POMODORO TIMER

Af Zulay Bashueva

Programmering B

3.x, EUC Syd Haderslev

24. april 2025

Indhold

1 Projektbeskrivelse

Hvordan kan vi ved hjælp af de metoder og værktøj fra programmering lave et Pomodoro program og hvordan kan en Pomodoro-app forbedre produktiviteten og koncentrationen hos brugere, og hvilke faktorer spiller en rolle i dens effektivitet?

For at besvare denne problemstilling vil projektet undersøge:

- Hvad Pomodoro-teknikken er, og hvordan den fungerer i en digital kontekst.
- Hvordan en Pomodoro-app påvirker brugernes arbejdsvaner og produktivitet.
- Hvilke begrænsninger og potentielle forbedringer der kan implementeres for at optimere en Pomodoro-apps effektivitet.

2 Indledning

For at være en god elev skal man vide, hvordan man lærer. Dette kræver både koncentration og tid, men hvad gør man, når koncentrationen svigter, eller man har svært ved at holde fokus i længere perioder?

Pomodoro-teknikken er en studiemetode, der opdeler arbejdstiden i intervaller. Typisk er intervallerne struktureret på denne måde hvor man begynder med 25 minutters fokuseret arbejde efterfulgt af en 5-minutters pause, med en længere pause efter fire arbejdssessioner. Ikke alle har en Pomodoro-timer til rådighed, og mange ønsker ikke at bruge penge på et fysisk objekt. Derfor kan en gratis app være en enkel løsning på dette problem.

Dette projekt vil fokusere på følgende problemstilling:

Hvordan kan vi ved hjælp af de metoder og værktøj fra programmering lave et Pomodoro program og hvordan kan en Pomodoro-app forbedre produktiviteten og koncentrationen hos brugere, og hvilke faktorer spiller en rolle i dens effektivitet?

For at besvare denne problemstilling vil projektet undersøge:

- Hvad Pomodoro-teknikken er, og hvordan den fungerer i en digital kontekst.
- Hvordan en Pomodoro-app påvirker brugernes arbejdsvaner og produktivitet.
- Hvilke begrænsninger og potentielle forbedringer der kan implementeres for at optimere en Pomodoro-apps effektivitet.

Rapporten indleder med en teori og metode afsnit hvorefter der udarbejdes en kravspecifikation. Efter det følger analyseafsnittet og diskussionen. Rapporten konkluderes med en konklusion.

3 Teori og metode

Pomodoro-teknikken er en tidsstyringsmetode udviklet af Francesco Cirillo i slutningen af 1980'erne. Teknikken går ud på at opdele arbejdstiden i fokuserede intervaller også kaldt pomdoro'er, typisk på 25 minutter, efterfulgt af en kort pause på 5 minutter. Efter fire sådanne arbejdsintervaller tages en længere pause på 15-30 minutter. (https://altomledelse.dk/pomodoroteknikken/) Metoden er opkaldt efter det italienske ord for "tomat", da Cirillo oprindeligt brugte et køkken ur formet som en tomat til at holde styr på tiden. (https://en.wikipedia.org/wiki/Pomodoro_Technique)

En af de største fordele ved Pomodoro-teknikken er dens evne til at øge produktiviteten og koncentrationen. De faste tidsintervaller gør det lettere at undgå overspringshandlinger, da man arbejder intenst i en kort periode og derefter belønnes med en pause. Derudover kan teknikken hjælpe med at reducere mental udmattelse, fordi regelmæssige pauser sikrer, at hjernen får tid til at restituere. Den strukturerede tilgang kan også gøre store opgaver mere overskuelige ved at bryde dem ned i mindre dele.

På trods af dens fordele har Pomodoro-teknikken også nogle ulemper. En af de mest nævneværdige er, at den ikke altid passer til arbejdsopgaver, der kræver længere perioder med fordybelse. For eksempel kan en kompleks opgave kræve mere end 25 minutters uafbrudt arbejde, og en pause kan i sådanne tilfælde virke forstyrrende. Desuden kan teknikken være svær at implementere i arbejdsmiljøer, hvor uforudsete afbrydelser er almindelige, da det kan være vanskeligt at holde fast i de stramme tidsintervaller.

Samlet set kan Pomodoro-teknikken være en effektiv metode til at forbedre fokus og arbejdsflow, men den er ikke nødvendigvis den bedste løsning for alle typer af opgaver og arbejdssituationer.

I dette projekt anvendes pseudokode, rutediagrammer og klassediagrammer for at gøre udviklingen mere struktureret og effektiv. Programmet er objektorienteret og udviklet i Processing's Java for Android, hvilket sikrer en interaktiv og brugervenlig mobilapplikation.

Appen fungerer som en Pomodoro-timer, der automatisk skifter mellem 25 minutters arbejde og 5 minutters pause, med en længere 30-minutters pause efter fire cyklusser. En rød cirkel fungerer som visuel timer, og en start/stopknap giver brugeren kontrol over nedtællingen.

4 Kravspecifikation

Appen skal opfylde en række krav, og på baggrund af teori- og metodeafsnittet opstilles følgende kravspecifikation.

Pomodoro-appen skal kunne udføre en række centrale funktioner.

Hårde krav omfatter:

- En timerfunktion, der automatisk sætter 25 minutters fokuseret arbejdstid.
- Registrering af, hvor mange pomodoro-intervaller der er gennemført.
- Automatisk indsættelse af en længere pause efter fire gennemførte pomodoro-intervaller.

Bløde krav omfatter:

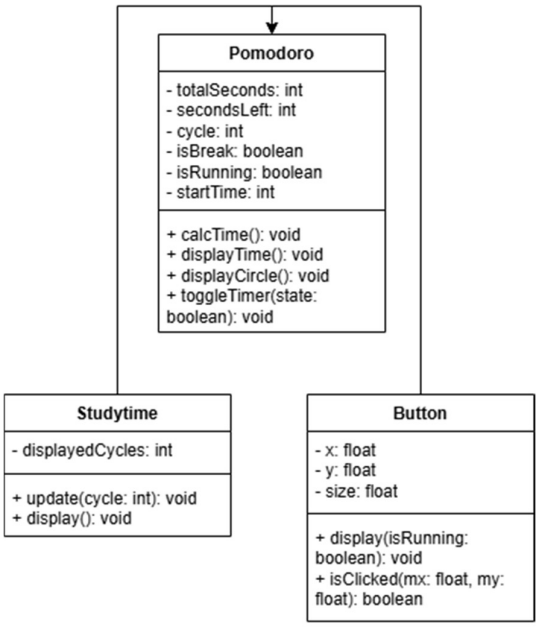
- En brugervenlig brugerflade, som er nem at navigere i.
- Et visuelt design, der appellerer til målgruppen og understøtter motivationen til at bruge appen regelmæssigt.

Programmet vil blive brugertestet hvori vi kan se om appen opfylder kravene.

5 Analyseafsnit

Udviklingen af programmet begyndte med en grundig planlægningsfase. GitHub Projects blev brugt til at organisere arbejdet og skabe overblik over opgaver og deadlines. Ved hjælp af tavler og opgavestatus kunne hele processen opdeles i trin som design, kodning og test. Denne form for planlægning gjorde det lettere at holde fokus og følge en tydelig arbejdsproces.

Til det visuelle overblik blev Draw.io anvendt. Her blev der udarbejdet både et rutediagram og et klassediagram. Rutediagrammet viser hvordan programmet reagerer på brugerens input og hvordan det skifter mellem arbejds- og pausefaser. Klassediagrammet viser programmets struktur med klasserne Pomodoro, Button og StudyTime samt deres funktioner.
Pomodoro, som styrer tiden og cyklusserne, med variabler som totalSeconds, secondsLeft, cycle, isBreak, isRunning og startTime samt metoder som calcTime(), displayTime(), displayCircle() og toggleTimer(state).
StudyTime, som håndterer visningen af gennemførte arbejdsperioder med displayedCycles og funktionerne update(cycle) og display().
Og Button, der gør det muligt at starte og stoppe timeren, med variablerne x, y og size samt funktionerne display(isRunning) og isClicked(mx, my).



I et udvidet diagram ses desuden mulighederne for nedarvning, hvor fælles funktioner kan samles i overordnede klasser som Timer. Men i dette program sker der en kommunikation gennem klasserne da isRunning er en funktion som viser om timeren er i gang og cyklusser.

Der blev også brugt Figma til at lave en plan for appens udseende.

To forskellige farvepaletter og designforslag blev testet af på brugere, hvoraf 3 ud af 5 foretrak palette 2 frem for palette 1.

Programmet starter med at oprette de nødvendige objekter. I setup()-metoden initialiseres timeren, knappen og lyden. Variablen isRunning bruges til at holde styr på om timeren kører eller ej. Når programmet er startet, gentages draw()-metoden løbende. Her bliver

skærmen opdateret, tiden beregnes, grafikken vises og knappen tegnes. Det er i denne løbende opdatering programmets kernealgoritme udføres.

Logikken bag Pomodoro-systemet er bygget op omkring en sekvens af arbejdsperioder og pauser. Efter fire arbejdsperioder gives en længere pause. Dette styres af koden i klassen Pomodoro og beskrives bedst med følgende pseudokode:

```
START PROGRAM

INITIALISER:
  - Opret Pomodoro-objekt
  - Opret knap
  - Indlæs lyd
  - isRunning = FALSE

LOOP:
  - Ryd skærmen
  - Opdater Pomodoro hvis isRunning er sand
  - Vis timer som tekst og cirkel
  - Vis knap

VED KLIK PÅ KNAP:
  - Skift isRunning fra sand til falsk eller omvendt
  - Opdater timerens starttid

KLASSE: Pomodoro
  INIT:
    - totalSeconds = 25 minutter
    - isBreak = FALSE
    - cycles = 0
    - startTime = tidspunkt programmet starter

  update():
    - elapsedTime = tid der er gået
    - secondsLeft = totalSeconds - elapsedTime
    - hvis tiden er udløbet:
        - hvis det ikke er pause:
            - cycles++
        - skift mellem pause og arbejde
        - hvis det er pause:
            - hvis fire cycles er gået, lav lang pause
            - ellers lav kort pause
        - ellers ny arbejdsperiode
        - start tiden igen

  toggleTimer(state):
    - isRunning = state
    - hvis state er sand:
        - startTime = nuværende tid minus den tid der er gået

```

Pseudokoden viser det logiske forløb, som derefter implementeres direkte i Java/Processing. Herunder ses et udsnit af den faktiske kode i calcTime():

```
void calcTime() {
  if (isRunning) {
    int elapsedTime = (millis() - startTime) / 1000;
    secondsLeft = totalSeconds - elapsedTime;

    if (secondsLeft <= 0) {
      if (!isBreak) {
        cycle++;
        studyTime.update(cycle);
      }
```

```
11.       isBreak = !isBreak;
12.       totalSeconds = (isBreak) ? ((cycle % 4 == 0) ? 30 * 60 : 5 * 60) : 25 * 60;
13.       secondsLeft = totalSeconds;
14.       startTime = millis();
15.     }
16.   }
17. }
18.
```

Her foretages den centrale beregning af tiden og skiftet mellem arbejds- og pausefaser. Hvis tiden er gået, kontrolleres det om det er tid til en pause eller en ny arbejdsperiode. Samtidig opdateres antallet af gennemførte Pomodoro-intervaller.

Brugerens interaktion sker via skærmtryk. Når der trykkes på knappen, starter eller stopper timeren, og der aktiveres eller deaktiveres lyd. Det styres af følgende metode:

```
1. void mousePressed() {
2.   if (startStopButton.isClicked(mouseX, mouseY)) {
3.     isRunning = !isRunning;
4.     p.toggleTimer(isRunning);
5.   }
6.   if (isRunning) {
7.     noise.loop();
8.   } else {
9.     noise.stop();
10.  }
11. }
12.
```

Denne funktion sikrer en enkel betjening, hvor én knap styrer hele systemet. Hvis timeren kører, høres en baggrundslyd som signalerer at man er i gang. Lyden, der spilles under arbejdet, er white noise, som forskning har vist kan hjælpe med fokus og koncentration både under arbejde og til søvn. (https://www.bps.org.uk/research-digest/can-listening-white-noise-help-you-focus) Cassette-biblioteket bliver importeret igennem:

```
1. import cassette.audiofiles.SoundFile;
```

Fordi det understøtter lyd på Android, hvorimod Processing's eget lydbibliotek ikke er optimeret til dette.

Visualiseringen af gennemførte arbejdsperioder vises som cirkler øverst på skærmen. Det styres af klassen StudyTime, som løbende opdaterer og tegner et mønster ud fra hvor mange intervaller der er afsluttet. Denne funktion fungerer som en motiverende visning af brugerens fremskridt.

For at gøre timeren mere underholdende og samtidig sætte en naturlig grænse for hvor meget der studeres, har både teksten med tid og de viste pomodoroer samme farve. Når skærmen bliver fyldt, og tallene ikke længere er synlige, kan det tolkes som et signal om at dagens studie bør afsluttes.

6 Diskussion

Pomodoro-appen har flere gode funktioner, men der er også plads til forbedringer, som kan gøre appen bedre og nemmere at bruge. En vigtig forbedring kunne være at lave en

overklasse til timerfunktionen. Det ville samle al timerlogik ét sted, så koden bliver lettere at vedligeholde. Det kunne også gøre det nemmere at tilføje flere timer-relaterede funktioner i fremtiden.

En anden forbedring kunne være at bruge en HashMap til at gemme brugerens fremgang. Ved at gemme antallet af Pomodoro-sessioner for hver dag, kunne brugeren få et klart billede af deres fremskridt, hvilket kan være motiverende. Det ville også gøre det muligt at vise statistik over langvarige studievaner.

Desuden kunne gamifikation[1]-elementer som præstationsbelønninger, badges eller mål for gennemførte sessioner gøre appen mere engagerende. Det kunne hjælpe med at holde brugeren motiveret og give en følelse af fremgang. Derudover kunne påmindelser og notifikationer om, hvornår det er tid til at starte eller tage en pause, hjælpe brugeren med at holde fokus.

En anden nyttig funktion kunne være at integrere en kalender, så brugeren kan planlægge deres Pomodoro-sessioner sammen med andre aktiviteter. Det ville gøre det lettere at få styr på tiden og sikre, at man holder sig til sin plan.

Samlet set kunne disse forbedringer gøre Pomodoro-appen mere brugervenlig, fleksibel og motiverende, hvilket vil hjælpe brugeren med at blive mere produktiv og holde fokus længere.

Et problem i appen er, at lyden stopper helt, når timeren pauses, og det var ikke muligt at finde en løsning på dette.

7 Konklusion

Pomodoro-teknikken er en effektiv metode til at forbedre koncentration og produktivitet gennem strukturerede arbejdsintervaller og pauser. Dette projekt har undersøgt, hvordan en Pomodoro-app kan implementeres ved hjælp af programmering, og hvordan den kan hjælpe brugere med at øge deres fokus og arbejdsflow. Appen har vist sig at være en god løsning, der understøtter teknikkens principper.

Der er dog plads til forbedringer, såsom at bruge en HashMap til at gemme brugerens fremgang og tilføje funktioner som gamifikation og kalenderintegration. Disse opdateringer vil gøre appen mere motiverende, hvilket vil optimere brugerens produktivitet.

Samlet set viser undersøgelsen, at en Pomodoro-app kan forbedre brugernes produktivitet og koncentration, og de foreslåede forbedringer vil gøre appen endnu mere effektiv og tilpasset den enkelte brugers behov.

[1] Et fænomen hvor man tilsætter spille-elementer til ting eller aktiviteter som ikke er spil.

8 Kildeliste

Buch, Jesper. *Programmering*. 1. udgave. Aarhus: Systime, 2016.

Wikipedia. "Pomodoro Technique." Wikipedia. 24. april 2025. https://en.wikipedia.org/wiki/Pomodoro_Technique.

Alt om Ledelse. "Pomodoroteknikken." Alt om Ledelse. 24. april 2025. https://altomledelse.dk/pomodoroteknikken/.

Processing. "Reference." Processing. 24. april 2025. https://processing.org/reference.

British Psychological Society. "Can Listening to White Noise Help You Focus?" Research Digest. 24. april 2025. https://www.bps.org.uk/research-digest/can-listening-white-noise-help-you-focus.

9 Bilag

9.1 Kildekode

```
import cassette.audiofiles.SoundFile;

Pomodoro p;                  // Objekt af Pomodoro-klassen, styrer timer-logikken
Button startStopButton;     // Knap til at starte og stoppe timeren
StudyTime studyTime;        // Objekt der holder styr på, hvor mange cyklusser der er gået
boolean isRunning = false;  // Boolean der angiver om timeren er aktiv
SoundFile noise;            // Lydfil der afspilles ved aktivitet

void setup() {
  fullScreen();                     // Starter appen i fuld skærm
  textSize(50);                     // Indstiller tekststørrelsen

  p = new Pomodoro();               // Initialiserer Pomodoro-objektet
  startStopButton = new Button(width / 2, height / 2, width * 0.8); // Placering og størrelse for knappen
  studyTime = new StudyTime();     // Initialiserer studie-tids-trackeren
  noise = new SoundFile(this, "WN.mp3"); // Indlæser lydfil
}

void draw() {
  background(246, 244, 210);       // Baggrundsfarve

  p.calcTime();                     // Opdaterer timeren
  p.displayTime();                  // Viser nedtælling som tekst
  p.displayCircle();                // Tegner visuel timer-cirkel

  studyTime.update(p.cycle);      // Opdaterer cyklusserne hvis nødvendigt
  startStopButton.display(isRunning); // Tegner knappen
  studyTime.display();              // Viser hvor mange pomodoro der er gået
}

void mousePressed() {
  if (startStopButton.isClicked(mouseX, mouseY)) {
    isRunning = !isRunning;         // Skifter mellem start og stop
    p.toggleTimer(isRunning);       // Starter eller stopper Pomodoro-timeren
  }
  if (isRunning) {
    noise.loop();                   // Afspiller baggrundslyd under fokus
  } else {
    noise.stop();                   // Stopper lyden når pausen starter
  }
}

```

```
class Button {
  float x, y, size;

  Button(float x, float y, float size) {
    this.x = x;
    this.y = y;
    this.size = size;
  }

  void display(boolean isRunning) {
    fill(164, 74, 63);           // Ydre cirkel farve
    noStroke();
    circle(x, y, size);          // Tegner selve knappen

    fill(212, 224, 155);         // Indre ikon-farve
    noStroke();

    if (!isRunning) {
```

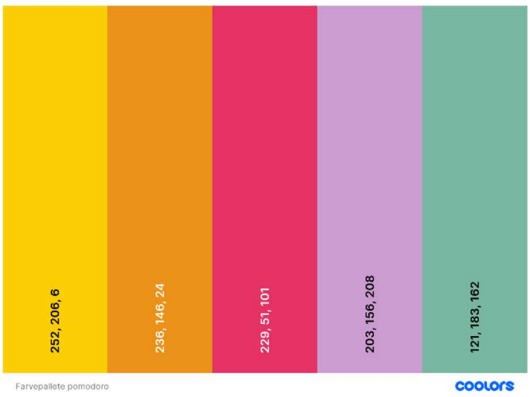
```
19.       // Play-ikon
20.       float tSize = size * 0.4;
21.       triangle(x - tSize / 2, y - tSize / 2, x - tSize / 2, y + tSize / 2, x + tSize / 2, y);
22.     } else {
23.       // Pause-ikon (Ref. til ChatGPT)
24.       float barW = size * 0.15;
25.       float barH = size * 0.5;
26.       rect(x - barW * 1.5, y - barH / 2, barW, barH);
27.       rect(x + barW * 0.5, y - barH / 2, barW, barH);
28.     }
29.   }
30.
31.   boolean isClicked(float mx, float my) {
32.     return dist(mx, my, x, y) < size / 2;  // Returnerer true hvis trykket inden for knappen
33.   }
34. }
35.
```

```
 1. class Pomodoro {
 2.   int totalSeconds = 25 * 60;       // Antal sekunder i én pomodoro
 3.   int secondsLeft = totalSeconds;   // Tid tilbage i nuværende session
 4.   int cycle = 3;                    // Antal gennemførte pomodoros
 5.   boolean isBreak = false;          // Om det er pause eller arbejde
 6.   boolean isRunning = false;        // Om timeren er aktiv
 7.   int startTime = 0;                // Hvornår timeren blev startet
 8.
 9.   void calcTime() {
10.     if (isRunning) {
11.       int elapsedTime = (millis() - startTime) / 1000;     // Tid der er gået i sekunder
12.       secondsLeft = totalSeconds - elapsedTime;            // Hvor meget tid der er tilbage
13.
14.       if (secondsLeft <= 0) {                              // Hvis tiden er gået
15.         if (!isBreak) {
16.           cycle++;                                         // Øger cyklus hvis det var en ar-
bejdsperiode
17.           studyTime.update(cycle);
18.         }
19.         isBreak = !isBreak;                                // Skifter mellem arbejde og pause
20.         if (isBreak) {
21.           totalSeconds = (cycle % 4 == 0) ? 30 * 60 : 5 * 60; // Længere pause efter 4 cy-
klusser
22.         } else {
23.           totalSeconds = 25 * 60;                          // Ellers ny arbejdsperiode
24.         }
25.
26.         secondsLeft = totalSeconds;                        // Nulstiller sekunderne
27.         startTime = millis();                              // Nulstiller starttid
28.       }
29.     }
30.   }
31.
32.   void displayTime() {
33.     int minutes = secondsLeft / 60;
34.     int seconds = secondsLeft % 60;
35.     fill(164, 74, 63);
36.     textAlign(CENTER, CENTER);
37.     text(minutes + ":" + nf(seconds, 2), width / 2, height / 4); // Viser minutter og
sekunder
38.   }
39.
40.   void displayCycle() {
41.     textSize(40);
42.     text("Pomodoro: " + cycle, width / 2, height / 6); // Til intern brug: viser antal
cyklusser
43.   }
44.
45.   void displayCircle() {
46.     fill(164, 74, 63);
47.     noStroke();
```

```
48.     circle(width / 2, height / 2, width * 0.8); // Tegner visuel timer (stor cirkel)
49.   }
50.
51.   void toggleTimer(boolean state) {
52.     isRunning = state;
53.     if (isRunning) {
54.       startTime = millis() - (totalSeconds - secondsLeft) * 1000; // Justerer starttid hvis
pauset
55.     }
56.   }
57. }
58.
```

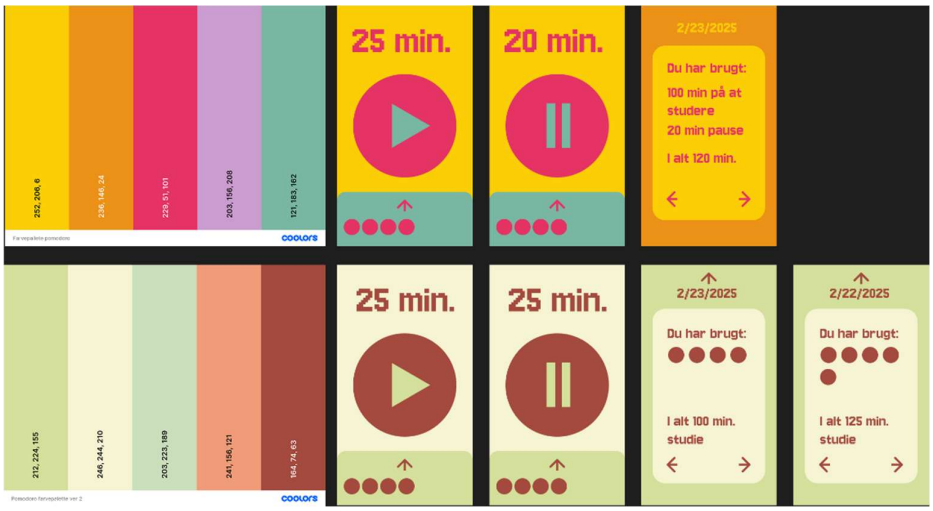
```
 1. class StudyTime {
 2.   int displayedCycles = 0;
 3.
 4.   void update(int pomodoroCycle) {
 5.     displayedCycles = pomodoroCycle;  // Opdaterer antal gennemførte cyklusser
 6.   }
 7.
 8.   void display() {
 9.     int perRow = 4;                          // Hvor mange cirkler per række
10.     float size = width / 10;                // Størrelsen på hver cirkel
11.     float spacing = width / (perRow + 1); // Afstand mellem cirkler
12.     float startY = height - (height - size); // Y-placering
13.
14.     fill(164, 74, 63);
15.     noStroke();
16.
17.     //(Ref. til ChatGPT)
18.     for (int i = 0; i < displayedCycles; i++) {
19.       int row = i / perRow;
20.       int col = i % perRow;
21.       float x = spacing * (col + 1);
22.       float y = startY + row * (size + size/2);
23.
24.       circle(x, y, size);  // Tegner en cirkel for hver gennemført pomodoro
25.     }
26.   }
27. }
28.
```

9.2 Farvepalette ver. 1

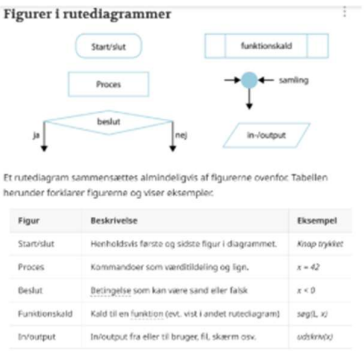


9.3 Farvepalette ver. 2

9.4 Prototyper



9.5 Rutediagram



9.6 Klassediagram