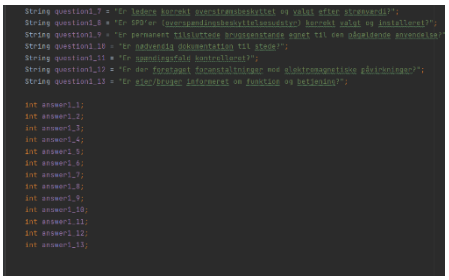
Programmering af variabler (Kristian)

Her ses et problem, nemlig at der er overvældende mange variabler. Hver variabel indeholder et tal mellem -1 og 2 for ja nej irrelevant eller ikke udfyldt.



Jeg forsøger at løse problemet med et array, men for at holde styr på adskillige variabler med flere tal i hver, vil det blive et multidimensional array: int answer1[][]; i første [] skal henvise til nummeret på spørgsmålet, det anden [] skal henvise til svaret. Som det er nu, er det også nødvendigt at lave en metode for hver enkelt tjekboks hvilket vil være et uoverkommeligt stort arbejde. Jeg håber at et multidimentionalt array også vil løse den udfordring ved at gøre det muligt at lave en metode som bliver genbrugt.

Ved nærmere eftertanke vil det nok være bedre med et tredimensionelt array som tager hver gruppe, hvert spørgsmål og hvert svar. int answers[][][];

```
//Array with group, question no, answer. answer is: -1 = not filled, 0 = yes, 1 = no, 2 = NA (not applicable)
int answersGQA[][][] = new int[10][20][4];
```

Det lykkedes at få 715linjers kode ned på 370linjer.

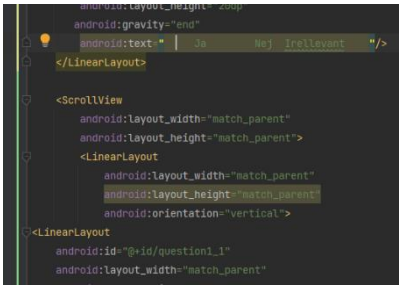
Metoden til at gemme valgene fra tekstboksene, er nu en metode som henter alle svar ved tryk på en gem knap. Men arket er større end skærmen og den kan ikke scrolle endnu. Derfor tilføjes scrolle funktionen i xml.

Scrolle funktionen kan indeholde et Linear layout som så indeholder de andre layouts. På den måde følger det træstrukturen i xml filen:

<ScrollView> -her startes scrollView

<LineaarLayout> -her starts linear layout

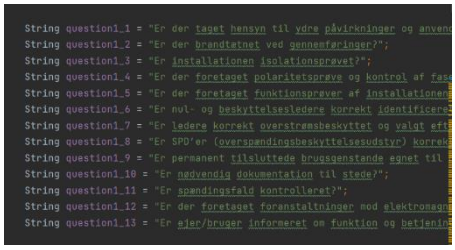
Her I linear layout kommer tekst og checkbokse, noget tekst ligger uden for scrolleviewet, det er for at den hele tiden skal være synlig mens man scroller på skærmen.



</LinearLayout> -linear layout sluttes først fordi den er child af scroll view

</ScrollView> -dernæst sluttes scroll view

For at gøre alle de mange spørgsmål nemmere at arbejde med vil jeg gerne have lagt dem i et todimensionalt array.



Lige nu ligger de i seperate strings. Først opretter jeg et todimensionelt arrayet list til tekst. Arraylist<Arraylist<String> questions = new Arraylist<arraylist<String>>(); fordelen ved array list er at man ikke behøver at definere størrelsen på forhånd.

Arraylisten genbruges til tekstfelterne fra xml og defineres som en liste med View i stedet for tekst. Formålet er senere at kunne autogenerere teksten til tekstfelterne i xml, samt at bruge arrayet til at fejlfinde på om checkboksene er udfyldt korrekt. Jeg vælger også at bruge arraylist fordi der er stor forskel på antallet af spørgsmål i hver gruppe af spørgsmål.

Når der gemmes, er hver enkelt information fra hver tjekboks skrevet ind manuelt, det tog tid og der er risiko for tastefejl. Med det nye todimensionelle array for Textview kan hele denne kode autogenereres. Det er cirka 90 linjer som ændres til to nestede loops. Der er dog det problem at arrayet answer fra verificationChart som den skal gemmes i er foruddefineret i størrelse og det vil helt sikker give problemer før eller side når loopene vil gennemløbe forskellige tomme elementer. Derfor må jeg ændre det til en todimensionel arrayList, det er egentligt et tredimensionelt array, men jeg synes det er på tide at forenkle lidt. Derfor vil spørgsmålene nu være fortløbende uanset hvilken gruppe i tjeklisten de tilhører.

Nu er det tid til at forenkle koden som sætter teksten til xml tekst felterne. Lige nu er der 13 linjer, men der kommer flere. Det skal gerne kunne forenkles med en for løkke.

```
        textFields.add( index: 12,(TextView)findViewById(R.id.text13));

    // using the xml id, here the text from the class VerificationChart is set
        textFields.get(0).setText(vc.questions.get(0));
        textFields.get(1).setText(vc.questions.get(1));
        textFields.get(2).setText(vc.questions.get(2));
        textFields.get(3).setText(vc.questions.get(3));
        textFields.get(4).setText(vc.questions.get(4));
        textFields.get(5).setText(vc.questions.get(5));
        textFields.get(6).setText(vc.questions.get(6));
        textFields.get(7).setText(vc.questions.get(7));
        textFields.get(8).setText(vc.questions.get(8));
        textFields.get(9).setText(vc.questions.get(9));
        textFields.get(10).setText(vc.questions.get(10));
        textFields.get(11).setText(vc.questions.get(11));
        textFields.get(12).setText(vc.questions.get(12));
```

```
// using the xml id, here the text from the class VerificationChart is set

    for (int i = 0; i < textFields.size(); i++) {
        textFields.get(i).setText(vc.questions.get(i));
    }
```

Nu er koden nede på tre linjer. Desuden er den klar til at modtage flere linjer. Det vil den håndtere uden at skulle tilpasses, fordi den tilpasser loopet efter størrelsen på arraylisten.

Når checkfelterne skal gemmes, er det endnu en gang en fordel at få oprettet en liste. Man kan oprette an arrayList, men det vil være svært at holde overblikket når hvert spørgsmål har tre svarmuligheder. En simpel liste er mere overskuelig, fordi vi her har brug for en oversigt. Der er dog også en tredje mulighed og det er at lave en todimensionel arrayliste, så vil vi kunne søge på nummeret for spørgsmålet først og dernæst på svar valget. Det vil sikre at svarmulighederne ikke bliver blandet rundt på de forkerte spørgsmål.

Her bliver boolean listen answer fuldt med svarene fra xml som er gemt i listen check. Der bruges en while løkke til at sikre at der er indhold i listen check, inden der gemmes som boolean true eller false i answer listen fra klassen VerificationChart. Efter at jeg har kodet denne del af programmet, har vi lavet

```
    int i = 0;
    while (chek[i][0] != null) {
        for (int j = 0; j <= 2; j++) {
            vc.answer[0][j] = chek[0][j].isChecked();
        }
        i++;
    }
}
```

et E/R diagram og haft kundemøder. Og de overvejelser som der er gjort på E/R diagrammet samt kundesamtalerne betyder at koden ikke bare skal være mere enkel og autogenereret, men at den skal blive fuldt ud autogenereret, så antallet af spørgsmål kan variere efter behov.

Generering af pdf (Kristian)

For at danne tjekskemaet som pdf benytter vi iText. iText biblioteket importeres i programmet så at vi med enkel kode kan kalde iText filerne og danne pdf'erne.

I AndroidManifest.xml gives der lov til at programmet må skrive til anden lagerplads (External storage). Derefter tilføjes afhængighed (dependencies) i filen build.gradle. blandt andet tilføjes web-adressen så android studio nu kan synkronisere iText biblioteket med versionen itext7-core:7.2.2.

For at gøre det muligt for programmet at gemme en pdf på den enhed den kører på skal den have en tilladelse. Det gøres ved i filen AndroidManifest at tilføje linjen: android:allowBackup="true" , hvorpå den ved første opstart af programmet vil bede om tilladelsen. Når tilladelsen er givet vil der kunne gemmes pdf'er på enheden.