primary Actor: Admin**

**Main Success Scenario:**

1. The coach opens the system
2. The coach chooses to add a competition swimmer to a discipline.
3. The coach add/changes the swimmers score.
4. The system asks if the management wants to see a list of top 5 swimmers or exit.
5. The coach chooses to see the list of top 5 swimmers.
6. The coach exits the system.

**UC #5: Swimming Results**

**Primary Actor: Coach**

**Main Success Scenario:**

1. The coach opens the system
2. The coach chooses to see swimming results
3. The system prints a list of users and their results.
4. The coach exits the system.

## Casual Use Case:

**UC #2: Subscription**

**Primary Actor: Cashier**

**Main Success Scenario:**

1. The cashier opens the system
2. The cashier chooses to edit subscriptions for a user.
3. The cashier enters a user for payment of subscription.
4. The cashier enters the payment.
5. The system prints an amount of arrears and accepts the paid amount.
6. The cashier exits the system.

**Alternative Flows: (extensions)**

1. At all times, if system fails.
   1. The system returns to previous page.
   2. The system cannot return to previous page.
      1. The system crashes.
      2. The cashier starts over.
2. If user doesn't exist.
   1. The system shows an error.
   2. The system cannot return to previous page.
      1. The system crashes
      2. The cashier starts over

**UC #4: Register swimmer**

**Primary Actor: Coach**

**Main Success Scenario:**

1. The coach opens the system
2. The coach chooses to add a competition swimmer to a discipline.
3. The coach add/changes the swimmers records.
4. The system asks if the coach wants to see a list of top 5 swimmers or exit.
5. The coach chooses to see the list of top 5 swimmers.
6. The coach exits the system.

**Alternative Flows: (extensions)**

1. At all times, if system fails.
   1. The system returns to previous page.
   2. The system cannot return to previous page.
      1. The system crashes.
      2. The cashier starts over.
2. If swimmer doesn’t exist
   1. The system shows an error
   2. The system cannot return to previous page
      1. The system crashes.
      2. The coach starts over.

## Use Case Diagram:

Vores Use Case Diagram går kort ud på at forklare hvilke actors der har adgang til de forskellige dele af systemet.

##### Chairman:

Som set på Use Case Diagrammet har “Chairman” adgang til “Registration” som udgør den del af systemet hvor personer bliver oprettet til indmeldelse, enten som “Casual” eller “Competitive” medlem.

##### Cashier:

Står for medlemskaber hos de forskellige tilføjet kunder. Derudover er “Cashier” ansvarlig for medlemskabstyper, om de betaler deres kontingent, og i dette program har “Cashier” mulighed for at tilføje kontingent til alle medlemmer, samt har mulighed for at se et overblik over alle tilføjet medlemmer.

##### Coach:

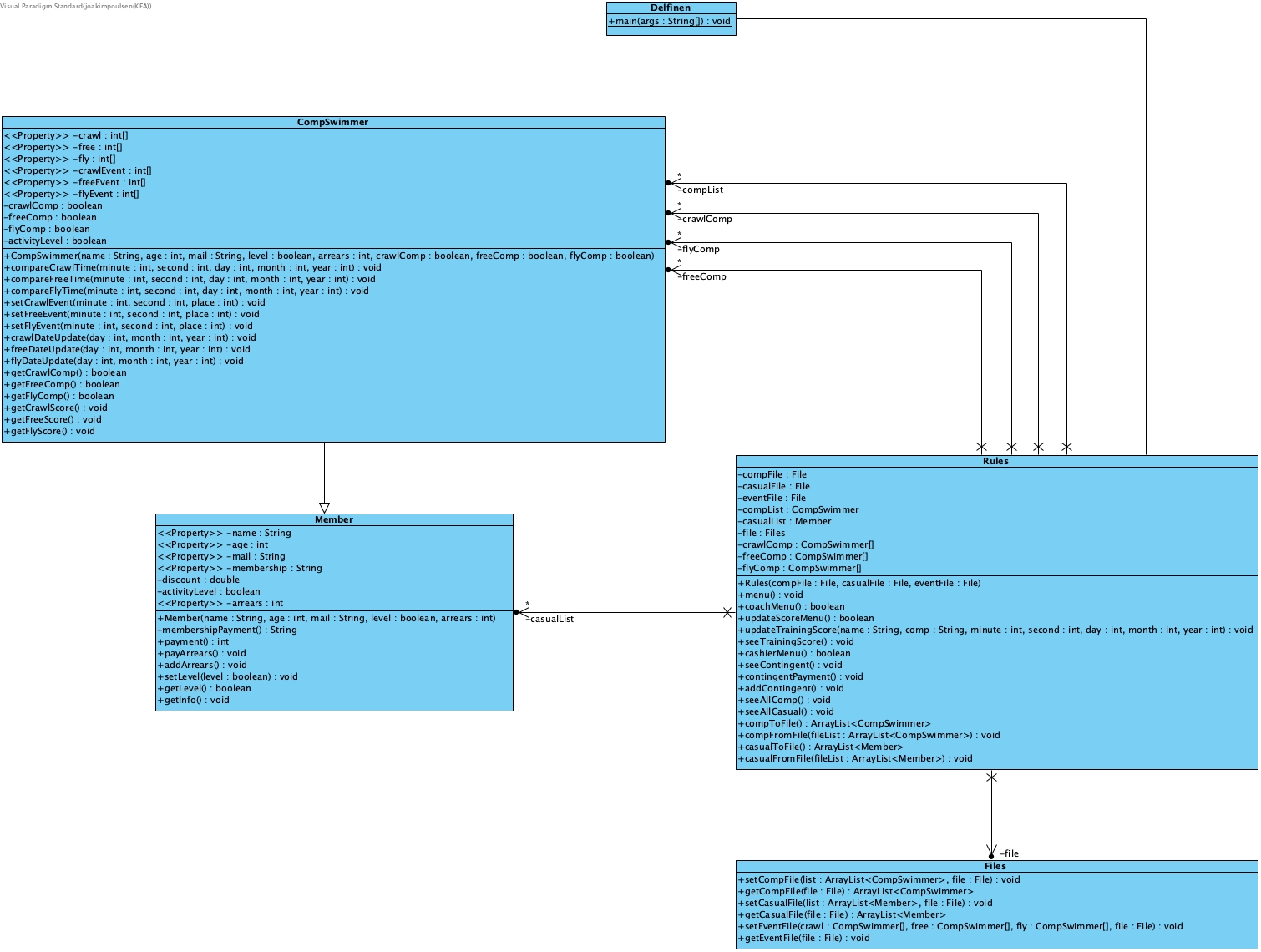
Står for at registrere de enkelte konkurrencesvømmeres trænings- og stævneresultater.

Han har også mulighed for at se en liste over hver enkelt svømmers træningsresultat eller se en hele listen over medlemmer der har været til de skellige stævner.

## Domænemodel:

Efter vi havde udarbejdet vores use cases og vores use case diagram, lavede vi en domænemodel, den skulle være med til at give os et overblik over hvordan vore program ville komme til at se ud. Vores domæne model indeholder en del konceptuelle klasse så som *chairman*, og *subscription*. Disse klasser er ikke med i det færdige program, men deres tilstedeværelse i domænemodellen har været med til at kortlægge hvilke metoder vi har haft brug for, og deres metoder er efterfølgende blevet inkluderet i andre klasser, hvor de var mere passende.

## Klassediagram:

Efter vi havde lavet vore domænemodel gik vi igang med at lave kode til vores program. Til at hjælpe os med at få et overblik over kode udarbejde vi et klassediagram. 

Vi kan ud fra klasse diagrammet se at vore program består af 5 klasser der alle har forskellige ansvarsområder.

## System Sequence Diagram:

Dette viser forholdet til vores 2 casual use cases.

SSD’en til venstre viser sekvensen for subscription, hvor kassereren vil opkræve kontingent fra et medlem. Kassereren indtaster navnet på medlemmet, vælger at ændre på kontingentet fra ikke betalt til betalt, systemet accepterer betalingen og printer en kvittering på at medlemmet nu skylder 0 kr. I kontingent.

SSD’en til højre viser sekvensen for register swimmer, hvor træneren vil registrerer en svømmer til en disciplin. Træneren indtaster navnet på svømmeren, hvorefter han indtaster disciplinen svømmeren skal deltage i. Til sidste afslutter han programmet.

# Konstruktion

## Kode Dokumentation:

##### Delfinen:

Klassen Delfinen er vores main klasse. Det er den klasse vi køre programmet fra, og dens ansvar er oprette et objekt af CompList.txt og CasualList.txt, der indeholder informationer om klubbens motionister og konkurrencesvømmere, samt EventFile.txt der gemmer resultater fra tidligere stævner.

Den opretter også et objekt af klassen Rules, for så at kalde rule.menu() metoden som er den der eksekvere hele programmet.

##### Member:

Member klassen har ansvaret for at holde styr på informationerne omkring klubbens motionister, såsom navn, alder, medlemskabstype etc.

##### CompSwimmer:

CompSwimmer klassen har ansvaret for at holde styr på informationerne omkring klubbens konkurrencesvømmere, hvilket den gør ved at nedarve standard oplysninger såsom, navn, alder og medlemskabstype fra Member, hvorefter den selv udvider med at opdatere træningstid og konkurrencetid.

##### Rules:

Rules klassen har ansvaret for at alle krav bliver overholdt. Det er den klasse der opretter alle programmets menuer, og sørger for at for at informationerne bliver lageret de rigtige steder. Ydermere er det også den klasse der opretter arraylister for hhv. motionister og konkurrencesvømmere som bliver brugt af vores Files klasse.

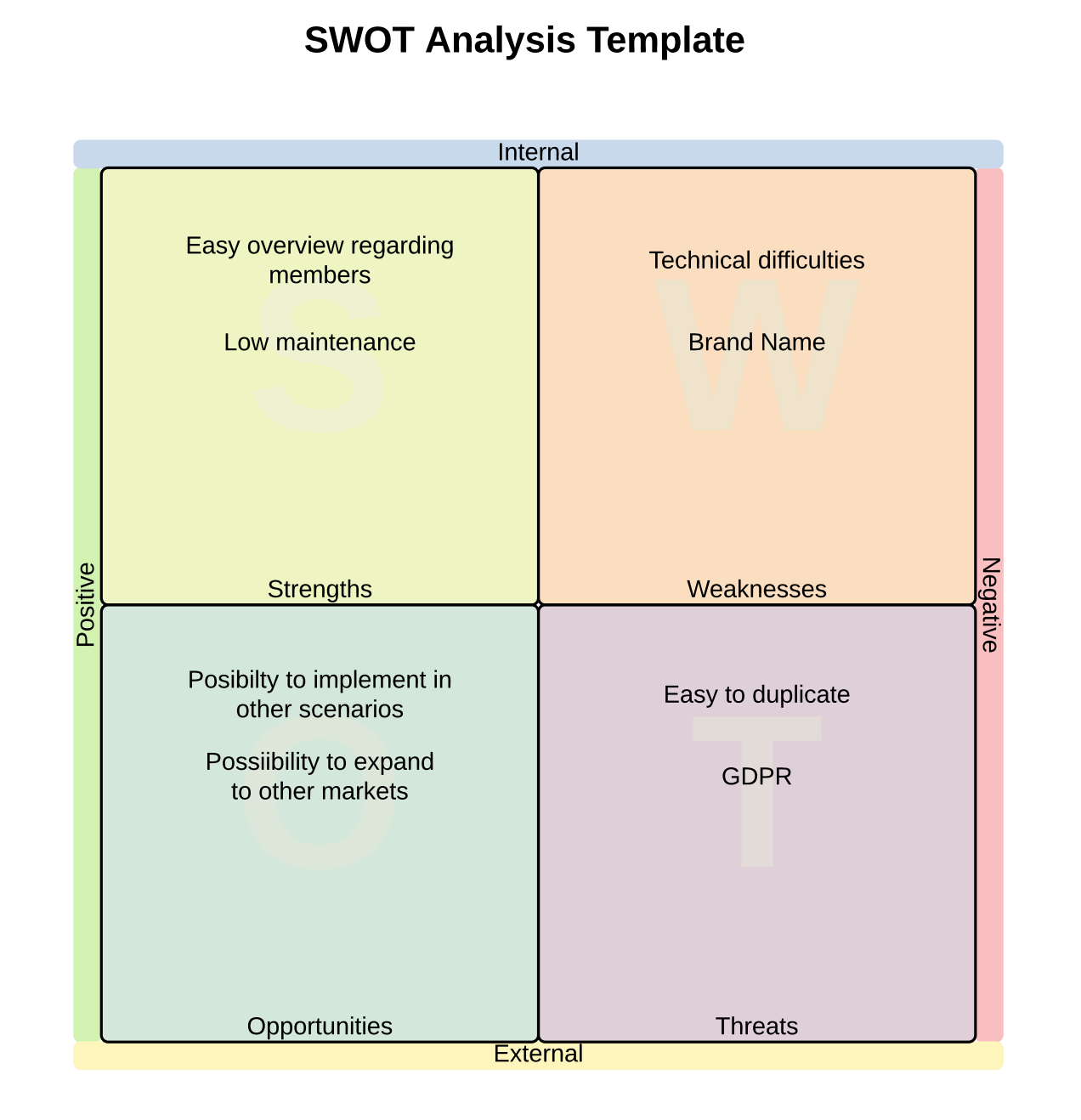
##### Files:

Files klassen har ansvaret for at gemme de informationer den får om klubbens motionister og konkurrencesvømmere i 2 separate filer.

Den gemmer også informationerne om hvordan klubbens konkurrencesvømmer klarer det til de forskellige stævner.

# ITO

## SWOT Analysis:



Vi har i vores SWOT analyse valgt hvilke styrker og svagheder man står overfor som udviklere til et program som dette eller lignende. SWOT analysen er delt op i 4 dele, “Strengths”, “Weaknesses”, “Opportunities” samt “Threats”.

I en Swot analyse vil man have interne og eksterne faktorer

Styrker er en intern faktor, det vil altså sige her kigger man indad, i denne kasse har vi placeret “Easy overview regarding member” fordi man som ansat i denne svømmeklub vil have meget nemmere adgang til svømmeklubbens medlemsdatabase gennem det system vi har programmeret i forhold til, hvis man for eksempel skulle gennemgå et gammeldags kartotek.

Den anden styrke, som også er intern er low maintenance, det vil altså sige det ikke kræver store mængder ressourcer at opretholde vores system, da det hele er lagret i databaser. Her kan man igen sammenligne det med et kartotek system, der kræver en del flere ressourcer, blandt andet skal alle informationer printes ud.

Næste interne faktor er “Weaknesses”, her har vi valgt technical difficulties, for selv om det kræver mindre i generel vedligeholdelse, betyder det også man kan få problemer, der kunne undgås hvis man havde det som et fysisk kartotek. Et problem kunne f.eks. Være strømsvigt eller IT-nedbrud.

Vores anden weakness vil være Brand name. Dette bunder i at man som udvikler til sådan et program vil have vanskeligheder med at “brande” sig selv, da vores opgave ikke er specielt specifik og vi ikke har konkret specialiseret os i noget. Vi udvikler trods alt bare et program til at håndtere data.

Næste firkant indeholder Opportunities og er en ekstern faktor, her er der altså tale om muligheder, lidt i forlængelse af vores svaghed angående Brand recognition, kan det faktisk også komme os til gavn. Da, det er et meget generelt system vi udvikler, vil man også have mulighed for nemt at kunne skræddersy det til andre virksomheder. For eksempel en fodboldklub.

Desuden behøver vi ikke kun holde det inde for klubadministration, det kunne også sagtens benyttes hos f.eks. En fabrik, der ønsker at benytte systemet til at administrere medarbejdernes arbejdstider.

Sidste firkant indeholder, “Threats” og er den sidste eksterne faktor. Her har vi vurderet at vores er simpelt nok til nemt at kunne kopieres og implementeres af andre virksomheder, det indeholder altså hverken specielt meget innovation eller specialisering.

Sidste trussel vil være GDPR. Altså den nye persondatalovgivning. Her vil det være vigtigt for os at have helt styr på reglerne i forhold til denne, da vores system kan indeholde personlige data, som måske skal behandles på en specifik måde.

## 

## Interessentanalyse:

I en stakeholder analyse vælger man hvilke personer i projektet, der kan have indflydelse og ud fra hvor de placeres i diagrammet kan man lægge strategi i forhold til hvordan man skal håndtere dem.

I venstre hjørne i bunden har man personer med lidt indflydelsen og lidt interesse, disse er ikke særlig relevante.   
Højre hjørne i bunden, har man personer med mere interesse, dog ikke med så meget indflydelse. Disse skal man generelt bare holde informeret.

Venstre hjørne i toppen har man personer med meget indflydelse men ikke med den store interesse. Disse skal man bare tilfredsstille.

Til sidst har vi personer i højre hjørne i toppen, disse personer har meget indflydelse og er meget interesserede i produktet. Disse skal man virkelig være obs på.

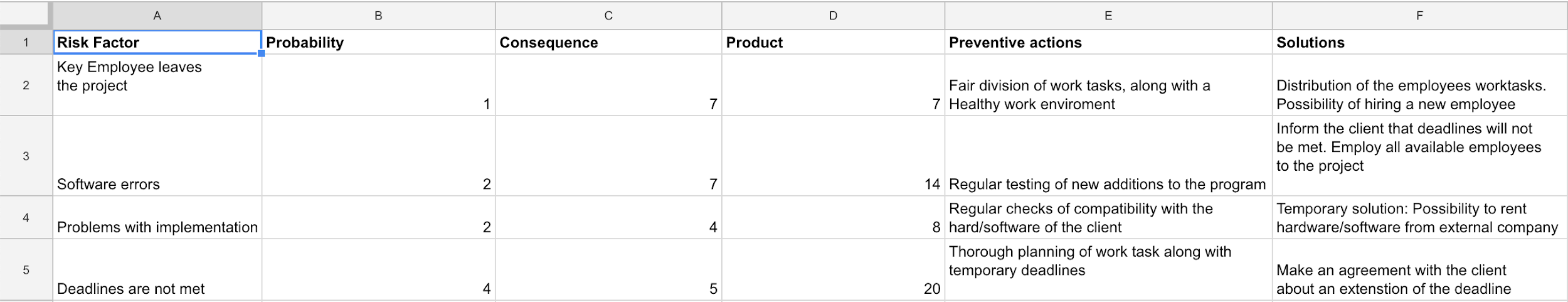
Vi har placeret Livreddere, generelle ansatte og juniorholdet, i højre hjørne nederst. Det har vi gjort fordi ingen af disse vil have den store interesse i programmet, samt de heller ikke har nogen særlig indflydelse.

I venstre hjørne nederst har vi placeret seniorholdet, dette har vi gjort fordi de vil sandsynligvis være lidt mere interesseret i hvordan programmet vil kunne komme til fungere, dog ville de nok ikke have den store indflydelse alligevel.

I toppen af højre hjørne har vi ejeren af svømmehallen, han har stor indflydelse over svømmeklubben, men han vil sandsynligvis ikke være meget engageret i systemet, da han ikke rigtigt vil benytte det.

Til sidst har vi i toppen af venstre hjørne træneren og kassereren, disse to personer er de vigtigste i projektet, da de har en del mere indflydelse end for eksempel livredderen og de har en meget stor interesse da de vil være primære brugere af programmet.

## Risikoanalyse:



I risikoanalysen har vi valgt at tage udgangspunkt i 4 risikofaktorer som kan blive aktuelle og derfor skal der lægges en præventiv plan samt en løsning på problemet.

Første risikofaktor er hvis en nøgleperson forlader projektet, her har vi vurderet at risikoen ikke er særlig høj, men det er konsekvenserne, derfor har vi vurderet det til henholdsvis 1 og 7 på skalaen. Vores præventive tiltag inkluderer en fair arbejdsfordeling og et godt arbejdsmiljø der skal sørge for at personer ikke ønsker at forlade projektet. Skulle det dog blive aktuelt har vi bestemt at opgaverne skal uddelegeres og der skal sættes ind på at finde en ny medarbejder.

Anden risikofaktor omhandler Softwarefejl, her har vi vurderet at risikoen er en smule højere mens konsekvensen forbliver det samme som i risikofaktor 1. Vores præventive tiltag vil være løbende kontrol af nye tilføjelser til programmet, for hele tiden være forsikret om at det virker.

Skulle vi løbe ind i problemer alligevel, vil vores løsning være at gøre klienten opmærksom på at deadlinen ikke kunne holdes og vi vil indsætte al ledig arbejdskraft på programmet.

Tredje risikofaktor indebærer inplementaionsfejl, her taler vi om hardware og software. Denne har vi vurderet som en forholdsvis lav sandsynlighed og med en knap så stor konsekvens som i de andre. Vores præventive tiltag, er som i faktor 2 løbende kontrol om systemet kan implementeres. Skulle det hænde alligevel har vi en midlertidig løsning, at vi vil låne bla. Hardware som vi ved systemet vil kunne køre på.

Fjerde risikofaktor indebærer at vi ikke holder deadlinen, denne har en høj sandsynlighed og en forholdsvis stor konsekvens. Det præventive tiltag vil være at planlægge vores arbejdsindsats korrekt samt sætte deadlines til os selv i løbet af processen. Skulle det ske alligevel vil vi i fællesskab med klienten aftale at den endelige deadline kunne rykkes.