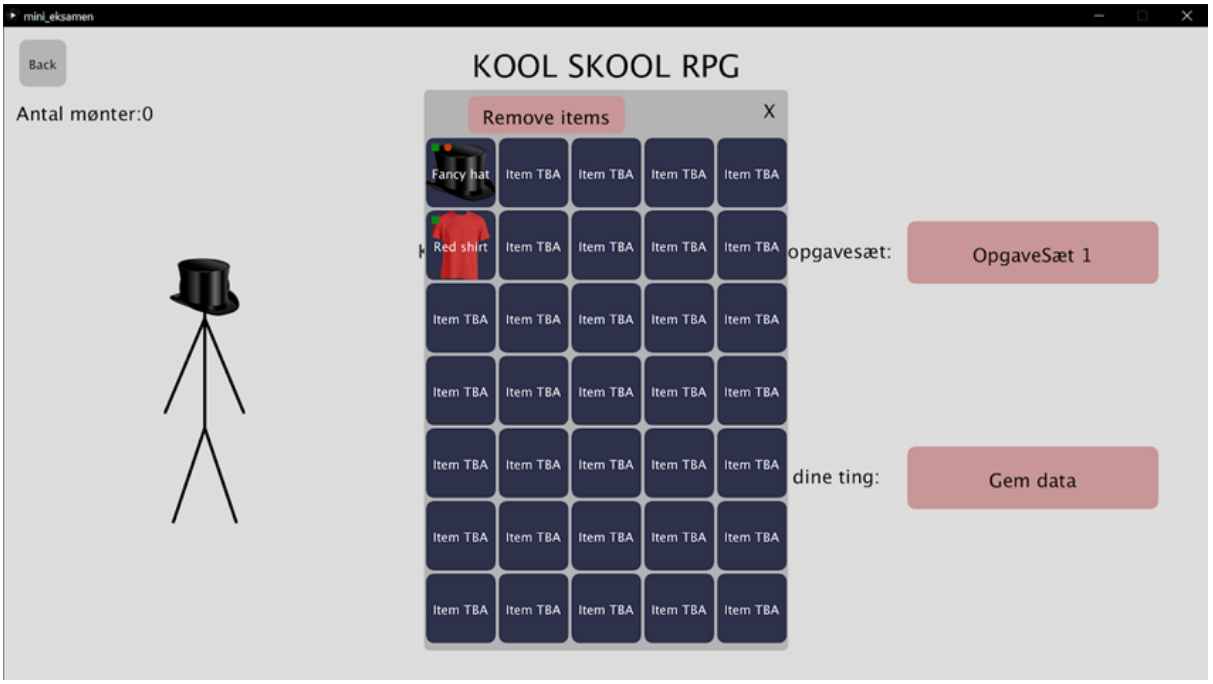
Udvikling af læringsprogram

Mini-eksamensprojekt i Digital Design og Udvikling



Frederik Cayré Hede-Andersen, Jonas Borup Randløv og Viktor Tranto Bräuner 3.a2

H.C Ørsted Gymnasium Lyngby

10/11/2021 - 19/12/2021

Vejledere:
Anders Juul Refslund Petersen (ajrp) & Kristian Krabbe Møller (kkm)

Teknikfag digital design og udvikling: Mini eksamensprojekt

3.a2/DDU 1 TK (årgang 2021 - 2022)

Resumé:

I den følgende rapport gennemgås en problemanalyse, hvor der slås ned på elevers udfordringer med at finde motivation i undervisning. Dette lægger op til et program, fundet gennem ideudvikling, hvor andre ideer blev kasseret efter overvejelse, der udvikles, der sættes fokus på at motivere eleven ved brug af gamification og cosmetics, samtidig med at give læreren mulighed for at teste eleverne. Konceptet uddybes og projektstyring gennemgås. Den tekniske dokumentation redegøres for. Undervejs i udviklingen blev enkelte elementer, dog ikke uforventet, skåret fra. Endeligt snakkes der om de tests af produktet, der blev udført, inden produktet diskuteres og der konkluderes, at produktet lever op til de krav, som opgaven satte.

Forord

Dette teknikprojekt i faget digital design og udvikling er lavet i forbindelse med forberedelse til det kommende eksamensprojekt. Givet for projektet var den opgave at fremstille et produkt, der i et undervisningssammenhæng ville kunne blive brugt til at en lærer kan oprette opgaver og få oversigt over resultaterne for en klasse på 30 elever. Det angives, at denne lærer har almindelige it-kompetencer. Projektet, som denne rapport redegør for, forløb fra den 10/11/21 og havde en afleveringsfrist den 19/12/21. Dette gav gruppens medlemmer 42 moduler at arbejde på projektet med, men i praksis var dette 38 moduler, idet skemalægning forhindrede deltagelse i 4 af modulerne nær slutningen af projektet pga ekskursion i andre fag.

Indholdsfortegnelse

Projektbeskrivelse

Som en del af den digitale omstilling flytter store andele af redskaberne anvendt i forbindelse med undervisning over i den virtuelle verden, ikke blot i forbindelse med nedlukning og deraf virtuel undervisning.[1] Det er dermed vigtigt, at lærerne bliver forsynet med digitale redskaber, der lader dem engagere eleverne samtidigt med, at læring stadig står i fokus.

Problemanalyse

Undervisning i folkeskolen kan for eleverne ofte blive kedelig og langtrukken, hvilket kan blive demotiverende og hæmme indlæringen, ifølge folkeskolernes trivselsundersøgelse i 2018. En tredjedel af folkeskoleelever fra 4.-9. klasse mener at undervisningen tit er kedelig, hvilket er en større andel end de tidligere år[2].

Dette kan blandt andet være grundet børnenes dårligere koncentrationsevne, sammenlignet med unge og voksne. Til dette kan der bruges forskellige programmer og værktøjer, der også kan gøre det nemmere for lærerne at følge med i børnenes læringsproces og udvikling. Disse programmer hjælper med at ændre børnenes indstilling til undervisningen, til en mere positiv en[3]. Der findes dog ikke mange forskellige muligheder i forhold til undervisningsprogrammer, og en del af dem er ikke optimale til både at holde undervisningen interessant, samtidig med at lærerne nemt kan se udviklingen i elevernes faglige kompetencer. Læringen i disse programmer opdeles tit i mindre spil, hvilket kan gøre det svært for eleverne at se det større billede[4].

Programmet har to forskellige brugere hvilket også medfører to forskellige brugssituationer. Den første slags bruger er folkeskoleelever, hvilket vil betyde børn og unge i en alder fra 6-15 år. Brugssituationen for dem vil være læring, enten i undervisningen eller som en lektie i hjemmet. Den anden bruger er en lærer, der skal bruge programmet til at holde styr på deres elevers faglige udvikling.

Problemformulering

Hvordan kan man med en digital løsning skrive et program der hjælper med at teste en elevs faglige kompetencer?

Kravsopstilling

I forbindelse med denne problemformulering blev der opstillet en række krav, der blev separeret i hårde- og bløde krav såvel som nice to haves. Disse krav blev derefter prioriteret ud over deres kategoriske inddeling. Disse krav anvendes i forbindelse med at vælge den rette ide at arbejde videre med, hvilket beskrives i nærmere detalje i det følgende afsnit, idéudvikling. Ud over dette vil disse krav senere hen også lægge grundlaget for de user stories, som er udgangspunkt for programmets design. Disse krav ses opstillet herunder.

Hårde krav:

- Måde for læreren at lave og tilrettelægge sine egne spørgsmål/opgaver, 8
- Måde for læreren at sammenligne resultater, 7
- Måde for eleven at svare på spørgsmål/opgaver, 10
- Måde for dataene at blive gemt og flyttet, 10

Bløde krav:

- Programmet skal være interessant for eleverne at bruge flere gange over en længere periode, 6
- Programmet skal fremme elevernes læring, 5
- Intuitivt og nemt at bruge, 5
- Nemt at sætte op på flere computere, 3
- Svardata kan kun læses af læreren, 3

Nice to have:

- Cool graphics, 5
- Eleven kan tracke egen progress, 2

Idéudvikling

Med passende krav til produktet opstillet kunne idégenereringen begynde. Gruppens medlemmer gjorde brug af individuel brainstorming, hvor der blev arbejdet ud fra og videre på de digitale hjælpemidler og redskaber, som de selv havde stødt på i deres skolegang. Dette gav en række ideer, som ses længere inde i dette kapitel. Forinden denne individuelle brainstorm havde gruppemedlemmerne en samtale om deres forskellige umiddelbare idéer, hvilket også ses afspejlet i de fundne idéer, eftersom nogle af disse diskuteret temaer såsom "gamification" går igen.

Disse idéer blev da skrevet sammen, således at de koncepter, der gik igen, blev forenet. Dette gav en række konkrete koncepter, som der blev uddybet på. Disse ideer gennemgås længere inde i kapitlet.

Herfra blev et PxV-skema (se figur 1) for ideerne brugt til at beslutte ideen, der blev arbejdet videre med. Som lagt op til, blev de hårde og bløde krav, der tidligere blev nævnt, brugt i skemaet. Ideerne blev givet point fra 1-5 alt efter, hvor godt de opfyldede forskellige krav, og de blev herefter ganget med kravets vægtning. Det blev valgt ikke at inddrage den tekniske løsning i PxV-skemaet, men udelukkende at bruge det at bestemme programmets koncept. Den valgte løsning for det tekniske design beskrives senere. Som en følge af det faktum, at det tekniske aspekt ikke inddrages i PxV-skemaet, medtages de to følgende krav ikke.

- Måde for dataene at blive gemt og flyttet, 10
- Svardata kan kun læses af læreren

Følgende er ideerne, der ses i PxV skemaet:

Abacus 2:
Et matematikprogram der er inspireret af programmet abacus, hvor der arbejdes videre på konceptet. Nøglefunktionaliteten ville her være adaptive autogenerede opgaver i forskellige kategorier. Programmet ville været rettet imod 7-9 klasses elever. Denne ide scorede højt på mange punkter, da abacus er et ganske velfungerende program, men scorede lavest på kravet "Interessant". Dette skyldes, at produktet langt hen ad vejen ville ende op med at være en kopi af abacus uden nogen væsentlige ændringer bortset fra nogle få ideer, der ville blive fundet under yderligere arbejde med konceptet.

Gameshow

Gameshow gik ud på at kombinere forskellige velkendte spilformater såsom Jeopardy, The Price is Right og Family Feud med læringsorienteret opgaver og spørgsmål. Spillet ville foregå med 2 hold og formatet ville skifte undervejs for at skabe mere variation. Denne ide scorede lavest i PxV skemaet, hvilket blandt andet skyldes ideens løse rammer, som gjorde det uhåndgribeligt at forstå præcist hvor godt den ville fungere.

Lite RPG

Lite RPG ideen opstod originalt som en kombination mellem "RPG" og "Abacus 2". Her kombineres elementer fra RPG, der lader en have en karakter, der repræsenterer ens fremskridt med den høje modularitet format af Abacus 2, der tillader læringstungt gameplay. Her belønnes eleven for at besvare opgaver, og denne belønning kan bruges på at udsmykke deres karakter, således at opgaverne giver eleven noget konkret at arbejde sig hen til. Denne ide scorede højest og blev arbejdet videre med.

Mars Metropolis

Mars Metropolis er inspireret af det gamle men populære matematikspil, Matematik i Måneby, der har opnået kult-status blandt årgangene fra de tidligste 2000'ere. Her kunne en elev prøve magter i forskellige minispil, der hver testede elevens matematiske egenskaber i forskellige discipliner på nye og sjove måder, samtidigt med at de spillede ind i historien i byen, hvor spilleren skulle hjælpe med Månebys mange forskellige gøremål.

RPG

Ideen RPG er et spil med gameplay i stilen af old school jRPGs, hvor encounters er matematikopgaver med tid på. Spillet er inspireret af "Learn Japanese to Survive!" serien. Læreren kan da spore progress ved at se, hvor langt i spillet man er kommet. Som historien progresser bliver man introduceret til mere avanceret matematik. Spillet er sigtet mod 7-9 klasse, men det tager i senere dele af historien også hul på 1.g matematik.

Krav	Vægt	Abacus 2	Minigames	RPG (lite)	Mars metropolis	RPG	Gameshow
Lave og tilrettelægge egne spørgsmål	8	4 | 32	4 | 32	5 | 40	4 | 32	5 | 40	4 | 32
Sammenligne resultater	7	5 | 35	3 | 21	5 | 35	3 | 21	3 | 21	2 | 14
Elev kan svare på spørgsmål	10	5 | 50	5 | 50	5 | 50	5 | 50	5 | 50	5 | 50
Interessant	6	1 | 6	3 | 18	4 | 24	5 | 30	5 | 30	4 | 24
Fremme læring	5	5 | 25	3 | 15	5 | 25	3 | 15	3 | 15	5 | 25
Intuitivt	5	5 | 25	4 | 20	4 | 20	3 | 15	2 | 10	5 | 25
Sætte op på flere computere	3	4 | 12	4 | 12	4 | 12	4 | 12	3 | 9	4 | 12
Sum	-	185	168	**206**	175	175	182

Figur 1: PxV skema brugt til at bestemme produktkoncept. I kolonnerne for de individuelle ideer ses de opnået point og dernæst værdien efter vægtningen er ganget på.

Koncept

Som set i afsnittet forinden blev det valgte koncept et form for spil med lette roleplaying elementer, der gennem gamification skal bruges til at undervise i matematik.

Dette produktkoncept bygger op om nogle forskellige ting. Der er blevet gjort overvejelser angående, hvem produktet er rettet imod, den tekniske udformning af produktet og det visuelle aspekt af produktet.

Programmets brugere

Programmet, der som belyses under det følgende afsnit, teknik, og dernæst layout, dækker over 2 forskellige tilstande, der hver især er lavet til programmets forskellige typer af brugere, lærere og elever. Det er selvfølgeligt vigtigt at tage hensyn til begge af disse.

Læreren

Lærerens rolle er at oprette opgavesæt til eleven. For at kunne gøre dette skal læreren have en hvis mængde teknisk know-how, men da ikke alle lærere er lige gode med computeren er det vigtigt at holde dette så simpelt og intuitivt som muligt. Dette er der dermed fokus på, når den del, som læreren interagerer med designes. Brugergrænsefladen skal være intuitiv, og de tekniske dele, som er nødvendigt at læreren interagerer direkte med skal komme med instruktioner.

Eleven

Elevens to vigtigste opgaver i programmet er at kunne besvare de opgavesæt, som læreren giver, og at kunne udsmykke deres karakter. Det lægger for kernen af programmets koncept, at eleven kan motiveres af at få adgang til forskellige cosmetics. Denne spillestil og type af belønning er dog allerede blevet afprøvet og har vist sig at være ganske successful i spil, og spiller på nuværende tidspunkt en stor rolle i det økonomiske aspekt af spilindustrien.[5] Ud fra dette formodes det, at cosmetics ville have en motiverende effekt for eleverne, og på den måde kan hjælpe eleverne på vej med at få lavet de opgaver, som læreren giver. For at sørge for, at disse cosmetics ikke føles gemt væk er det også vigtigt, at en elevs karakter er tilstede i så megen tid som muligt, når programmet er i brug, ligesom den ville være i en normal RPG.

Teknik

Mere detaljeret og teknisk, så fungerer programmet således, at man som en lærer, der åbner programmet, kommer ind på en opgave-opretter. Her beskrives der, hvordan opgavesættene oprettes, og forbindes til et kort over opgavesæt. Læreren kan oprette opgaverne og de bliver derefter formateret som en XML-fil. For at indblande kryptering bruges private og public keys. Idet læreren færdiggører opgavesættet bliver en public key lagt ind i XML filen, og programmet viser en public key, som læreren så kan skrive ned. Når eleven gemmer sine resultater, bliver de automatisk krypteret ved hjælp af den public key, der er i opgavesættet. Når læreren modtager svar-filerne og loader dem ind i sit program, kan de skrive sin private key ind for at dekryptere opgaverne.

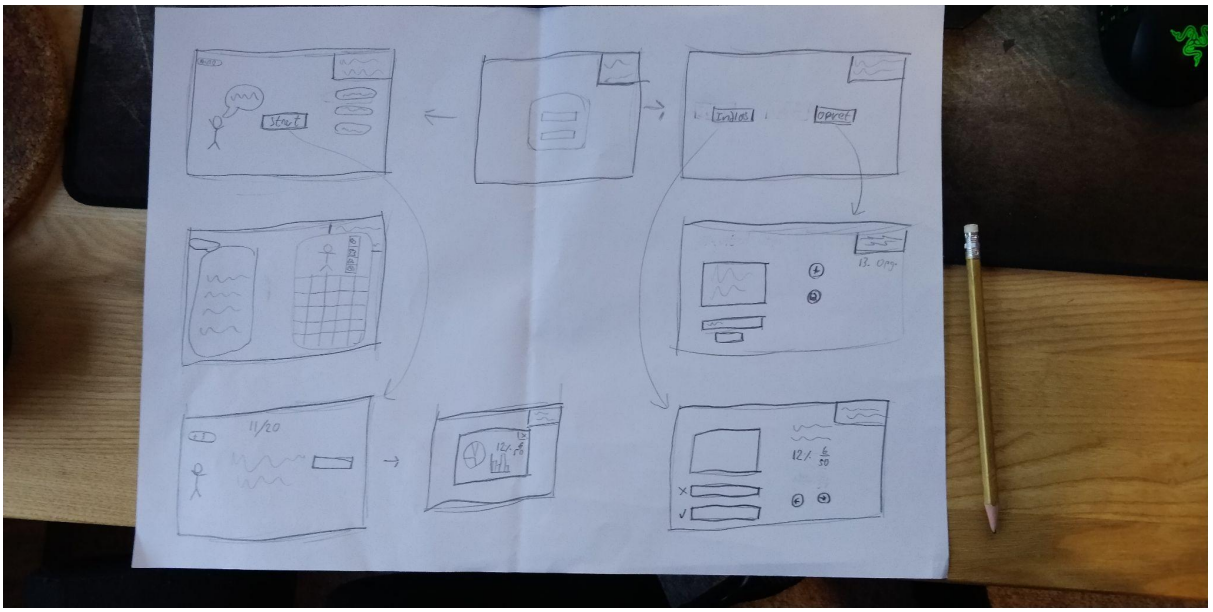
Eleverne kan få denne fil af læreren, i for eksempel en lectio-besked eller forældreintra-besked, hvis programmet bruges i en folkeskole, eller eventuelt gennem et USB-stik. Filen kan åbnes med programmet, hvor eleven skal indtaste deres navn. Efter dette kommer de ind på et kort over opgavesættene. På kortet ses også en karakter, der står på startpunktet/start-opgavesættet. Når eleven begynder på et opgavesæt, vil de få de opgaver læreren har lavet i en tilfældig rækkefølge. Efter at have besvaret opgaverne vil eleven få et antal af mønter, baseret på opgavens sværhedsgrad og hvor mange rigtige spørgsmål de har besvaret. Sværhedsgraden er bestemt af læreren. Med disse mønter kan de købe cosmetics til deres karakter, for eksempel hatte, beklædning eller andet. Når et opgavesæt er besvaret kan eleven gå videre på de designede kort. Der kan være steder på kortet, hvor vejen deler sig eller andet, og antallet af opgavesæt kan varierer, dette kan alt sammen bestemmes af læreren.

Når eleven er færdig, kan de trykke gem. Resultaterne vil både gemmes som en ny XML-fil og i en lokal database. Dette vil gøre så læreren både kan få resultaterne, hvilket vil visualiseres i programmet, når de skriver den korrekte private key. I den lokale database gemmes elevens karakters cosmetics, hvor mange mønter de har fået, og hvor langt på kortet de er.

Denne ide giver læreren mange muligheder for selv at tilrettelægge opgaverne og opgavesættene. Man kan for eksempel bruge et kort gennem et helt forløb, hvor eleverne kan svare på opgavesæt løbende, men man kan også designe et kort, der er lavet til at bruges i en undervisningstime. Derudover er der også mulighed for at plotte elevernes resultater og udvikling i programmet. For eleverne er det en sjov måde selv at se deres egen udvikling gennem deres karakters udseende og cosmetics, hvor der også kan opstå konkurrence om, hvem der kan opnå den flotteste eller sejeste karakter. Læringen er stadig i fokus, hvor opgaverne formateres som normale opgaver, og ikke små spil for eksempel.

Layout

Fra et brugergrænsefladeperspektiv, så beskrives konceptet ved brug af en wireframe, som ses herunder på figur 2.



Figur 2: Wireframe og skitser af UI

På venstre side af papiret ses programmet for en elev, og på højre er en lærers perspektiv. Den øverste skærm er i midten en landingpage, der leder til enten en elevs eller en lærers startskærm. En elev har sin karakter på venstre, som har en sjov besked, der bliver sagt. Der er de tilgængelige opgavesæt på højre, som kan vælges og en startknap i midten. En lærer har mulighed for enten at indlæse en besvarelse eller at oprette et opgavesæt.

Det midterste sæt af skærme viser en elev, der har nogle menuer åbne på venstre og på højre en lærer, der er i gang med at lave et opgavesæt. Eleven har mulighed for at kigge på noget statistik over hvordan de klarer sig, og de kan customize deres karakter. Læreren har mulighed for at indtaste en opgavetekst og et svar, såvel som tal på antallet af opgaver i sættet, en gem knap, ny knap osv. De kan herudover også indtaste, hvor mange mønter opgaven er værd.

På det sidste sæt af skærme ses en elev, der er i gang med at lave en opgave og efterfølgende en skærm med lidt statistik. På højre ses en lære, der er i gang med at kigge en besvarelse igennem. Når eleven er i gang med en opgave er deres karakter ude til venstre. Denne karakter kan evt. også komme med kommentarer som man laver opgaverne, såsom "Godt lavet!", når en elev har svaret på en af opgaverne.

Projektstyring

Kommunikation

For at udvikle dette produkt er en struktureret tilgang til arbejdet nødvendigt. Indledningsvis tilgås projektstyringen ved brug af et github repository, hvor der kan oprettes grene af programmets kode, der senere kan flettes tilbage, således forskellige elementer af programmet kan udvikles parallelt samtidigt af forskellige medlemmer af gruppen.

Gruppen valgte for projektets forløb at arbejde hjemmefra, hvilket selvfølgelig har haft en kommunikativ indflydelse på, hvordan koordinering af projektets fremgang har fungeret. Generelt foregik en arbejdsdag ved, at der blev afholdt et gruppemøde på skolen med medlemmerne samlet, hvor der blev diskuteret, hvor langt i planen arbejdet var kommet i forhold til det forventede fremskridt, hvilket et burn down chart, som vendes tilbage til, blev anvendt til at spore. Efter det var aftalt, hvad folk her især lavede, tog gruppemedlemmerne hjem. Kommunikation mellem gruppemedlemmer foregik herfra på det digitale medie discord, og der blev taget kontakt til vejledere når nødvendigt over microsoft teams.

Planlægning for udvikling

Programmets funktionalitet blev findelt i user stories, der hver især beskriver noget en bruger skal kunne gøre, opleve eller have gavn af igennem programmet. Disse user stories blev da yderligere inddelt i tasks, der beskriver mere tekniske tilgang til, hvordan en user story skal udvikles. Herunder ses user stories med de tilknyttede tasks. Hver user story har en prioritet såvel som et tidsestimat. På baggrund af de erfaringer, der er blevet gjort med tidligere projekter ganges alle estimater med en koefficient på 2.7, hvilket giver en samlet arbejdssum på 105 timer. Idet forløbet har 42 moduler, og gruppen består af 3 medlemmer, er der 126 timer givet til projektet. De resterende 21 timer, der er tilbage, når der ses bort fra tiden, der skal bruges på udvikling, blev brugt på brainstorming, ideudvikling, konceptuddybelse og projektstyring. Her ses disse user stories delt op i de tre iterationer af programmet, som udviklingen foregik ud fra.

Iterationer

Iteration 1 (uge 47 & 48)

- Knap-Klasse
- Startskærm
- UI til design af spørgsmålene/læreren
- Metode til at konvertere til/fra XML-fil
- Kort over opgavesæt

Iteration 2 (uge 49)

- Customizable character
- Pointsystem
- Database

Efter denne iteration kan første test laves. Den laves på bekendte der er en del af målgruppen, nemlig folkeskoleelever. Testen laves som et ustruktureret interview. Da programmet er stort, sammenlignet med tidligere udviklede programmer, er det vigtigt at det er intuitivt. Derfor får testpersonen forskellige opgaver de skal løse - for eksempel "Opret et opgavesæt," eller "Køb en cosmetic til din karakter." Testen optages, og ud fra det kan det observeres hvor nemt det er at bruge programmet. Derudover bliver testpersonerne også spurgt ind til programmet, efter de har prøvet det.

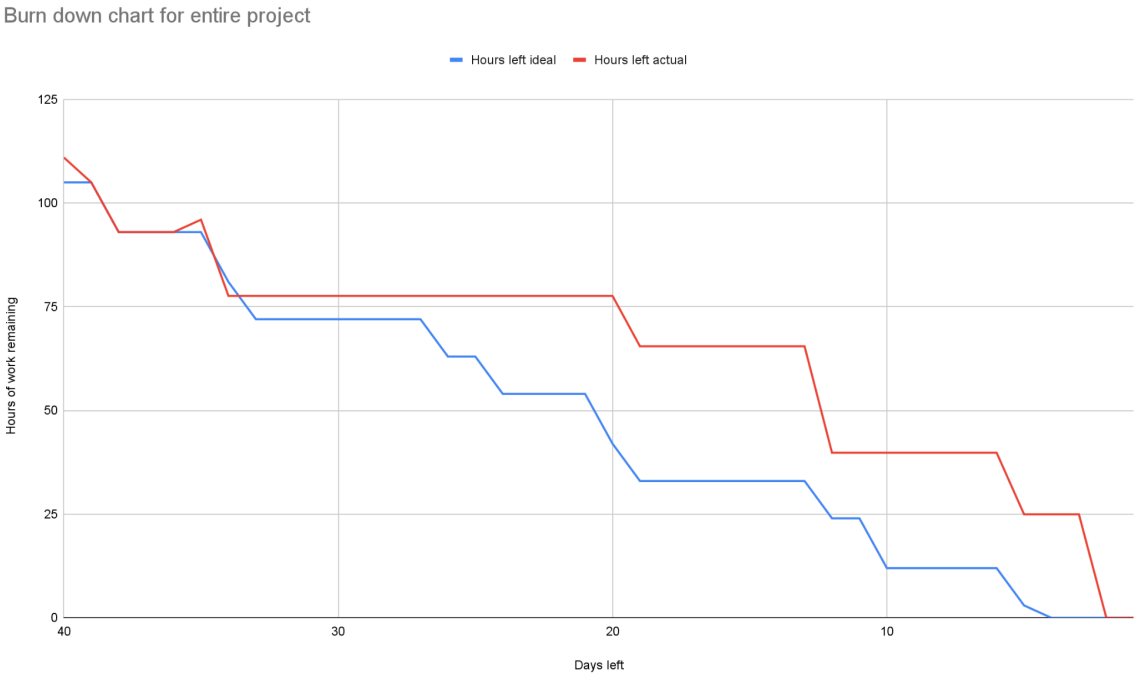
Iteration 3 (uge 50)

Kryptering
Cool graphics

En anden test laves når programmet er færdigt efter sidste iteration. Testen laves på samme måde som den første test. Den sidste iteration dækker udvidelser, som ikke er nødvendige for programmet, men ville finpudse programmet at få med. Disse er dermed kun noget, der sættes fokus på, hvis der er god til tilovers, når uge 50 nås.

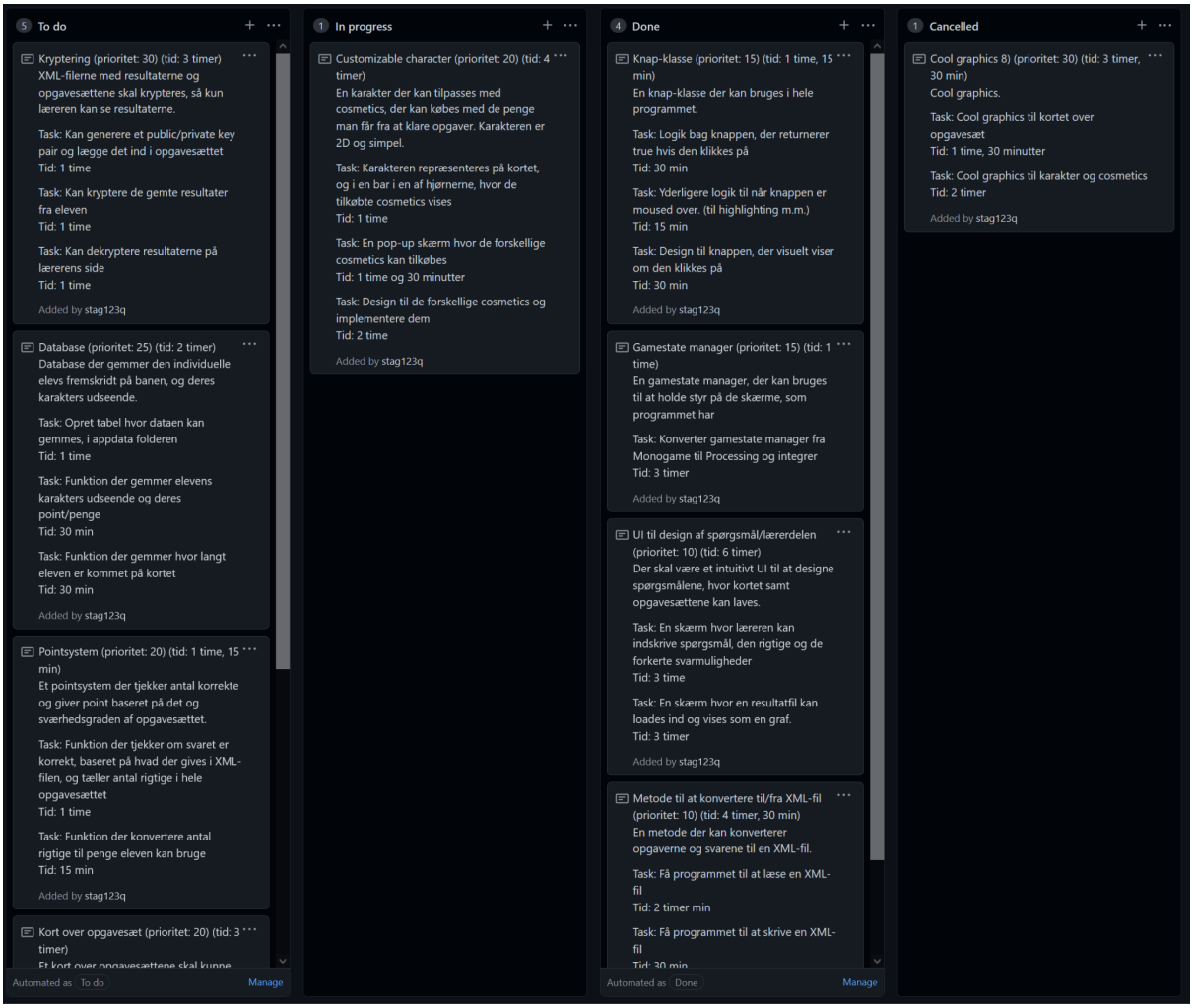
Burn down chart og projekt board

Ydermere gøres der brug af et burndown chart (se figur 3), der sporer fremgangen i arbejdet sammenlignet med den forventede fremgang.



Figur 3: Burndown chart for det fulde projekt

Denne fremgang spores ikke blot i arbejdstimer men også tilstanden af diverse user stories, som set i det anvendte github project board (se figur 4).



Figur 4: Screenshot af project board undervejs i projektets forløb. User stories er fordelt på 4 kategorier, to do, in progress, done og cancelled.

Ændringer i planlægning

Som ses afspejlet i project boardet, hvor en user story er blevet placeret i "cancelled" og burn down chartet har der været enkelte afvigelser fra den originale plan, der blev lagt under den indledende projektstyring.

Statusmøder blev afholdt undervejs i projektets forløb, typisk før gruppens medlemmer tog hjem og arbejdede individuelt. Det blev til et af disse aftalt at aflyse Cool Graphics user storien på baggrund af, at der var enkelte ting i iteration 1 og 2, der ville kræve ekstra arbejde, som der ellers var blevet bevæget sig videre fra, og disse ting tog højere prioritet end ændringer, der ville være meget tidskrævende samtidigt med udelukkende at være visuelle forbedringer. Senere i projektet skete det samme med Kryptering user storien, da der nær projektets afslutning ikke var nok tid til at begynde på noget nyt.

Ydermere var der i projektets forløb enkelte user stories, der viste sig at tage betydeligt længere tid end påkrævet, hvilket også forklarer det hop opad, der ses nær starten af burn down chartet. Det ses også på burn down chartet at fremgangen ikke har været lige så glidende som forventet ud over det enkelte spring opad. Dette skyldes, at de user stories, der blev lavet og sporet i project boardet formentlig burde have været delt mere op. Dette medførte, at der var store mellemrum mellem færdiggørelsen af en user story og dermed fremgang på burn down chartet, hvilket lagde op til en fejlagtig afbildning af arbejdets fremskridt sammenlignet med, hvis user stories havde været mere findelt.

Grupperoller

Projektstyringen og planlægningen, der satte rammen for projektet, betragtede også grupperoller. Grupperollerne har ikke fungeret som en hård opdeling af, hvem der laver hvad, men er mere en beskrivning af hvilke opgaver, hvert medlem har haft fokus på at tage sig af for at sikre, at alle arbejder på det, som de føler sig mest sikre på.

Frederik

Min del af arbejdet har hovedsageligt været at programmere nogen af de systemer der ligger i baggrunden af programmet. Gamestate systemet til håndtering af skærme samt klasserne der håndterer de forskellige filtyper og konvertering af dem, samt databasens design og implementering.

Jonas

For bedst at udnytte mine styrker har mit fokus hovedsageligt været design af UI og håndtering af projektstyring, da jeg har et øje for grafik og et godt overblik, når det gælder administration. I selve kodning af programmet har mit fokus været på menuen for cosmetics og karakteren, og hvordan dette interagerer med resten af programmet. Herudover har jeg været inde over opstilling af grafik.

Viktor

Min rolle i gruppen har hovedsageligt været at programmere. I forhold til det har mit fokus været i at lave de forskellige skærme der skal kommunikere med de dele Frederik har lavet, for eksempel skærmen hvor læreren kan oprette opgavesættene. Derudover har jeg også gjort små forskellige jobs, når der har manglet nogle små dele på andres dele af programmet. Da jeg ikke har nogle specielle områder indenfor programmering jeg er meget stærke i, sammenlignet med andre områder, har dette været den bedste udnyttelse af min arbejdskraft.

Produkt

Denne projektstyring gav anledning til kodning og fremstilling af det endelige produkt, der er ved navn “Kool Skool RPG” - et navn, der ligger op til brug i skolen, samtidigt med at det er lidt sjovt og anderledes.

Produktion

Som beskrevet i afsnittet “Ændringer i planlægning” under kapitlet “Projektstyring”, var der enkelte afvigelser fra planen lagt fra starten af, men ellers har produktionen af produktet gået forholdsvis ukompliceret. Herunder ses et overblik af, hvordan de individuelle dele af programmet blev udarbejdet. Et mere detaljeret kig på, hvordan selve koden fungerer, ses under kapitlet “Dokumentation”.

Digital illustration

Programmets brugergrænseflade blev først designet med papir og blyant som ses tilbage på figur 2. Dette design gennemgik flere omgange af ændringer ud fra forskellige gruppemedlemmers ønsker til, hvordan interaktion med programmet skal fungere. Ligeledes, blev programmets brugergrænseflade også ændret yderligere som det blev udviklet, da bestemte forhold og størrelser er lettere at give korrekte proportioner, når det er på en skærm frem for papir.

Grafikken i programmet er lavet ved at tegne geometriske former i processing. Der er dog enkelte undtagelser til dette, netop cosmetics og tekstfelter. Til udformning af tekstfelterne er librariet controlP5, som tidligere nævnt, blevet udnyttet. Til ikonerne og teksturer af cosmetics er der blevet brugt licensafgiftsfrie stock-billeder med tilladelse. Disse er blevet redigeret til i open-source billedredigeringsprogrammet GIMP.

Script

Udformningen af scriptet har været ganske ligetil. Der er blevet gjort brug af de redskaber og metoder, der ligger i processing, og tilhørende dokumentation på processings hjemmeside. Løbende er der udført “blackbox-” og “whiteboxtest” under programmets udvikling, hvor der blev kigget på forventet output i forhold til input, og hvor der er kigger på programmets logik, for at kontrollere, at programmet løber som forventet. Ved tidspunkter, hvor dette har fundet fejl, er der anvendt breakpoints, println() metoden og andre debugger redskaber til at diagnosticere problemet og løse fejlen. Til tider under udviklingen var der funktionalitet vi

skulle bruge men som ville være for besværlig at kode selv. Til de problemer brugte vi eksterne biblioteker der havde de rette funktionaliteter, såsom BezierSQLib biblioteket, vi brugte til at kommunikere med SQLite databasen.

Hen mod slutningen af udviklingen af programmet var der enkelte kompromiser, der var nødt til at blive lavet. Som allerede nævnt blev den user story ved navn "Cool Graphics" fravalgt. Ydermere var der små detaljer, såsom at undlade at gemme de cosmetics, som en bruger havde købt, men ikke havde på, og i stedet blot konvertere dem til mønter, som blev gemt i programmets SQLite database, idet arrayList datatypen er besværlig at gemme i en SQLite database.

Endeligt produkt

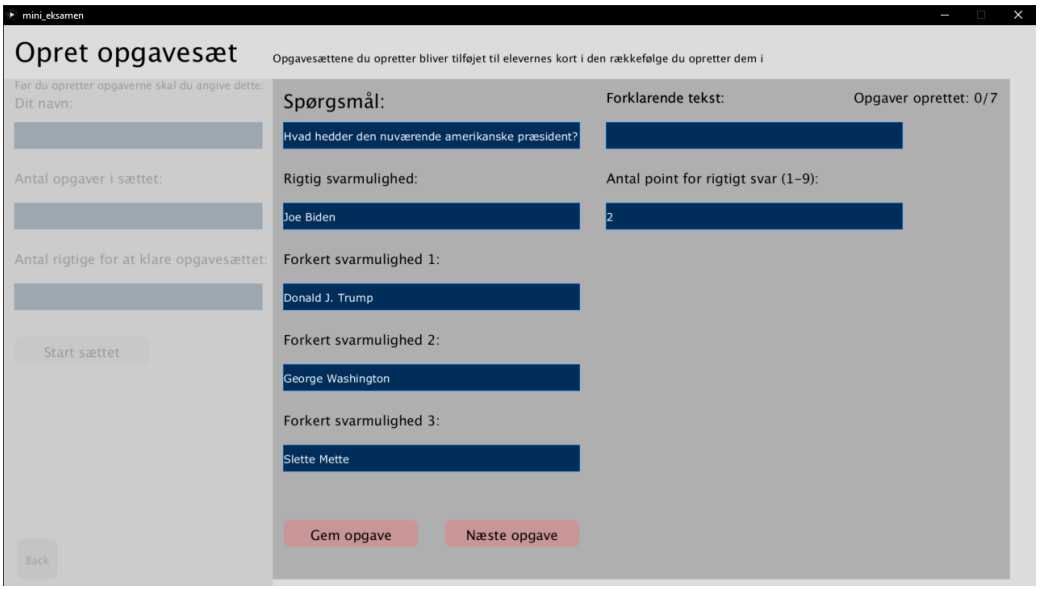
Sammenfaldet af produktionens elementer gav anledning til et færdigt produkt. En brugers rejse gennem programmet gennemgås herunder.

Når en bruger åbner programmet mødes de af en velkomstskærm, der giver mulighed for at vælge, om brugeren er en lærer eller elev.

Figur 5: Skærmbillede af velkomstskærm

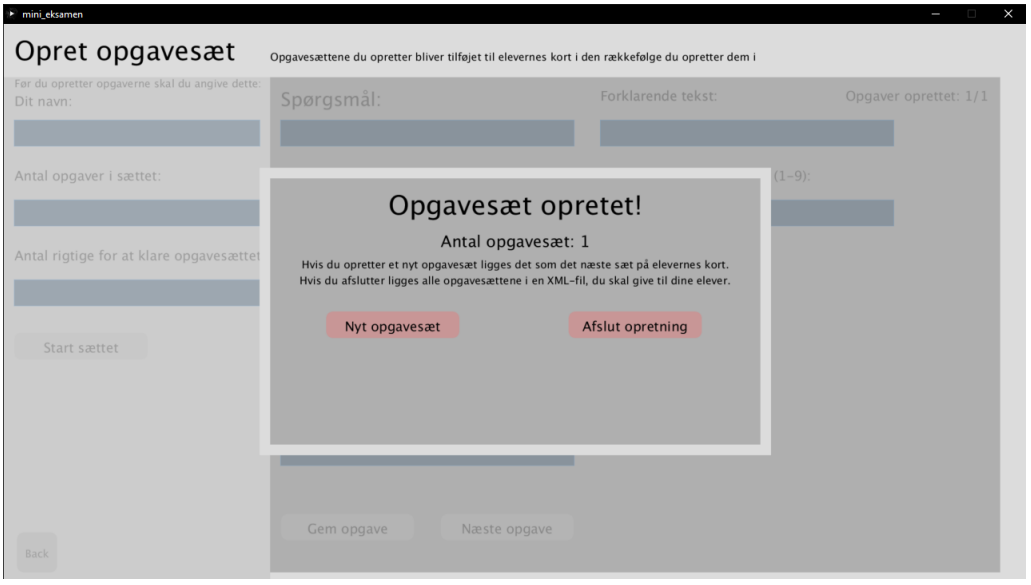
Lærer

Vælger man, at man er en lærer, kommer man videre til siden til at oprette opgavesæt. Her skal man angive sit navn, antallet af opgaver og det påkrævede mængde korrekte svar for at have klaret opgavesættet. Herfra kan man trykke "start sættet", hvorefter man kan begynde at fylde spørgsmål, rigtigt svar, forkerte svar, forklarende tekst og antal point, som spørgsmålet giver, ud.



Figur 6: Skærmbillede af brugergrænseflade til at oprette opgavesæt

Når opgavesættet er færdiggjort mødes læreren af et dialogvindue, som set herunder på figur 7, der giver mulighed for at starte et nyt opgavesæt eller at afslutte opgaveopretningen. Vælges det at lave et nyt opgavesæt fortsættes der, indtil læreren vælger at afslutte opretning af opgave.

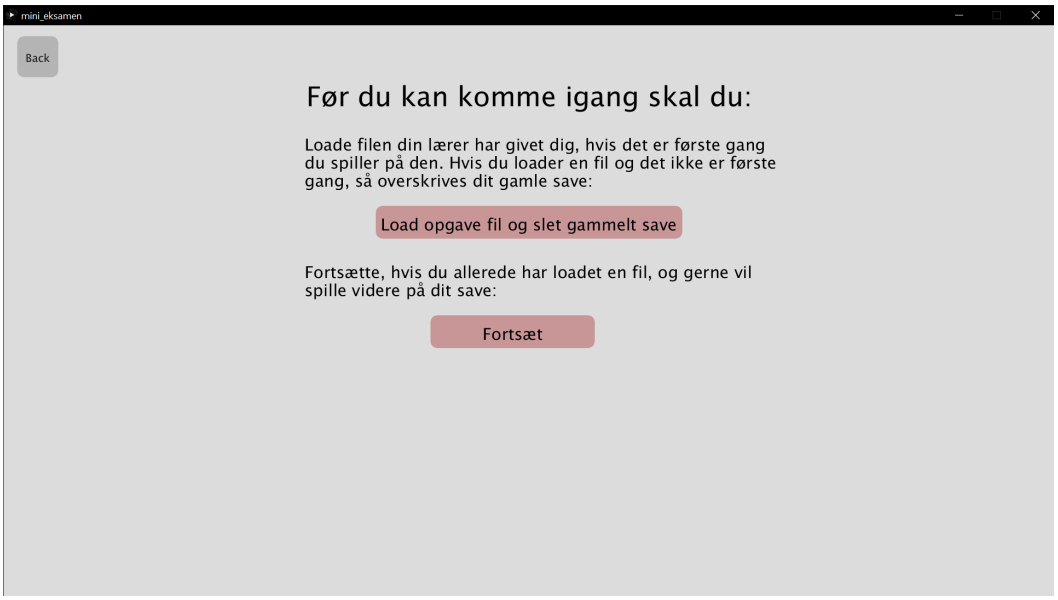


Figur 7: Skærmbillede af dialogvindue

Den XML fil, der repræsenterer opgavesættet oprettes i samme folder som programfilen, og det er da op til læreren at dele den ud til eleven. Dette ville oftest ske ved brug af en besked på en platform såsom elevIntra, Lectio, andre læringsplatforme, eller ved en mere lavpraktisk maner, hvor filen kan deles ud ved USB drive.

Elev

Vælges elev i velkomstskærmen mødes brugeren af endnu en landingpage, som set herunder på figur 8, hvor der skal vælges, om man vil bruge et gammelt save af spillet eller oprette et nyt.

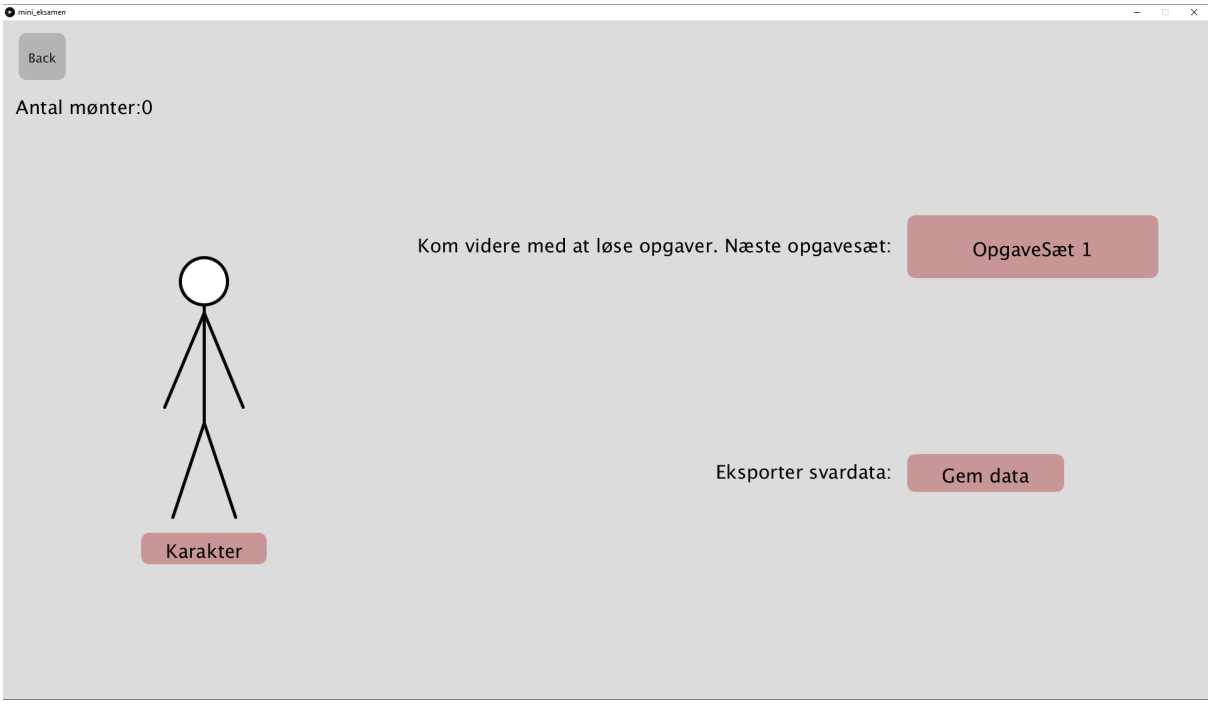


Figur 8: Skærmbillede af load-fil dialog

Vælges "Load opgave fil og slet gammelt save" åbnes et windows dialogvindue, hvor man kan vælge en fil. Trykkes fortsæt kommer eleven videre, hvis de tidligere har haft et save loadet.

Loades et nyt save, angiver eleven sit navn i en ny dialog, som set på figur 9.

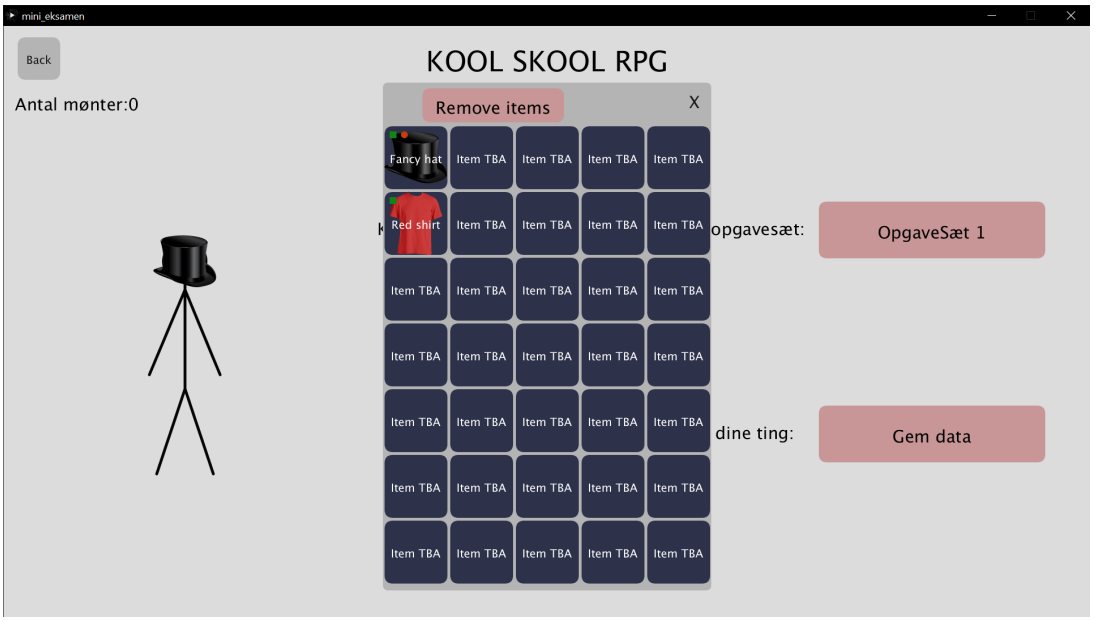
Figur 9: Skærmbillede af dialog, hvor eleven skal angive sit navn



Figur 10: Skærmbillede af elevens hovedskærm

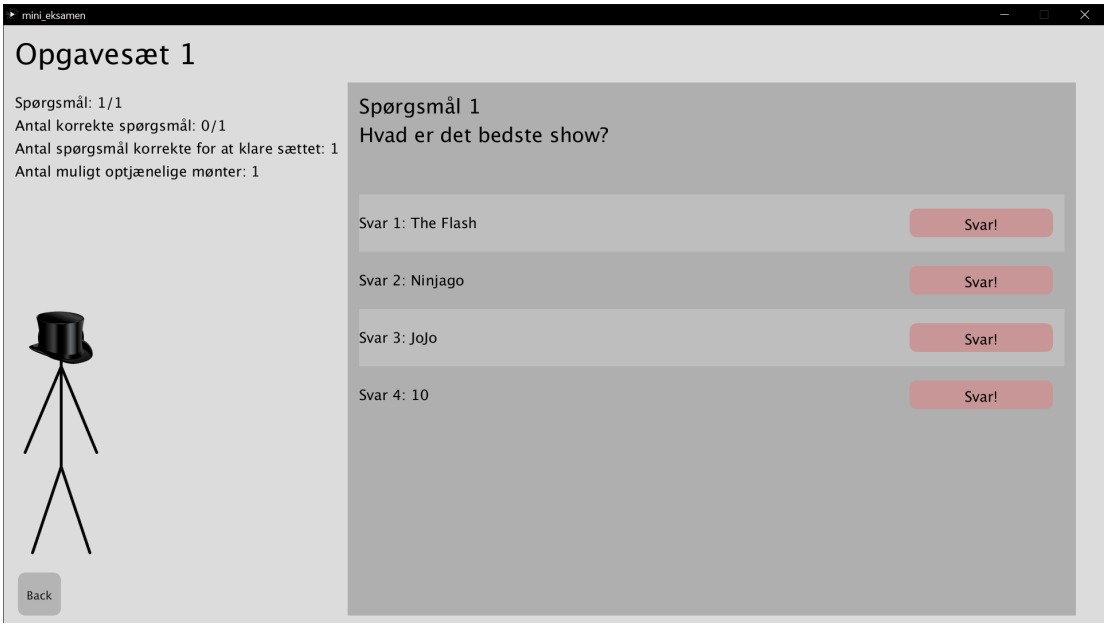
Om eleven er kommet hertil ved at loade et nyt save eller ved at fortsætte, ankommer eleven til en form for home page, som ses ovenfor på figur 10. Her bydes eleven velkommen af deres

karakter, og har herfra en del forskellige muligheder. Elven kan trykke på "Karakter" for at åbne en menu, der lader dem købe og tage cosmetics på, som ses nedenfor på figur 11.



Figur 11: Skærmbillede af menu til at vælge cosmetics. Her er begge cosmetics allerede købt, hvormed pris ikke vises.

Vælges "Gem data" gemmes fremgang i en .csv fil i programfolderen. Trykkes der på et opgavesæt sendes eleven videre til opgavebesvarelse, som set på figur 12 herunder.



Figur 12: Skærmbillede af opgavebesvarelse som elev

Mens eleven besvarer opgaver vil karakteren kommentere på lignende vis af den velkomst, den giver, når man kommer ind på elevens hovedskærm.

Når opgavesættet er besvaret mødes eleven af endnu et dialogvindue, som set på figur 13, der informerer om, hvordan man har klaret det, og hvor mange mønter man har fået.

Figur 13: Skærmbillede af dialogvindue for færdiggjort opgavesæt

Dokumentation

Generel opbygning

Med et overblik dannet over, hvordan en bruger navigerer igennem programmet, kan der nu gåes i dybde med den tekniske opbygning og dokumentation for koden.

Programmet er udviklet i processing 3.5.4, som gør brug af Java. Der er gjort brug af libraries control P5 til enkelte grafiske elementer og SQLite til database.

GameState-logik

Til at holde styr på de mange forskellige skærme i programmet er en `GameState`-logik. Hver af de forskellige skærme er en subklasse af parentklassen `GameState`. Fordelen ved dette er at de forskellige skærme så kan tilføjes til en HashMap, hvor de forskellige skærme så refereres til med en String. Metoderne til at tilføje objekter af de forskellige skærm-klasser, samt til at skifte i mellem skærmene er i `GameStateManager`-klassen. En anden fordel ved denne logik er at ting, der skal være tilfældes for alle skærme, kan skrives i `GameState` klassen, der jo er en parent-klasse til skærmene. Her sikres det blandt andet at alle skærme kan nå databasen.

Knapper

Et system af en parent-klasse og forskellige subklasser er også brugt til knapperne i programmet. Klassen `BaseButton` er parent-klassen, hvor logikken bag knappen ligger. Her tjekkes det om musens position ligger indenfor knappens areal gennem et if-statement, og om musens venstre knap er clicked. Hvis den er bliver booleanen `clicked` true, og den kan returneres i de forskellige klasser, hvor der er objekter af en af knap-klasserne, gennem metoden `isClicked`. De forskellige subklasser af `BaseButton` er blandt andet `ButtonWPause`, der i dens run-funktion skal passes en boolean `pause`, og hvis den er true er det kun `Draw` metoden der kaldes, og ikke `Update`, så knappen stadig tegnes men der ikke sker noget hvis den klikkes på. Dette skal blandt andet bruges i `OpgaveScreen` klassen, hvor man kun må kunne gå i gang med at oprette opgaver, efter man har givet information om hele opgavesættet.

Læreren

XML opgavesæt

Klassen `XMLHandler` stod for at skrive og læse de relevante xml filer og konvertere dataen i dem til noget resten af programmet kunne forstå. Disse xml filer var opgavesættene som lærerne vil oprette og eleverne ville indlæse i programmet. Xml blev valgt til dette formål da processing allerede havde et xml bibliotek bygget ind i der kunne behandle xml strukturen. De to store metoder i klassen er `WriteToXML` og `ReadFromXML`. Begge klasser konverterer mellem tredimensionelle string arrays og xml. `WriteToXML` er ret simpel da den er to for-løkker inde i hindanden der bruger `.addChild` og `.setContent` til at skrive informationen fra arrayet til et xml objekt. `ReadFromXML` er mere kompliceret på grund af den måde strukturen af xml objektet skal dekonstrueres. Her er der gjort brug af flere for-løkker inde i hinanden til at læse xml objektet. `.getChildren` og `.getContent` metoderne bruges til at få henholdsvis alle under xml objekter og selve dataen af et xml objekt. Disse metoder bruges derfor til at gå igennem hele xml objektet systematisk for at opbygge det tredimensionelle string array.

Opgavesætlaver

`OpgaveScreen` klassen, som er en subklasse af `GameState` klassen, bruges til at lave lærerens side af programmet hvor de kan oprette opgaver. Klassen består hovedsageligt af en `update` og `draw` metode, og udnytter `TextField` klassen, til at give læreren et sted hvor de kan inputte deres spørgsmål og andet info. Denne klasse beskrives senere. I klassen er det todimensionelle string-array `opgave`, hvor de forskellige input læreren giver opbevares. På den første plads, altså `[0][n]`, hvor n er en int fra 0-2, opbevares generel info om programmet, såsom antallet af spørgsmål. Disse strings med info indsættes når der klikkes på `startSaetKnap`. De resterende pladser i arrayet, `[N][n]`, bruges til spørgsmål, hvor spørgsmål 1 er ved N = 1, og så videre. For at sikre at alle strings læreren giver i tekstfelterne er som de skal være, for eksempel at der ikke kan være flere end 99 spørgsmål i et opgavesæt, bruges forskellige metoder i tekstfelt klassen, hvor forskellige variabler såsom `canSave` sikre i de forskellige if-statements, at der ikke indsættes noget ikke korrekt i arrayet. Når længden af arrayet bliver lig antallet af spørgsmål læreren har angivet der skal være i opgavesættets kaldes `update` og `draw` i `OpgaveSaetMenu` klassen, hvor der når update-metoden kaldes, gives det todimensionelle array med opgaverne læreren har oprettet. I denne klasse tilføjes det todimensionelle array til en ArrayList `OpgaveSaetAL`. Læreren kan så vælge at oprette endnu et opgavesæt, hvor de forskellige pladser i `opgave`-arrayet vil sættes lig null. Denne proces gentages indtil læreren har oprettet de opgavesæt de vil. Derefter vil arraylisten med det todimensionelle array gennem et for-loop, der looper for hver af pladserne i arraylisten, konverteres til et tredimensionelt array, hvor den første plads beskriver opgavesæt n. Metoden

`writeToXml` i `xmlHandler` klassen kaldes, med det tredimensionelle array, og gennem `skiftGameState` metoden skiftes tilbage til startskærmen.

Tekstfelt

Librariet ControlP5 er blevet brugt til tekstfelter i programmet, da der blev vurderet i planlægningen af projektet at der ikke var tid til at lave vores egne tekstfelter. I `TextField` klassen udnyttes forskellige dele af librariet, såsom `setText` metoden til at fjerne tekst fra tekstboksene, når metoden `removeText` kaldes fra `opgaveScreen` klassen.

Eleven

Karakteren

`Character` klassen beskriver elevens karakter. Den er opbygget af en `update` metode og en række forskellige metoder til at tegne grafik. De essentielle egenskaber ved en spiller karakter er dens muligheder for customization og at tale til spilleren. Hvilkene cosmetics en spiller har tilgængelige opbevares i SQL databasen, som gennemgås længere nede. Karakterens dialog opbevares i et array af strings, og det bestemmes hvorvidt karakteren skal snakke ved den boolske public variabel `speakLine`, der ligger i `mainLogic` klassen, og som `update` metoden i `Character` kaldes til i `mainLogic`, hvor `Character` instantieret. Når `speak` sættes til true bliver variablen `speakTimeFrameStart` sat til at være lig return-værdien af metoden `millis`, der returnerer hvor mange millisekunder programmet har forløbet. I metoden, der tegner selve karakterens dialog er da et if-statement, der tjekker om `millis` returnerer en værdi der er højere end `speakTimeFrameStart` plus en anden integer, `speakTimeMillis`, der beskriver hvor længe karakteren skal snakke, og hvis dette if-statement er sandt, bliver `speakLine` sat til false, hvormed karakteren stopper med at snakke, med mindre der gives signalet til det igen. Et lignende resultat kunne selvfølgelig også have været opnået ved at lave en metode, der har funktionalitet magen til `speakLine`.

Item menu

Menuen til valg af cosmetics beskrives gennem to klasser, `CustomMenu` og `ItemButton`. Objekter af `ItemButton` instantieres i `CustomMenu`, der instantieres i `MapScreen` klassen, der er nedarvet fra `GameState`.

`Update` og `Draw` metoden i `CustomMenu` kaldes ligeledes også i `MapScreen`. Disse metoder gives en boolean `Menu`, der er true, hvis menuen skal vises. Den sættes true når "Karakter" knappen, som er et objekt af `Button`, der oprettes i `MapScreen`, trykkes, og den sættes false, når "ExitButton" knappen, der også er et objekt af `Button`, der oprettes i `MapScreen`, trykkes.

I `Update` køres "removeItemsButton", der vendes tilbage til. I første loop af programmet vil et if-statement, der sætter sin egen konditioner false, så den kun bliver kørt en enkelt gang, blive kørt. Her gennemlæses filer i programmets data-folder ved brug af metoder, der importeres fra `java.io.File`. De af disse filer, der har .png filtypen tilføjes til arraylisten `itemTextures`. Herefter kaldes metoden `initializeItemsBasic`, der opretter objekter af `ItemButton`. Her laves beregninger ved row/column metoden for at give disse objekter de rette koordinater, når de konstrueres. Herudover skælnes der mellem knapper, der har et cosmetic, og knapper, der pt. ikke har et item, men stadig skal tegnes. Ydermere bruges en matrice af to for-loops til at sammenligne arraylisterne `itemTextures` og `ownedItems`, således at knapperne har den rette information om, hvorvidt brugeren allerede har købt genstanden eller ej.

I `CustomMenu` oprettes som sagt også et objekt af `Button`, der når klikket kalder metoden `removeItems`. Denne metode itererer igennem de oprettede objekter af `ItemButton`, og sætter variablen `wearing` til at være false, og kalder herefter også en metode i `Character`, der fjerner items fra karakteren.

Opgavekort

Opgavekort, eller `MapScreen`, er klassen der står for at sende eleven ind i det rigtige opgavesæt der skal besvares. Der er en knap som har tallet på det næste opgavesæt der står i række på sig. Den knap sender eleven ind i `QuestionScreen`, og giver skærmen det korrekte opgavesæt med. For at finde ud af hvad det næste opgavesæt er tjekkes der i databasen for hvad det højeste opgavesæts id er. Klassen har det tredimensionelle string array og tager det korrekte opgavesæt idet skærmen skifter til questionscreen, f.eks. Map[3]. Klassen sørger også for at eksportere svardata til en .csv fil.

Når excel loader en .csv fil, ses der ikke på filens kodning, den læses bare som var dens kodning ANSI. Og da processing kun skriver til filer med UTF-8 kodning, bliver æøå læst forkert af excel. For at komme uden om dette tog vi de rå bytes af første del af csv filen, (som altid er den samme og som har æ, ø og å), med ANSI kodning og skrev dem til csv filen. Derefter brugte vi javas FileWriter til at skrive resten af dataen til csv filen via nogle for-løkker.

Opgave-besvarer

Når spørgsmåls-knappen klikkes i `MapScreen` klassen, kaldes den specielle `SkiftGameState` metode, `SkiftGameStateQuestion`, der passer videre det todimensionelle array med spørgsmålene, som beskrevet tidligere, samt nummeret på opgavesættet. I denne klasse oprettes fire knapper og tekstfelter, for hver af svarmulighederne. Grundet formatet hvorved læreren skriver spørgsmålene ind, vil det rigtige svar altid stå øverst. For at forhindre dette oprettes der en IntList, med fire pladser, med tal fra 1-4. Det rigtige svars position afhænger af første plads på IntListen, den første forkerte svars position afhænger af den anden plads, og så videre. Gennem den indbyggede metode `shuffle` byttes tallene på IntListen rundt, hver gang en af svar-knapperne trykkes på, hvorved det rigtige svar vil have en ny plads i rækkefølgen af svarmulighederne hvert spørgsmål. En int `antalKorrekte` stiger med en, hver gang metoden `isClicked` returnerer true for knappen til det rigtige spørgsmål. Når opgavenummeret eleven er på bliver lig antallet af opgaver i sættet, tjekkes der i et if-statement om antallet af korrekte er lig det antal man skulle have, der fås fra det todimensionelle array. Hvis det er gemmes dataen i elevens SQLite fil, `progress`. Dette bliver gjort gennem en `INSERT` kommando. Der returneres til `MapScreen`, gennem de tidligere nævnte metoder.

Database

Databasen der bruges er en SQLite database. I den database er der to tabeller. Tabellen 'info' bruges til opbevaring af tilfældig data der gerne skal kunne gemmes, som navnet på eleven og antallet af mønter eleven har. Den anden tabel 'progress' bruges til at opbevare data om hvert besvarede opgavesæt, som kan eksporteres til en .csv fil senere, så læreren eller eleven kan behandle dataen og føre statistik i et program som excel. Til kommunikation bruges BezierSQLib biblioteket. Selve SQLite filen ligger i en folder under appdata som programmet selv laver ved startup, det sikrer at vi altid har databasen et kendt sted selvom programmet flyttes rundt. Når der laves et nyt save bliver begge tables slettet og genoprettet for at få en frisk start. Der bliver lavet et enkelt SQLite objekt instantieret i programmet, som alle klasserne derefter får en reference til. På den måde undgår vi at der kommer for mange åbne forbindelser til databasen hvilket gør den utilgængelig.

Fil loader

Fil loader klassen, eller `LoadFileScreen`, som den hedder i programmet, nedarver fra `gameState`. Her kan eleven vælge at loade en en xml fil, opgavesættet, eller fortsætte hvis der

allerede eksisterer et save. Når et opgavesæt loades kopieres filen til appdata folderen som SQLite filen også er i, så er vi sikret at vi altid har opgavesættet. Hvis der trykkes fortsæt tjekkes der om der er en xml fil i appdata, og hvis der er, så loades den og man bliver sendt videre til kortet.

Kvalitetssikring

For at sikre kvalitet i produktet, er der som tidligere beskrevet foretaget tests, efter anden iteration, og da programmet var færdiggjort, altså efter tredje iteration. Testpersonerne har været bekendte af de forskellige gruppemedlemmer, som har været en del af målgruppen, altså folkeskoleelever. Der blev i starten af projektet, som tidligere beskrevet, besluttet at det var vigtigt at prioriterer kvalitative og detaljerede interviews, over mere kvantitative interviews. Programmet har en størrelse der gør at det tager lidt tid at forklare, og for testpersonerne at prøve, det hele. Dette ville gøre et forsøg, på at lave en kvantitativ undersøgelse, dårligt, da de spurgte personer ikke ville få et helt indtryk af programmet. Dette ville føre til upræcise resultater. Af denne grund er den kvalitative form af testene valgt. Testene der blev foretaget endte med at være strukturerede som semistrukturerede interviews, med observationer. Først blev testpersonerne bedt om at klare forskellige opgaver, og hvor vellykket det var blev observeret. Opgaverne ses herunder:

- Find lærer-delen af programmet og opret et kort med opgavesæt, hvor der er 2 opgavesæt hver med 2 opgaver.
- Find elev-delen af programmet og load filen med dit opgavesæt ind.
- Svar på det første opgavesæt.
- Køb en cosmetic til din karakter.
- Eksporter dine svardata.

Efter observationerne af forsøgslederen var blevet gjort, blev forsøgspersonerne interviewet, og det blev spurgt hvor nemt de selv syntes opgaverne var. Derudover blev generelle spørgsmål til programmets brugerinterface også spurgt. Herunder beskrives resultaterne.

Test efter 2. iteration

Efter denne test var det klart at flere instruktioner skulle ses på skærmen. Specielt i de to første opgaver havde forsøgspersonerne svært ved at finde rundt i programmet, og det tog længere end optimal tid for dem at forstå for eksempel hvordan de skulle oprette en opgave. Dette udtrykte de fleste af forsøgspersonerne også. For at forbedre på dette blev der tilføjet flere kommentare i porgrammet, det giver noget information. Et eksempel kunne være i OpgaveScreen-klassen, hvor der nu står “Opgavesættene du opretter tilføjes til elevernes kort i den rækkefølge du opretter dem i.” Der var også nogle forskellige bugs i programmet der blev opdaget gennem disse tests, for eksempel at der var besvær med at loade et nyt kort over opgavesæt ind, efter man allerede havde gjort det en gang. At fikse disse bugs blev tilføjet som tasks til den 3. iteration.

Udover dette var resultaterne af testene positive. Alle testpersoner mente at det var nemt at loade XML-filen for at få opgavesættet. Derudover udtrykte de fleste testpersoner også, at det at man kan købe nye ting til sin karakter, baseret på hvor godt man klarer sig ville være motiverende i forhold til at få dem til gerne at ville løse opgaverne. EN observation der blev gjort var at resultaterne af testene var mere positive hos yngre testpersoner. Det var ikke muligt at finde testpersoner med samme alder som en i indskolingen, men testpersonerne i mellemskolingen nævnte i højere grad at karakteren ville motiverer dem, sammenlignet med elever fra udskolingen.

Test efter 3. iteration

Testen der blev gjort efter programmets færdiggørsel var magen til den test efter 2. iteration. originalt var det planlagt at bruge nye testpersoner til denne test, da resultaterne bedre ville kunne sammenlignes, da begge tests ville blive gjort med personer der så programmet for første gang. Dog endte dette med ikke at være muligt, og testpersonerne blev bare spurgt ekstra godt ind til om de syntes det var nemmere - og det gjorde de. De tilføjede kommentare hjalp meget med programmets brugervenlighed. Ellers forløb testen ligesom den tidligere test, med positive resultater. Nogle af de ting der ikke blev nået i programmet, som tidligere er beskrevet, blev dog pointeret af en enkelt testperson.

Diskussion

Vurdering af produkt og produktion

Som nævnt under afsnittet "Ændringer i planlægning" under kapitlet "Projektstyring" og vendt tilbage til i afsnittet "Produktion" under kapitlet "Produkt", var der enkelte elementer af produktet, hvor det var nødvendigt at skære små ting fra. På trods af dette vurderer gruppen selv, at programmet opfylder projektets krav og gruppens egne forventninger, men dette betyder ikke, at der ikke er forbedringer, som kan laves.

Helt konkret lød formålet med programmet, der blev givet som en del af opgaveformuleringen, at *"Der skal laves et it-system, som kan hjælpe en underviser med at se om eleverne har lært et pensum hen over f.eks 3 måneder"*. Da dette er programmets formål, er det især vigtigt at reflektere over, hvorvidt dette krav opfyldes.

En lærer opretter selv et opgavesæt, hvormed læreren til samme grad som ved en normal prøve eller aflevering kan bestemme sværhedsgrad og hvad eleverne skal testes i. Dette opgavesæt gives da til eleven, som besvarer det, og gemmer sin besvarelse som en .csv, som eleven kan give tilbage til læreren, hvormed læreren kan se, hvad eleven har besvaret i et excel ark. På denne måde kan læreren som minimum benytte programmet til at teste eleven til samme grad som ved normale opgaver, men der er et par nøgle-features, der giver læreren mulighed for at få mere ud af programmet end blot dette. Først og fremmest motiveres eleven yderligere takket være de cosmetics, som eleven kan arbejde sig hen imod, hvilket kan hjælpe eleven på deres motivation, og ved at styrke elevens engagement for undervisningen, gøres lærerens andre opgaver ud over blot at spore faglig præstation lettere. Som sagt modtager læreren elevens besvarelse som en .csv, som kan åbnes i excel. På denne måde digitaliseres elevens besvarelse automatisk, hvilket sparer læreren en masse arbejde, hvis læreren ønsker at føre mere dybdegående statistik over elevens præstation end blot rigtige opgaver ud af samlet mængde opgaver.

Som sagt var der dog ting, der blev skåret fra undervejs. Helt konkret er der tale om indholdet af iteration 3, kryptering og cool graphics. Dette vurderes dog ikke som værende et problem, da iteration 3 fra starten af blev tiltænkt som en iteration, der indeholder udvidelser, som kunne blive lavet, hvis der viste sig at være ekstra tid. Altså er det ikke et stort tab, at disse ting ikke er blevet lavet.

Udvidelse af program

Ud over selvfølgelig at få lavet indholdet af iteration 3, er der bestemte features, som ville forbedre programmet at tilføje. Herunder menes der, at opgavesættene ville blive krypteret, således at en tech-savy elev ikke ville kunne smugkigge ind i den .XML fil, som læreren giver og på den måde vide, hvilke spørgsmål og svar, der kommer. Herudover ville brugergrænsefladen få en makeover, der skal gøre programmet mere interessant at se på. Karakteren ville få et komplet grafisk remake med original pixel art, frem for en pindefigur, som der sættes .png'er af tøj ind over.

Først og fremmest ville det gøre lærerens arbejde lettere, hvis den .csv fil, som blev eksporteret kom med behandlet data og et statistisk overblik over resultaterne, og ikke blot en gennemgang af elevens svar. Dette ville også give eleven mulighed for bedre at forstå, hvordan det er gået for dem. Yderligere kunne der laves en statistisk analyse inde i programmet der eliminerer nødvendigheden for eksterne programmer og selv at stille grafer op.

Vurdering af test

Kigges der tilbage til starten af produktionen af programmet, ville det have været ønsket at have udført en papirprototypetest, for at sørge for, at den udviklede brugergrænseflade er intuitiv, inden den kodes ind i programmet. Ligeledes ville det også havde været rart at teste programmet på en person, der har den forventet alder og tekniske know-how af en folkeskolelærer, og ikke blot en elev.

Vurdering af gruppearbejde

Gruppearbejdet vurderes som værende ganske velfungerende. Denne gruppe arbejdede sammen i tidligere DDU projekter, og har der fundet en arbejdsrytme og gruppedynamik, som fungerer for alle. Som beskrevet i større detalje under kapitlet "Projektstyring" har arbejdet foregået hjemmefra men med fysiske gruppemøder, og arbejdet er blevet opdelt på en sådan måde, at alle har mulighed for at fokusere på de opgaver, som lægger mest op til deres individuelle styrker, samtidigt med at kunne hente hjælp fra resten af gruppen.

Konklusion

Problemformuleringen for opgaven lød *"Hvordan kan man med en digital løsning skrive et program, der hjælper med at teste en elevs faglige kompetencer?"*, og det givne formål med programmet at *"Der skal laves et it-system, som kan hjælpe en underviser med at se om eleverne har lært et pensum hen over f.eks 3 måneder"*.

Der blev fundet en løsning, der involverer gamification og brug af cosmetics til at motivere eleverne, samtidig med at der blev udviklet et versatilt system, der lod læreren designe sine egne opgaver til at teste eleven, som holdes motiveret af sin karakter og muligheden for at få disse cosmetics.

Kilder

1. Undervisningsministeriet . (2018). *Handlingsplan for teknologi i undervisningen* . https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiKqZLC_970AhWSzaQKHVvcCGEQFnoECBQQAQ&url=https%3A%2F%2Fwww.uvm.dk%2Fhandlingsplan-for-teknologi-i-undervisningen%2F-%2Fmedia%2Ffb6c1b2aef1140598493999566ce034b.ashx&usg=AOvVaw2jV06y_lFMzZSthPYwbcKb
2. Kildall Rysgaard, K. (2018, 7. 12). Dagdrømme og gab: Flere elever keder sig i folkeskolen. *Danmarks Radio, Indland*. https://www.dr.dk/nyheder/indland/dagdroemme-og-gab-flere-elever-keder-sig-i-folkeskolen
3. Cheung, S. Y. & Ng, K. Y. (2021, 31. 3). Application of the Educational Game to Enhance Student Learning. *frontiers in Education: Digital Learning Innovations*. https://doi.org/10.3389/feduc.2021.623793
4. Nørby, J. (2012, 29. 10). *Eksperter: Populære læringsspil er 'tæt på ubrugelige'*. folkeskolen.dk. https://www.folkeskolen.dk/518129/eksperter-populaere-laeringsspil-er-taet-paa-ubrugelige
5. Takahashi, Dean. (2020, 18. 12). *Newzoo: U.S. gamers are in love with skins and in-game cosmetics*. Venturebeat. https://venturebeat.com/2020/12/18/newzoo-u-s-gamers-are-in-love-with-skins-and-in-game-cosmetics/