



02410 CDIO-Projekt: Den Endelige Rapport

Gruppe 16
24 June 2021



(a) Fadl Abbas
Matar - s195846



(b) Khaled Zamzam - s195487



(c) Mohamad Abdulfatah Ashmar -
s176492



(d) Bashar Bdewi -
s183356



(e) Anthony
Haidari - s141479



(f) Sid Ali Mounib
- s195484

Contents

1	Resume ark over status af projektet	2
2	Projekt plan	3
3	Risiko Analyse	5
4	Teknisk Afsnit	7
4.1	Analyse af problemet og Design af løsning	7
4.1.1	Analyse af problemet	7
4.1.2	Design af løsning	7
4.2	OpenCV	10
4.2.1	System komponenter	10
4.2.2	Systemet	10
4.2.3	Klassediagram	11
4.3	TensorFlow lite - Android app	12
4.3.1	Om Produkt	12
4.4	Fejlkilder	13
5	Skrivning til ingeniørstuderende, som skal starte på CDIO kurset	14
6	Konklusion	15
7	Bilag A: Arbejdsfordeling	16
7.1	Delopgave	16
7.2	Rapportafsnit	16
7.3	Videoafsnit	17
8	Bilag B: Kilderne	17

1 ResUME ark over status af projektet

Dette er et resumé på en afrapportering af et CDIO-projekt udfærdiget i juni 2021 på kurset 62410. Opgaven i kurset bestod i at udvikle et velfungerende IT-system, bestående af enten mobiltelefon eller computer, der hjælper én med at løse 7-kabalen. It-Systemet skal udvikles således, at det 'ser' hvordan kortene ligger og beslutter hvilke kort der skal flyttes, hvorefter spilleren flytter disse kort med kabalens regler for øje.

Fra projektets spæde start i 13-ugers perioden, tog gruppens medlemmer den endelige beslutning at udvikle en app i Android Studio vha. tredjeparts bibliotekerne, Tensorflow Lite samt YoloV4, til at løse 7-kabalen. Efter opstående komplikationer med detekteringen af spillekort og andre små problematikker med appen i begyndelsen af 3-ugersperioden i juni, gik nogen af gruppens medlemmerne sammen om at undersøge andre muligheder, mens de øvrige medlemmer fortsat arbejdede i Android Studio.

Det kan nu konstateres, at det er lykkedes os udvikle hele to IT-systemer der løser 7-kabalen. Det ene er blevet udviklet i form af en App til en mobiltelefon og det andet er blevet udviklet ved brug af OpenCV og Python til billedegenkendelse og Java til spillets logik. Det skal ligeledes nævnes at vi anvender samme logik til begge systemer. Det sidstnævnte softwaresystem, udviklet i Java med OpenCV, er vores hovedprodukt. Dette løser 7-kabalen ved brug af en computer samt et kamera.

2 Projekt plan

Gantt-diagram: Diagrammet nedenfor illustrerer vores endelige tidsplan over projektet i 3-ugers perioden i et gantt-diagram. Diagrammet kan virke lidt tom, taget betragtning af at projektet har strækket sig ud over 16 uger, men vi har her valgt kun at fokusere på de opgaver der skal laves i 3-uger perioden.

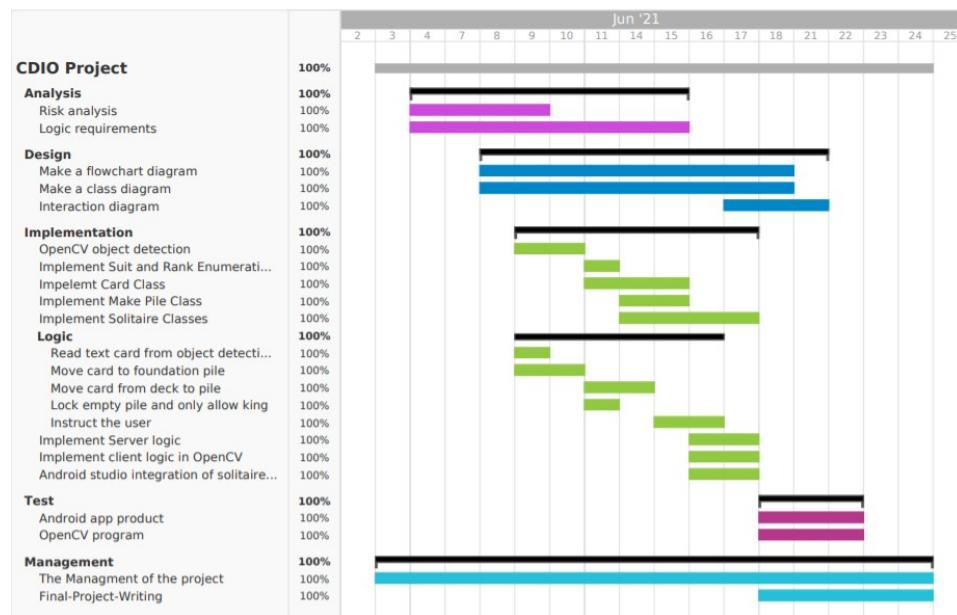


Figure 1: Oversigt over tidsplan

BurnUp chart: Nedenstående figur viser en visuel status over projekts opgaver.

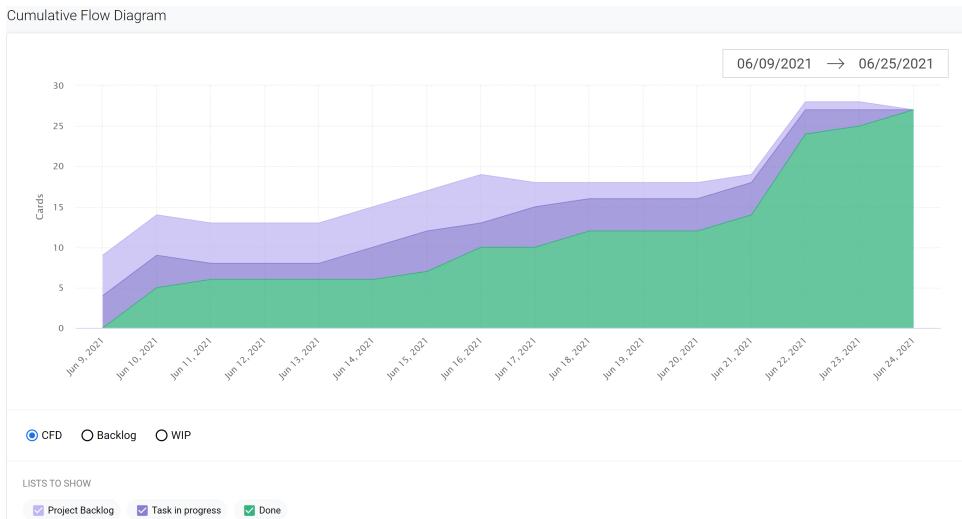


Figure 2: Oversigt over burnUp graph (Lys blå: Project Backlog, Mørke blå: In Progress, Grøn: Færdige)

Trello: Vores projekt var en samling af dele opgaver. Det var faktisk en samling af samlinger af dele opgaver med mulighed for at tilføje assignments, labels, checklists, items, attachments og meget mere. vi kunne også bestemme til hvilken kort dets prioritet, så vi ved godt hvem skulle der gennemføres først. Den stor fordel var at vores dele opgaver kunne ikke nødvendigvis gennemføres kronologisk i rækkefølge eller en ad gangen, medmindre nogle af opgavene var tæt forbundet. For at spore hver opgave brugte vi et rum, hvor vi og de andre gruppemedlemmer kan se dem alle, og hvad deres status er (Assigned , In progress , Completed).

Nedenstående figur viser status over projektets opgaver, herunder hvilke opgaver der er i gang, og hvilke opgave der er afsluttet.

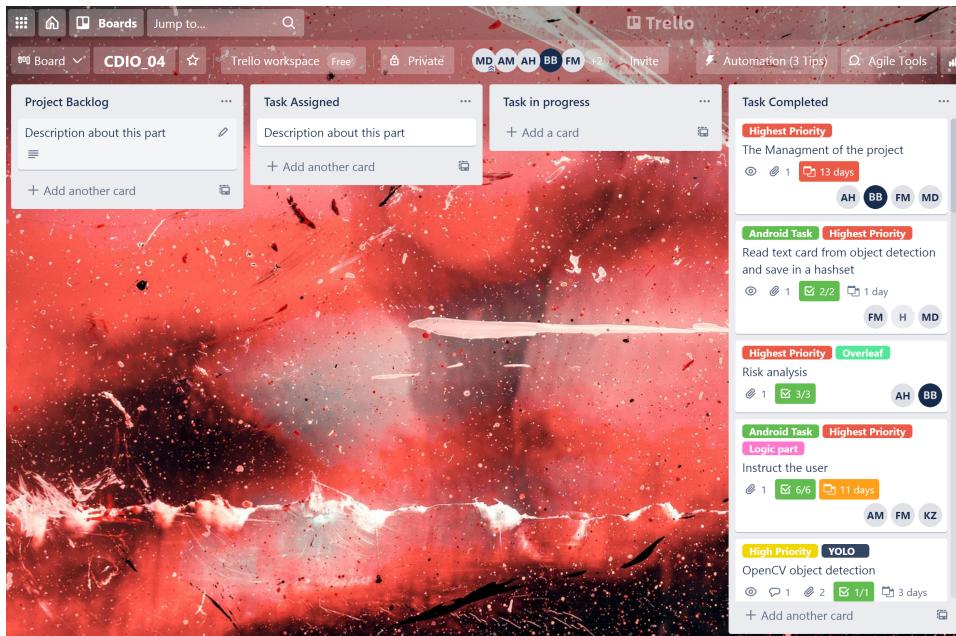


Figure 3: Oversigt over Trello

3 Risiko Analyse

Herunder ses vores opdaterede risici der er blevet fundet frem til. Disse risici er blevet vurderet og udefra vurderingen har ethvert risici fået en score på både dets påvirkning (impact) og deres sandsynlighed for at opstå (probability). De har alle også fået en samlet score som viser, hvor problematiske disse risici er.

Herunder er der blevet dannet strategier for de forskellige risici for at formindske deres effekt og alvorlighed. Yderligere er der blevet sat et ansvar for, hvem der skal holde styr på de forskellige risici og strategier. Sidst er der blevet dannet en ny score efter en vurdering af en løsning til de forskellige risici.

ID	Risiko beskrivelsen (Hvad kan gå galt?)	Arsager (Hvorfor kan det gå galt)	Konsekvens (1-5)	Sandsynlighed (1-5)	Risikofaktor (K x S)	Foranstaltninger (Forebyggende, afbødende)
r1	Guppemedlemmernes manglende deltagelse samt bidrag til projektet.	Sygdom (coronavirus) , manglende motivation eller disciplin. Angst for ikke at kunne præstere.	2	3	6	Opmuntring, anerkendelse, støtte og være behjælpelig.
r2	Gruppemedlemmer dukker ikke op til møder.	Sygdom, forsinkelser, tandlægebesøg osv.	2	2	4	Lav en liste med telefonnumre, så medlemmerne kan kontaktes i tilfælde af forsinkelser eller udblelvelse.
r3	Utilstrækkelig viden eller manglende færdigheder.	Der er endnu ikke opnået fagkundskaber indenfor det pågældende område.	3	4	12	Kritisk opsøg viden på internettet (ask google) og på den måde tilegne de nødvendige færdigheder. Spør om hjælp.
r4	Manglende kommunikation i teamet.	Introverte personligheder, manglende kommunikationsevner. Misinformation.	3	3	9	Afholdelse af møder på daglig basis. Inden mødet forebered da en agenda. Vær detaljeret og vedholdende i kommunikationen omkring projektet. Forventningsafstemme.
r5	Manglende håndtering af konflikter.	I stedet for at løse problemer med det samme, går man udenom dem. Konfliktsky.	4	3	12	Kommunikér og udvis respekt over for hinanden og lyt til hinanden.
r6	Manglende løsning af ansvaret.	Følelse af afmagt, udskyder alt i sidste øjeblik. Manglende troen på egen formåen. Sygdom blandt medlemmerne i gruppen.	3	2	6	Forvetningsafstemme for hvem skal lave hvad. Uddale ansvarsområder til de rette kandidater. Ikke være bange for at spørge om hjælp. Udvise empati.
r7	Manglende identifikation af komplekse funktionaliteter.	Manglende viden eller teknologiske egenskaber til at kunne identificere.	3	3	9	Opsøg viden og informationer (obs! Være kritisk).
r8	Uventede udvidelser af projektets omfang.	Manglende overblik, pludseligt forekomme af ændringer. Nye krav til projektet.	3	5	12	Vær omstillingssparat, planlæg og få lagt en ny strategi hurtigst muligt.
r9	Forkerte tidsestimeringer.	I starten af projektet bruger man slet ikke nok tid til at tale om formålet med projektet og hvad der præcis skal leveres i projektet. Vi lader det være til tilfældigheder.	3	4	12	I starten af projektet bruger man slet ikke nok tid til at tale om formålet med projektet og hvad der præcis skal leveres i projektet. Vi lader det være til tilfældigheder.
r10	Vanskeligheder med at implementere.	Produktet er kompletst eller de stillede krav synes at være tårnhøje.	2	2	4	Produktet er kompletst eller de stillede krav synes at være tårnhøje.
r11	Forkert uddeling af ansvarsområder.	Alle ressourcer som medlemmerne i gruppen, individers færdigheder osv., spores ikke korrekt.	2	1	2	Igen forventningsafstemme for hvem skal udføre hvilke opgaver. De rette kandidater de rette opgaver.
r12	Githubserver går ned.	"Force majeure" Naturekatasstrofer såsom storm og orkan, kan trære ramme ledningerne og forårsage strømafrydelse i githubs datacentre og på den måde slukke for alle serverne. Serveren udsættes for angreb af hackers.	4	1	4	Det forventes at vi stadig arbejder videre med projektet. Tag fat der hvor der kan tages fat i. Fasthold arbejdsmoralen.
r13	Versionsstyringsproblemer (Mergekonflikt etc.).	Der merges til de forkerte branches eller i værste tilfælde ikke aner hvordan man merger to branches sammen. Man glemmer at committe samt pushe for hver gang der tilføjes eller redigeres i koden og på den måde mistes data, vigtige kodestumper- og programmer.	3	2	6	Lav dit eget branch ud fra masteren i github og udfør dit arbejde deri. Husk altid at comitte og pushe ens arbejde på github. Merge kun med masteren, hvis det er brugbart og hvis du er i stand til at merge dit eget branch med masteren ellers spør om hjælp.

De fundne risici bliver sat ind i dette skema (Risiko Matrix) som yderligere kan give et overblik over alvorligheden af hver risiko.

			<ul style="list-style-type: none"> • Uventede udvidelser af projektets omfang 		
5			<ul style="list-style-type: none"> • Utilstrækkelig viden eller færdigheder. • Forkerte tidsestimeringer 		
4			<ul style="list-style-type: none"> • Gruppemedlemernes manglende deltagelse samt bidrag til projektet. 	<ul style="list-style-type: none"> • Manglende kommunikation i teamet. • Manglende identifikation af komplekse funktionaliteter. 	<ul style="list-style-type: none"> • Manglende håndtering af konflikter.
3			<ul style="list-style-type: none"> • Vanskeligheder med at implementere. • Gruppemedlemmer dukker ikke op til møder. 	<ul style="list-style-type: none"> • Manglende løsning af ansvaret. • Versionsstyringsproblemer (mergekonflikter). 	
2			<ul style="list-style-type: none"> • Forkerte uddeling af ansvarsområder. 		<ul style="list-style-type: none"> • Githubserver går ned.
1					
	1	2	3	4	5
	Konsekvens →				

Figure 4: Risiko Matrix

4 Teknisk Afsnit

4.1 Analyse af problemet og Design af løsning

4.1.1 Analyse af problemet

Spillet 7-kabalen er et simpelt kortspil, der består af et sæt regler. Helt kort fortalt så gælder det om at samle alle kort, således at de findes i deres respektive kulør, nemlig fra es til konge. Man skal altså have skabt fire bunker i foundation hvilket er det hjørne på spillets plade, som alle kulør, spar, hjerter og ruder samles. Hertil skal det nævnes, at man kun kan lave disse bunker i rækkefølgen fra es til konge.

4.1.2 Design af løsning

Nedenunder (*se figur 5*) vises et interaktionsdiagram af spillet. Med dette diagram forsøges der på at opfange samt beskrive den dynamiske interaktion, der er mellem spillets objekter i en single use-case.

Som det fremgår af UML-diagrammet, så består spillet af objekterne User Interface (UI), Camera, Detector, SolitaireGame, SolitaireSolver, som alle sammen er klasserne der er blevet defineret i Android Studio. De beskeder eller data om man vil, som sendes mellem objekterne, repræsenteres med

metoderne der er blevet defineret i deres respektive klasser, ovenover pilene mellem de rektangulær tidslinjer. Det ses også at beskederne som brugeren får i form af tilbagemeldinger, er repræsenteret som tekst ovenover de stiplede pile. Disse tilbagemeldinger informerer brugeren om hvordan denne skal bære sig ad for at kunne udføre 7-kabale.

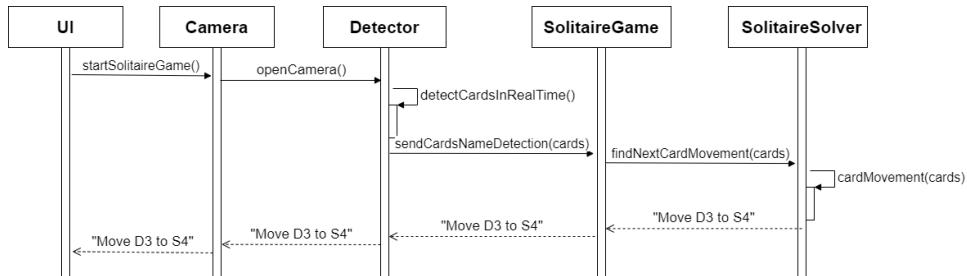


Figure 5: Interaction Diagram

Efter brugerens mobilkamera er blevet aktiveret vha. metoden **openCamera()**, bliver der kaldt på metoderne fra Detector klassen, hvor her bliver 7 kort læst og deres værdier gemt. Disse data sendes til klassen SolitaireGame (dette sørger metoden **sendCardsNameDetection()** for) der har som job at informere brugeren om de skridt der skal tages for at kunne udføre spillet. Informationerne til brugeren sendes som "string" til brugeren, der grundigt instruerer denne om, hvordan kortene skal flyttes og placeres.

Som det er blevet nævnt tidligere, så har 7-kabalen et sæt regler, der er nødvendige at følge for at spillet kan fungere på en korrekt måde. I vores løsning er det en nødvendighed for os, at danne yderligere regler til at løse spillet. Disse regler er fundamentet for vores SolitaireSolver, den følgende figur (*se figur 6*) repræsenterer algoritmen som bliver udført af SolitaireSolver klassen og giver programmet mulighed for at spille et 7-kabale spil.

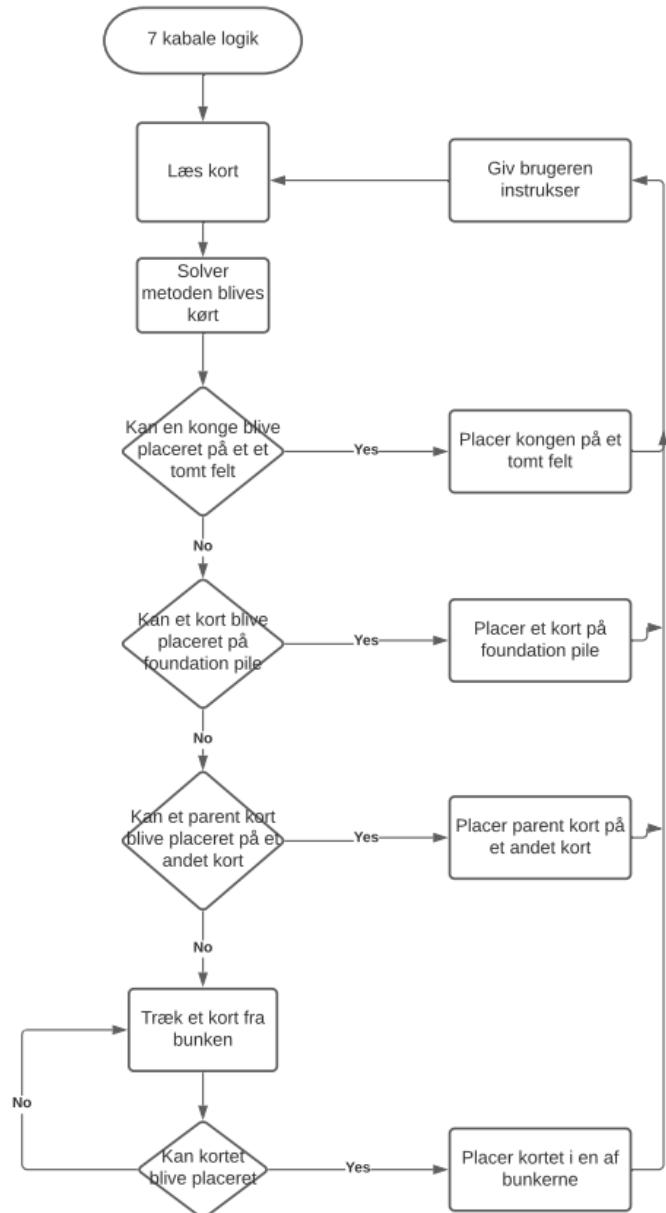


Figure 6: Flowchart Diagram

Som det kan ses, så starter algoritmen efter kortene er blevet læst og detekteret. Dernæst bliver data'en sendt videre til solver metoden som returnerer instrukser. Først tjekker vi om en konge kan blive placeret på et tomt felt. Hvis dette ikke er muligt tjekker vi om et kort kan blive placeret på Foundation pile (*Foundation pile er navnet for de 4 bunker vi samler kortene sammen*). Sidst tjekker programmet om et parent kort kan blive placeret på et andet kort (*Vi definere et parent kort som det kort med højest værdi i en bunke*). Hvis ingen af tidligere tilfælde er mulige, bliver programmet nød til at trække et deck fra decket som brugeren bliver nød til at scanne, hvilket giver programmet mulighed for se om en placering er mulig. Når man går gennem alle kort i decket uden nogle mulige træk betyder det at spillet er slut. Vores program giver ikke brugeren mulighed for at scanne alle synlige kort, hvilket betyder at vi håndterer disse søjler af kort i forskellige datasstruktur for at give programmet overblik over spillet.

4.2 OpenCV

Vi har lavet vores billedgenkendelse med OpenCV i Pycharm, hvor vi bruger Python. Vores logik er lavet i IntelliJ IDEA med Java.

4.2.1 System komponenter

For at køre vores program med OpenCV, benytter vi os af følgende komponenter:

- Et sæt spillekort
- En stationær computer med 8 kerner.
- Et kamera (hvilket som helst kamera som kan forbindes til computeren)
- Et tastatur

4.2.2 Systemet

I IntelliJ har vi fået lavet en server, hvor vi åbner en socket på port 8000, for at modtage data fra en klient. Her har vi lavet vores klient i Pycharm. Når man derfor skal køre vores system, er man først nødt til at starte serveren i IntelliJ, og derefter klienten i Pycharm. For at køre systemet, skal man forbinde et kamera til sin computer. Det kan fx være en mobiltelefon eller et webcam. Når man kører OpenCV-programmet i Pycharm, åbnes der op for kameraet, hvor man derefter benytter sig af kameraet til at vise systemet spillekortene, så den kan udføre billedgenkendelsen. Vi har lavet vores billedgenkendelse af spillekortene med YOLO Darknet, hvor vi bruger tre filer, en .weight fil (det trænede datasæt), .cfg fil (konfigurationsfilen) og .names fil

(som indeholder labels for alle kort) med OpenCV. Når man retter kameraet mod spillekortene, kan systemet genkende spillekortet, og man trykker derefter på 'a' på tastaturet, for at tilføje et spillekort til en liste. Når man har fået tilføjet kort til listen, sender man dataet over til Java-serveren, ved at trykke på 's' på tastaturet, og derefter 'q', for at lukke kameraet, for at Java-serveren kan modtage dataet og udføre 7-Kabale algoritmen.

4.2.3 Klassediagram

Figur 7 viser klassediagrammet for systemet. Vi har lavet to enumerations (Rank og Suit), som bruges i Card-klassen, til at give hvert kort en rank og et symbol. I et 7-kabalespil har man forskellige typer af bunker. Her har vi lavet en abstrakt Pile-klasse, hvor vi har to typer af piles, TableauPile og FoundationPile, som nedarver fra den. Da en bunke indeholder et antal af kort, har Card-klassen en 0 til mange relation til Pile-klassen. Når data bliver sendt til Server-klassen, bruger den SolitaireGame-klassen til at sætte spillet i gang, som så bruger SolitaireSolver-klassen til at beregne det næste træk.

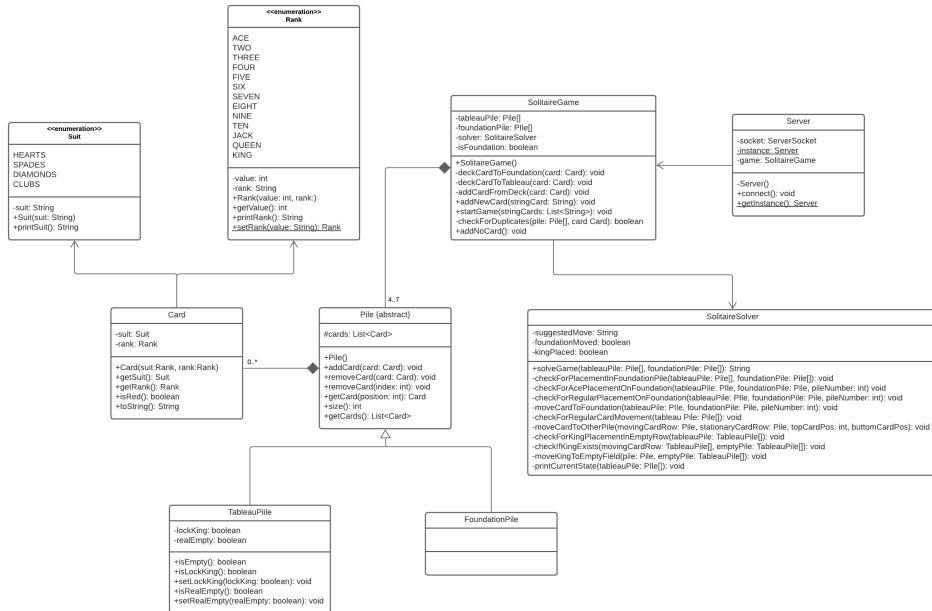


Figure 7: Klassediagram

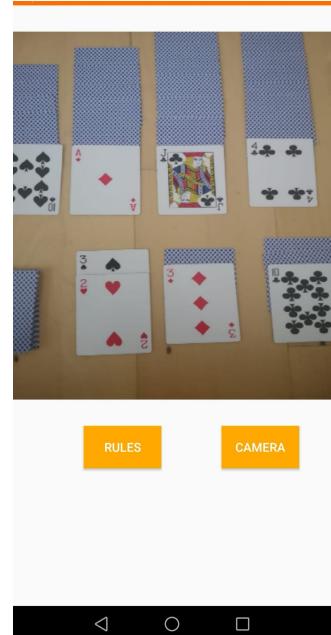
4.3 TensorFlow lite - Android app

Vi har arbejdet med TensorFlow Lite i android studio, hvor det er lykkedes os at udvikle en app der bruger mobilkamera til at læse og detektere kort og på denne måde hjælpe brugeren til at spille samt løse 7-kabalen.

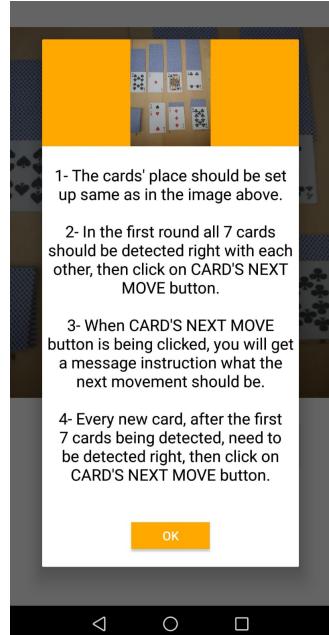
I det følgende forklares hvordan vi har båret os ad med designet af løsningen.

4.3.1 Om Produkt

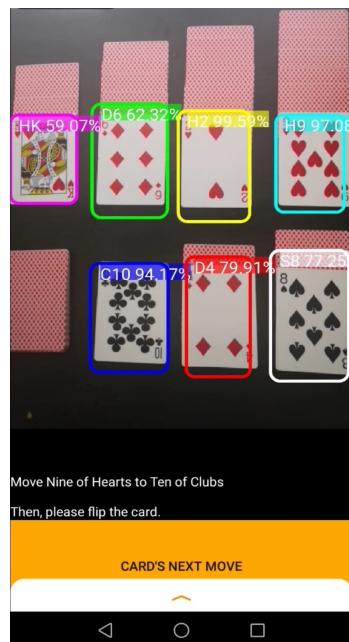
Produktet er en Android App, der kan hjælpe en bruger med at løse 7-kabalen. Appen er simple, den består nemlig kun af to sider. Når brugeren starter appen, har denne to valgmuligheder i form af to knapper der kan trykkes på. Den ene knap viser spillets regler, der kort fortæller hvordan appen bruges og den anden knap er CAMERA knappen. Inden man starter med at spille 7-kabalen, skal man først have sat sig grundigt i spillets regler, som findes i appen. For at starte spillet, trykkes på CAMERA knappen, hvor brugerens mobil kamera bliver aktiveret. Herefter starter appen med at detektere kort. Først skal alle 7 kort i tableau blive detekteret korrekt, og herefter trykkes på CARD'S NEXT MOVE knappen, så vil programmet returnere en besked, hvor den handler om det næste skridt.



(a) Første side i app



(b) 7-kabale regler dialog



(c) 7-kabale løsning side

4.4 Fejlkilder

Langsom detektion i Android	Appen har en langsom detektions tid på ældre telefoner, da de ældre telefoner ikke er ligeså hurtige som de nyere modeller.
Forkert billedegenkendelse	Til tider kan systemet genkende et spillekort forkert. Den kan fx nogle gange læse en 5 spar som en 5 kløver

5 Skrivning til ingeniørstuderende, som skal starte på CDIO kurset

CDIO-Projekts mål er at gøre den studerende i stand til at håndtere en komplet systemudvikling og gøre det til et vellykket færdigt produkt. Kurset indeholder både softwaremæssige og hardwaremæssige aspekter og bestræber sig på at skabe arbejdsmiljø som ligner de reelle arbejdsforhold i IT-virksomheder, og vil træne den studerende i at samarbejde i et udviklingsteam. Dette kursus giver de pågældende studerende erfaring indenfor ledelses området, risiko håndtering, projektstyring og software udvikling.

Det råd, vi kan give til dette kursus er følgende:

- Analysis: Lige gyldig hvilken form for opgave i får stillet i CDIO-kurset, så er det altid en god ide at starte med at analysere problemet og benytte sig af divide and conquer metoden, der går ud på at dele problemet i små dele og takle dem hver især.
- Search: En god tommelfingerregel er at man aldrig skal give op og aldrig skal fortælle sig selv at denne opgave er umulig at få gjort færdig. Der er altid mulighed for at søge på Google, her kan du næsten 100% af tilfældene finde det du erude efter. Men husk at man ikke skal tage tid på at lave alt dette Search, det er vigtigt at efter de rigtige keywords som beskriver problemet bedst, som i derefter kan bruge til at søge efter på Google.
- Trust your TEAM: Det er meget vigtigt at i deler små opgaver mellem hinanden, så hver person får ansvaret for en opgave. Det er her vigtigt at i altid tror på hinanden og i ikke siger, "Ingen i gruppen kan løse disse opgaver" for der er altid en løsning, så bare have tillid til hinanden også kommer det.

6 Konklusion

Det kan konkluderes, at vi i gruppen har færdiggjort et vellykket CDIO-projekt i kurset 62410. Vi har formåret, at imødekomme de stillede krav der var til projektet og det er gået over alt forventning. Successen skyldes den nøje planlægning af projektet helt fra dets spæde start og ikke mindst gruppemedlemmernes enorme arbejdesvillighed igennem hele forløbet. Vi hver især arbejdede med et ansvar under frihed, som blev respekteret, hvorfor dette har været et vigtigt grundlag for løsning af problemstillingen som kommer i hele to former.

Grundlaget for ethvert vellykket projekt er netop den effektive planlægning. Til denne gjorde vi brug af Trello, som er en web-baseret platform og et godt værktøj til styring samt organisering af især softwareprojekter. Her har vi opdelt det overordnet projekt i små opgaver med fornuftig deadlines sat til enhver delopgave.

Som det tidligere er blevet nævnt, så kommer vores løsning i to former. Vi har udviklet to velfungerende systemer, der aflæser tilfældige kort fra enten et mobilkamera eller et almindelig eksternt kamera, og hjælper med at løse 7-kabalen. Mobilapplikationen er blevet udviklet i Android Studio. Hertil brugte vi tredje partsbibliotekerne Tensorflow Lite samt YoloV4, mens i forhold det andet system brugte vi OpenCV og Pycharm. Og logikken til begge softwaresystemer blev programmeret i Java. Vi har yderligere understøttet vores logik med nogle design diagrammer, der tydeliggøre konceptet og sammenhængen.

Forløbet er gået over alt forventningen trods omstændighederne, grundet pandamien Covid-19. Det har været en smule udfordrende, at skulle lave projektet online, og det tør siges at undervisning generelt ikke fungere så godt som det fysiske. Men trods disse særlige omstændigheder var vores motivation højt og vi var altid opmærksomme på at vedligeholde denne under hele forløbet. Derfor bestræbte vi os på at opnå vores endelige mål så godt som muligt, således at vi i gruppe 16 samt vores undervisere (interessenter,stakholders) alle er glade og tilfredse over den endelige produkt.

Her til sidst skal det nævnes, at til den nærværende rapport er der vedhæftet en kort videopræsentation af vores projekt, der forklarer systemets komponenter og detaljer og giver en demonstration af det færdigudviklet IT-system der løser 7-kabalen.

7 Bilag A: Arbejdsfordeling

Nedenstående skema dokumenterer hvor meget hvert gruppemedlem har bidraget til hver enkelt delopgave i projektarbejdet, video og rapportskrivning.

7.1 Delopgave

Områder/Navne	Khaled	Anthony	Bashar	Sid	Fadl	Mohamad	I alt
Management	10	20	20	10	20	20	100
Design	35	0	0	35	0	30	100
Logic - Implementation	40	0	0	40	15	5	100
OpenCV	90	0	0	10	0	0	100
Tensorflow lite - Android studio	0	20	10	20	0	50	100
Test	16.67	16.67	16.67	16.67	16.67	16.67	100.02

Table 1: Tabel over timeregnskab

7.2 Rapportafsnit

Områder/Navne	Khaled	Anthony	Bashar	Sid	Fadl	Mohamad	I alt
Resume	0	50	0	0	50	0	100
Projekt Plan	0	0	33.34	0	33.34	33.34	100.02
Risiko Analyse	0	50	50	0	0	0	100
Teknisk Afsnit	25	25	0	25	0	25	100
Skrivning til ingeniører	0	0	33.34	0	33.34	33.34	100.02
Konklusion	0	50	50	0	0	0	100

Table 2: Tabel over timeregnskab

7.3 Videoafsnit

Områder/Navne	Khaled	Anthony	Bashar	Sid	Fadl	Mohamad	I alt
Presentation	0	0	0	0	100	0	100
System komponenter	40	0	0	0	60	0	100
OpenCV Demo	100	0	0	0	0	0	100
Android Demo	0	70	0	0	0	30	100
Teknisk Detalje	0	0	0	100	0	0	100

Table 3: Tabel over timeregnskab

8 Bilag B: Kilderne

References

- [1] Collect Data: 80% of data come from this Github:,
https://github.com/tooploox/yolo_android/tree/master/dataset
- [2] Roboflow: Annotate, Training the datd, and deploy the model,
<https://blog.roboflow.com/how-to-train-a-custom-mobile-object-detection-model/>
- [3] Tensorflow lite: Android studio deployment,
<https://github.com/hunglc007/tensorflow-yolov4-tflite>
- [4] YOLO OpenCV Object Detection,
https://www.youtube.com/watch?v=GGeF_3QOHGE/