

Automatisering af studievalg

Valg af 2. prioritet til videregående uddannelse

Christopher C. Jensen, Frederik Stenberg, Jacob Rasmussen,
Lasse D. Skaalum, Mike Johansen, Philip C. Greve og Rune H.
Andersen

Software, A320, 18/12-2019

P1-projektrapport





AALBORG UNIVERSITET
STUDENTERRAPPORT

Første Studieår - Software

Strandvejen 12-14

9000 Aalborg

<http://www.tnb.aau.dk>

Titel:

Valg af 2. prioritet til
videregående uddannelser

Projekt:

P1 projekt

Projektperiode:

Oktober 2019 - December 2019

Projektgruppe:

A320

Deltagere:

Christopher C. Jensen
Frederik Stenberg
Jacob Rasmussen
Lasse D. Skaalum
Mike Johansen
Philip C. Greve
Rune H. Andersen

Hovedvejleder:

Kurt Nørmark

Bivejleder:

Emil Pedersen

Sidetal: 67

Afsluttet: 18. december 2019

Synopsis

Dette projekt beskæftiger sig med, hvordan ansøgeren af en videregående uddannelse kan blive foreslået det bedst mulige 2. prioritetsvalg ud fra deres 1. prioritet i forbindelse med at søge ind på en videregående uddannelse. Flere undersøgelser viser at ansøgers valg af 2. prioritet er mindre velovervejende end 1. prioritetsvalget. Dette er en af grundene til, at frafaldsraten er betydeligt større for ansøgere, som kommer ind på deres 2. prioritet eller lavere. Et C program er udviklet som løsning på dette. På baggrund af brugerens 1. prioritet filtreres og sorteres der ud fra parametre, der i analysen har vist sig at være mest betydningsfulde for studievalget. Alt data om uddannelser i Danmark er fra en CSV-fil fra Uddannelses- og Forskningsministeriet. Desværre har denne fil mangel på data for en række uddannelser. Dette medvirker til at programmet i nogle tilfælde ikke kan outputte en 2. prioritet til brugeren, da det vurderes at dette ikke opfyldt uden tilstrækkelig data.

Rapportens indhold er frit tilgængeligt, men offentliggørelse (med kildeangivelse) må kun ske efter aftale med forfatterne.

Forord

Denne rapport er udarbejdet af gruppe A320 på bacheloruddannelsen i Software på Aalborg Universitet. Rapporten omhandler et P1 projekt lavet i efteråret 2019. Projektet havde basis i automatisering af studievalg på videregående uddannelser. Mere præcist havde projektet for øje at nedbringe frafaldsraten blandt studerende, der bliver optaget på deres 2. prioritets valg. Projektet var et studieprojekt som skulle munde ud i et C program, der kunne løse enten hele eller dele af den omkringliggende problemstilling. Rapporten beskriver den analyseprocess, der ligger til grund for fastlæggelsen af problemet. Herudover indeholder rapporten en gennemgang af problemløsningen, med diskussioner og eftertanker omkring de tekniske overvejelser, der har haft betydning for det endelige C programs udformning. Projektgruppen vil gerne give et stort tak til Kurt Nørmark og Emil Pedersen for deres vejledning og støtte i forbindelse med projektet.

For størst muligt udbytte af rapporten henvises der til disse elektroniske bilag:

- Bilag A: SoTA-tabel
- Bilag B: Spørgeskema-undersøgelse
- Bilag C: Doxygen som er en dokumentation af koden
- Bilag D: Programmets kildekode

Indholdsfortegnelse

Kapitel 1	Indledning	1
Kapitel 2	Problemanalyse	2
2.1	Metode	2
2.1.1	Spørgeskema	2
2.1.2	SoTA table	3
2.1.3	Kvantitativ metode	4
2.2	Konsekvenser af frafald	5
2.2.1	Økonomiske konsekvenser	5
2.2.2	Konsekvenser for individet	6
2.2.3	Delkonklusion	6
2.3	Årsager til afbrydelse af påbegyndt studie	7
2.3.1	Ukvalificeret og uafklaret valg	7
2.3.2	2. prioriteten er ikke ligeså velovervejende som 1. prioriteten.	8
2.3.3	Sociale Miljø	9
2.3.4	Efterfølgende jobmuligheder	10
2.3.5	Boligmuligheder	10
2.3.6	Kvaliteten af uddannelsen	11
2.3.7	Det individuelle ophav	12
2.3.8	Delkonklusion	14
2.4	Frafaldsrater i udlandet	15
2.5	Studievalgprioritering	16
2.6	Parametre der indgår i det gode studievalg	18
2.6.1	Generalisering af det gode studievalg	20
2.6.2	Delkonklusion	20
2.7	Eksisterende værktøjer til studievalg	20
2.8	Metodeovervejelser	21
2.8.1	Fordele og ulemper ved spørgeskema	22
Kapitel 3	Problemformulering	24

Kapitel 4 Problemløsning	25
4.1 Kravspecifikationer	25
4.2 Design	26
4.2.1 Parametre relevante for frafald	26
4.2.2 Proces og programflow	30
4.2.3 Algoritmiske krav	31
4.3 Implementering	32
4.3.1 Kode standard	32
4.3.2 Main	33
4.3.3 Generelle funktioner	34
4.3.4 Datastruktur	38
4.3.5 Dataindlæsning	41
4.3.6 Håndtering af brugerinput	44
4.3.7 Sortering og filtrering	48
4.3.8 Output	51
Kapitel 5 Test	54
5.1 Test af programmet	54
5.2 Alternative fremgangsmåder	56
Kapitel 6 Diskussion	57
6.1 Ét output i programmet	57
6.2 Sortering og filtrering	58
6.3 “Nice-to-haves”	58
Kapitel 7 Perspektivering	60
Kapitel 8 Konklusion	61
8.1 Opsamling	61
8.2 Indfrielse af kravspecifikationer	62
8.3 Opfyldelse af de faglige mål	62
8.4 Forbedringsmuligheder	63
8.5 Implementeringskontekst	63
Litteratur	64
Bilag	
Bilag A: SoTA-tabel	67
Bilag B: Det gode studievalg spørgeskemadata	67
Bilag C: Doxygen	67

Bilag D: Programmets kildekode	67
--	----

1 | Indledning

Denne rapport tager udgangspunkt i studerende som vælger at afbryde deres studie uden at færdiggøre uddannelsen. Dette har vist sig at være et udbredt problem, da det i mange tilfælde har konsekvenser både for samfundet og for individet. Da dette er et stort område, er der skrevet en problemanalyse i rapporten, som afgrænser problemet.

Analysen skaber et udgangspunkt for en problemformulering, som drejer sig om at mindske forskellen mellem frafaldsprocenten for studerende, der er blevet optaget hhv. på deres 1. prioritet og 2. prioritet. Dette blev forsøgt gjort ved at mindske frafaldsraten for 2. prioritets frafalds-studerende.

Løsningen på problemet er et program, som er blevet udviklet i programmeringssproget C. Dette er blevet uddybet i rapportens design- og implementations afsnit, som giver et komplet overblik over, hvordan dette program er blevet udarbejdet.

Da der er mange måder at udarbejde en tilfredsstillende løsning på, har rapporten anvendt data fra offentlige institutioner samt Uddannelse- og Forskningsministeriet, til at kunne skabe sig et klart billede af hvilke elementer, der er vigtige at tage højde for i forbindelse med det gode studievalg. Herudover er der også fundet frem til, hvilke parametre, der kunne have en negativ indflydelse på brugerens studievalg. Afslutningsvis indholder rapporten et diskussionsafsnit, der giver læseren indsigt i overvejelser og valg, der har været afgørende for projektets endelige udformning. Til sidst afrundes rapporten med en konklusion, der samler op og vurderer resultatet af projektet, heriblandt om det færdige program lever op til dets forudgående kravspecifikationer, samt om studieprojektets faglige mål er blevet nået.

2 | Problemanalyse

Med henblik på at udlede en problemformulering, er der i projektet skrevet en problemanalyse. Denne del af rapporten har til formål at analysere problematikker, som omhandler frafald på videregående uddannelser. Derudover diskuteres det, hvordan der træffes et "rigtigt" studievalg. I analysen er der anvendt tidligere rapporter og undersøgelser, som er udført af anerkendte universiteter og undersøgelsesinstitutter som grundlag for saglig argumentation og formidling. I projektet er der udarbejdet en kvantitativ spørgeskemaundersøgelse, med formålet at eftertjekke om konklusionerne fra de anvendte kilder havde lignende tendenser i den pågældende kontekst. Først vil der kort beskrives metodetilgange i projektet.

2.1 Metode

I denne rapport er der blevet udført en spørgeskemaundersøgelse. Udformningen og udarbejdelsen af dette spørgeskema tog udgangspunkt i resultaterne af en undersøgelse fra Danmarks Evalueringsinstitut (2018) (EVA).

Et spørgeskema er lavet ifm. projektet for at sammenligne tendensen med EVAs undersøgelse. Spørgeskemaet består hovedsageligt af kvantitative spørgsmål, men komplementerede med åbne kvalitative spørgsmål på områder, hvor det gav mening.

2.1.1 Spørgeskema

Der er to centrale mål forbundet med spørgeskema-undersøgelsen. Fortrinsvis er det ønsket at undersøge om studerendes valg af 1.- og 2. prioritet er velovervejede. Det sekundære formål var at undersøge, hvilke parametre ansøgeren baserer sit valg af uddannelse på.

I følgende afsnit vil de praktiske forhold bag undersøgelsen først nævnes, hvorefter resultaterne fremlægges og til sidst tolkes den samlede data.

Spørgeskemaet blev sendt ud på det sociale medie Facebook til diverse grupper, med henblik på at opnå flest mulige resultater. Derved blev der ikke taget højde ift. til målgruppe under distribution, da der efterfølgende blev sorteret ud i resultaterne, ud fra hvad der specifikt ville udledes fra datasættet.

Spørgeskemaet tager udgangspunkt i en undersøgelse fra Danmarks Evalueringsinstitut (2018). Det er forsøgt at gøre spørgsmålene så simple som muligt, og derfor er der benyttet korte og simple spørgsmål, hvor svarpersonen kunne vælge ud fra standardiserede svarmuligheder. Der er i nogle sammenhænge brugt spørgsmål med parametervægtning, hvor Likert-skalaen, (en skala fra 1 til 5), er benyttet til at vægte parametrene (Olsen, 2006). Der er ikke indført en reverse-skala, da det var vurderet, at dette ville skabe mere forvirring hos svarpersonen, end det ville gavne. (Aarhus Universitet, 2019)

Svarpersonerne blev indledningsvis bedt om at tage udgangspunkt i deres tanker, fra da de søgte ind på en videregående uddannelse for første gang. Spørgeskemaet indeholder mange spørgsmål, hvor svarpersonen bliver bedt om at tage udgangspunkt i de tanker man havde, da man foretog studievalget i sin tid. Dette resulterede i mange faktuel-episodiske spørgsmål i spørgeskemaet. Foreliggende var en nødvendighed, men også en udfordring, da svar kunne variere ud fra den individuelle hukommelse. Spørgeskemaet har derudover mange faktuel-generelle spørgsmål, hvilket for eksempel blev benyttet i et spørgsmål omkring svarpersonens alder. Der blev benyttet så mange af disse som muligt, for at få spørgeskemaet til at være let og overskueligt. Kundskabsspørgsmål og holdningsspørgsmål var der forsøgt at undgå at inkludere, da det ikke er relevant for vores spørgeskema, at have holdninger indblandet. Dog vil holdning oftest påvirke besvarelsen til en vis grad. (Olsen, 2006)

Spørgeskemaet blev udarbejdet som et primært lukket spørgeskema. Der er derfor brugt lukkede og delvist lukkede spørgsmål. Da spørgsmål med åben fortolkning og egne frie svar er svære at bearbejde og kan skabe langt større usikkerhed i sammenhængen mellem svarene. Derudover blev et spørgeskema benyttet, da det var den hurtigste og simpleste måde at indsamle empiri og data med et minimalt tidsforbrug. Som nævnt er et spørgeskema generelt meget lukket, (Olsen, 2006) hvilket rent metodisk komplementerede undersøgelsen.

2.1.2 SoTA table

Da en større række kilder kan være uoverskuelige at sammenligne, har denne rapport benyttet sig af, at lave et State of the Art (SOTA) tabel. Formålet med dette var, at danne et overblik over flere kilders data, undersøgelsesområder og resultater. Dette gav et godt

overblik over hvilke kilder som har sammenlignelig information og kunne bruges i samme kontekst, og hvilke som eventuelt slet ikke skulle inkluderes. Samtidig gav det mulighed for, at danne sig en forståelse for hvorfor kilder, med hvad ses som de samme betingelser, har opnået forskellige resultater. For eksempel kan man sammenligne mængden af brugere undersøgt, og hvis en af kilderne har en lav mængde undersøgelser, samt et varierende resultat, kunne der spekuleres i om den givne kilde ikke var brugbar. SOTA-tabellen kan findes i Bilag A.

2.1.3 Kvantitativ metode

I rapporten, både i undersøgelsen og i problemanalysen, blev der i høj grad brugt og fundet kvantitativ data, da kvantitativ data kan give det mest generelle billede af, hvordan befolkningen agerer i den givne situation, vælger uddannelse, samt hvilke faktorer der påvirkede det enkelte individs beslutning til at vælge sin uddannelse. (Olsen, 2006) Den kvantitative metode ift. kvalitativ giver et mere generelt billede, såfremt der er mange svarpersoner. Til gengæld giver den også ofte et mindre nuanceret og lukket billede af problemstillingen. (Olsen, 2006)

2.2 Konsekvenser af frafald

Ifølge Danmarks Evalueringsinstitut (2018) er det vigtigt at ens prioriteter er velovervejede, da det ellers kan føre til frafald. Derudover viste undersøgelsen, at der var cirka 50% flere der dropper ud, hvis de kommer ind på deres 2. prioritet, i forhold til dem som kommer ind på deres 1. prioritet. Frafaldet medfører en række konsekvenser, som både rammer individet og samfundet.

2.2.1 Økonomiske konsekvenser

Ifølge tal fra Uddannelses- og forskningsministeriet (2017), blev der i 2016 brugt 18 milliarder kroner på SU kun på videregående uddannelser. Ifølge Ritzau (2018) var der 35% frafald på de videregående uddannelser i 2017, hvilket resulterer i udgifter for flere millioner kroner. Ifølge Uddannelses- og forskningsministeriet (2019) var der i 2019 65.714 nye studerende. Det antages i undersøgelsen at ud af de 65.714 nye studerende i 2019 faldt 35% fra som i 2017, hvilket svarer til 23000 studerende. Her er det vigtigt, at bemærke 81% af frafaldet sker ved studieskift, og 19% forlader helt studiet, som følge af en undersøgelse udført af Uddannelses- og Forskningsministeriet (2018b). Hvis alle disse fik blot en måneds udeboende SU svarer det til 142 millioner DKK,- i spildt SU. Dette tal er højst sandsynligt langt højere, da der er mange der først falder fra senere end den første måned. Det er derimod også vigtigt at tage i betragtning, at ikke alle modtager udeboende SU, eller overhovedet modtager SU, men hvis gennemsnittet udregnes estimerer denne rapport, at tallet er langt højere. Igen understreges det, at denne estimering ikke er konkluderende, men blot har formål at danne en billede af hvilket prisleje problemet kan ligge i.

Institutionerne mister også penge når studerende dropper ud. Dette skyldes taxameterfordelingen, som indebærer, at universiteterne får penge for hver bestået studerende (Uddannelses- og Forskningsministeriet, 2019). Derfor, når der bliver investeret en sum penge i undervisningen og studerende falder fra, vil en del af disse penge gå tabt oveni den sum, som universitetet bliver tildelt efter endt studie. Frafaldet medfører derudover, at kvalificerede ansøgers plads på studiet bliver spildt. Derved går en mulig kandidat til arbejdsmarkedet tabt.

En undersøgelse udført af Finansministeriet viser samtidigt, at individer som ikke har en videregående uddannelse går som ledige længere og har otte år mindre på arbejdsmarkedet (Studenterrådet ved Aarhus universitet, 2000, s. 13). Det medfører at de går længere på efterløn eller kontanthjælp og derved har en større økonomisk last på samfundet. En anden følge af færre år på arbejdsmarkedet og lavere indkomst, er færre penge til Staten i form

af tabte skattepenge.

2.2.2 Konsekvenser for individet

Frafald er ikke nødvendigvis et udtryk for noget negativt, men kan være udtryk for en modningsproces, som indebærer den studerendes indforståelse for, at dette ikke var den rette vej og derfor blot skifter studie. Det kan dog også komme af mentale problemer, familieproblemer mm.

Ifølge Edwards et al. (1990) har familiens uddannelsesniveau indvirkning på, hvordan familien anskuer frafald. Hvis familiens højeste fuldførte uddannelse er mindre end en videregående uddannelse, ses det som normalt ikke at tage en videregående uddannelse. Derfor ses det ikke som noget negativt at droppe ud, men derimod en norm. Bogen konstaterer også, at hvis det derimod er normen i familien, at fuldføre en videregående uddannelse, kan det føre til mindre selvværd og en følelse af ikke at slå til. Dog er det langt de færreste der forlader uddannelsessystemet helt, men når det sker, får det ofte dårlig indvirkning på individet og samfundet. Det kan, som tidligere omtalt, have økonomiske konsekvenser, men udover det udvikler den pågældende person ikke de samme mentale kompetencer, i samme grad som en færdiguddannet studerende. De pågældende kompetencer indebærer intellektuel selvsikkerhed og selvværd, en øget selvstændig tankegang, større tolerance og intellektuelle kompetencer. Når tidligere studerende forlader uddannelsessystemet kan det derudover medføre, at de falder bagud i et hurtigt udviklende samfund og derved får mindre indflydelse på egen fremtid, samt en lavere løn (Edwards et al., 1990).

Det er vigtigt at bemærke, at artiklen fra Studenterrådet ved Aarhus universitet (2000) er fra år 2000, mens flere af deres kilder er fra år 1990. Artiklen er stadig af stor relevans, da samfundet udvikler sig mindst lige så hurtigt og på mange punkter hurtigere, som da artiklen blev skrevet.

Frafald har derfor ikke kun nuværende konsekvenser, men derimod konsekvenser der varer langt ud i fremtiden i form af tabt statsstøtte på samfunds niveau, et fysisk og for nogle mentalt, hårdere liv på individniveau.

2.2.3 Delkonklusion

Da det er tydeliggjort, at der er en bred vifte af konsekvenser ved høje frafaldsrater, både for samfundet og for individet, er det relevant at undersøge hvilke årsager forårsager dette

problem. Det kan spekuleres på forhånd, at der er mange faktorer på spil, hvilket gør det vigtigt at indskrænke, hvilke årsager, som er relevante, samt hvilke årsager, der er mulige at inkludere i projektets problemløsning.

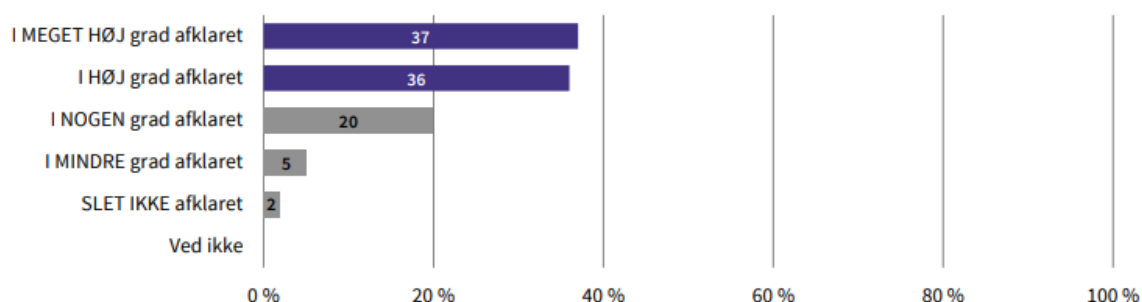
2.3 Årsager til afbrydelse af påbegyndt studie

Grundet betydningen af konsekvenser der opstår som følge af, at studerende afbryder deres studie, er det fordelagtigt at komme frafaldet til livs. Som følge heraf er det væsentligt at undersøge, hvilke potentielle årsager der er skyld i at frafaldet sker. Nogle er parametre som de studerende selv har indflydelse på, men der beskrives også kort en variabel, de studerendes ophav, som den studerende ikke er herre over.

2.3.1 Ukvalificeret og uafklaret valg

At den studerende ikke foretager et kvalificeret studievalg, er en af de frafaldsårsager som flere studier og undersøgelser udleder i deres konklusioner. I en rapport fra det samfundsvidenskabelige fakultet på Københavns Universitet udgivet i 2017, blev der blandt andet undersøgt hvilke årsager der var for, at studerende på bacheloruddannelsen i Økonomi afbrød deres studie. En af hovedårsagerne til frafaldet var, at uddannelsen havde et godt omdømme og at nogle af de studerende havde truffet deres studievalg på baggrund heraf. Ifølge Sørensen og Ibsen (2017) traf de studerende, på bacheloruddannelsen i Antropologi på Københavns Universitet, tilsynladende dette valg uden først grundigt at undersøge om uddannelsen passede til deres evner og interesser. Rapporten afslørede endvidere, at de frafaldne studerende ikke havde sat sig grundigt ind i uddannelsens faglige indhold. Som følge heraf passede den studiegang som de studerende oplevede ikke, med de forventninger som de havde til studiet. En anden undersøgelse, foretaget af Danmarks Evalueringsinstitut (2018), viser lignende tendenser. Her viste det sig at uafklarethed i forhold til studievalget var den største årsag til frafald. Rapporten fra EVA havde til formål at undersøge om der var en sammenhæng mellem hvor afklaret den studerende var med sit studievalg og dennes risiko for frafald. På figur 2.1 ses det at 27% af de studerende højst kategoriserede deres niveau af afklaring som, "I nogen grad". Rapporten konkluderede at studerende der ikke i høj grad var afklarede med deres studievalg, havde større risiko for at falde fra (Danmarks Evalueringsinstitut, 2018). Undersøgelsen fra EVA viste også, at 37% af de studerende som angav de slet ikke var afklarede med deres studievalg, afbrød deres studie.

De studerendes afklarethed på ansøgningstidspunktet mht. deres valg af den uddannelse, de blev optaget på (procent)

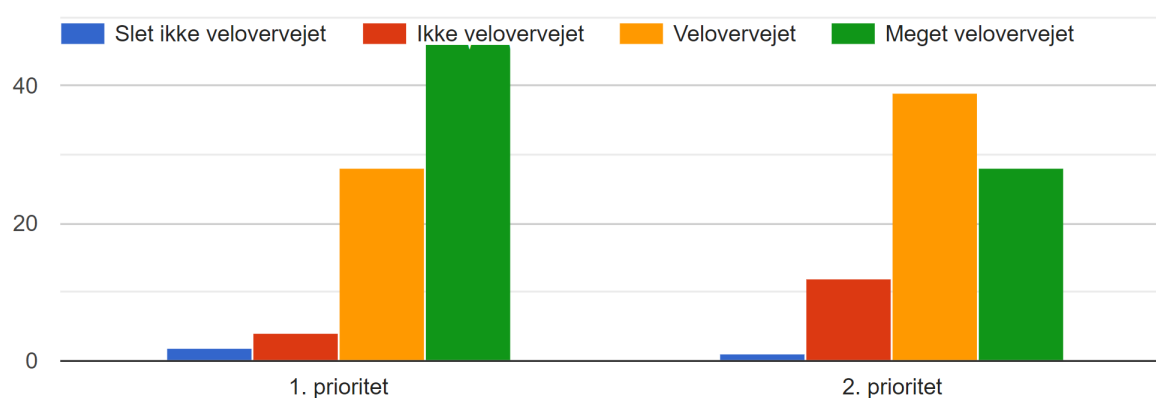


Figur 2.1. (Danmarks Evalueringsinstitut, 2018)

2.3.2 2. prioriteten er ikke ligeså velovervejet som 1. prioriteten.

Undersøgelsen fra Danmarks Evalueringsinstitut (2018) fandt frem til, at valget af 2. prioritet var mindre velovervejet end valget af 1. prioritet, se figur 2.3. På figur 2.2, som stammer fra dette projekts spørgeskemaundersøgelse, ses samme tendens. Herudover viste undersøgelsen, at der var større risiko for tidlig afbrydelse af studiet, hvis den studerende havde fået sin 2. prioritet, se figur 2.4.

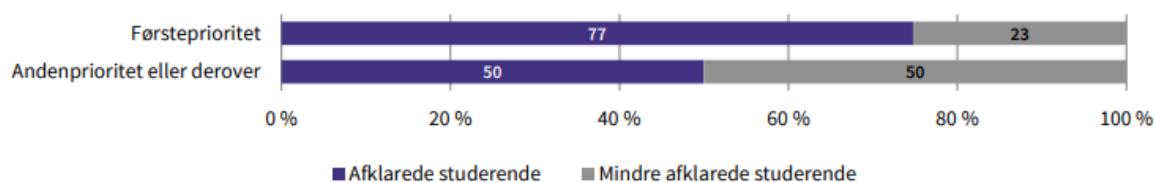
Hvor velovervejet var dit valg af...



Figur 2.2.

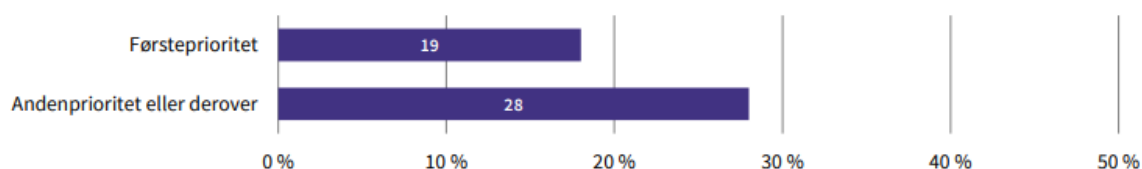
På baggrund heraf, ses det som værende betydningsfuldt for fastholdelsen af de studerende på et givet studie, at de gør sig grundige overvejelser omkring deres valg af studiet således, at de kan træffe to eller flere afklarede studievalg.

Sammenhæng mellem den prioritet af uddannelse, de studerende blev optaget på, og deres grad af afklarethed mht. deres valg af den uddannelse, de blev optaget på (procent)



Figur 2.3. (Danmarks Evalueringsinstitut, 2018)

Førsteårsfrafaldet blandt studerende, der kom ind på deres førsteprioritet, og studerende, der ikke gjorde det (procent)



Figur 2.4. (Danmarks Evalueringsinstitut, 2018)

I denne forbindelse, giver data fra Madsen (2016) et overblik over hvad som eventuelt kunne have effekten som udledte, at den studerende ikke har gjort sig nok overvejelser. For eksempel har studerende med gymnasial uddannelse, som ikke er i det samme område som deres nuværende universitetsuddannelse, større chance for at falde fra for tidligt.

2.3.3 Sociale Miljø

Den studerendes oplevelse af det sociale miljø kan have en betydning for, hvorvidt den studerende dropper ud af sit studie, viser flere kilder. Uddannelsens- og Forskningsministeriet (2018) (UFM), udgav en rapport omhandlende frafald og studieskift. Rapporten er baseret på en spørgeskemaundersøgelse som UFM selv lavede. Spørgeskemaundersøgelsen blev delt med studerende som afbrød deres studie i tidsperioden 1. oktober 2015 til 30. september 2016. Spørgeskemaundersøgelsen modtog 3965 besvarelser. Tal fra rapporten viste at 38 procent af besvarelsene tilkendegav, at det sociale miljø havde indflydelse på deres valg om at afbryde studiet. Den tidligere nævnte rapport, fra Københavns Univer-

sitet, viste at der var tale om et generelt problem angående den frafaldne studerendes inklusion i det sociale miljø (Sørensen og Ibsen, 2017).

Dette problem er ikke isoleret til det danske system, da en undersøgelse lavet på research-tunge universiteter i Southwestern Ontario, Canada, af Lehmann (2007) viser, at studerende bliver stærkt påvirket af de sociale holdninger til penge. I dette tilfælde nævnte flere af de studerende, at de enten følte sig for lavt stillede til at passe ind, eller at de var højt stillede og derfor tvunget ind i søgelyset. Begge perspektiver som har lagt enormt pres på de studerende, som i dette tilfælde er endt med at falde fra deres uddannelse. Det skal dog nævnes, at Lehmann (2007) ikke lavede undersøgelsen for at skabe et generelt billede af problemet, da undersøgelsen kun indbefattede samtaler med 25 personer. Det kan bruges til at støtte op om, at det sociale miljø har betydning uanset område og kontekst.

2.3.4 Efterfølgende jobmuligheder

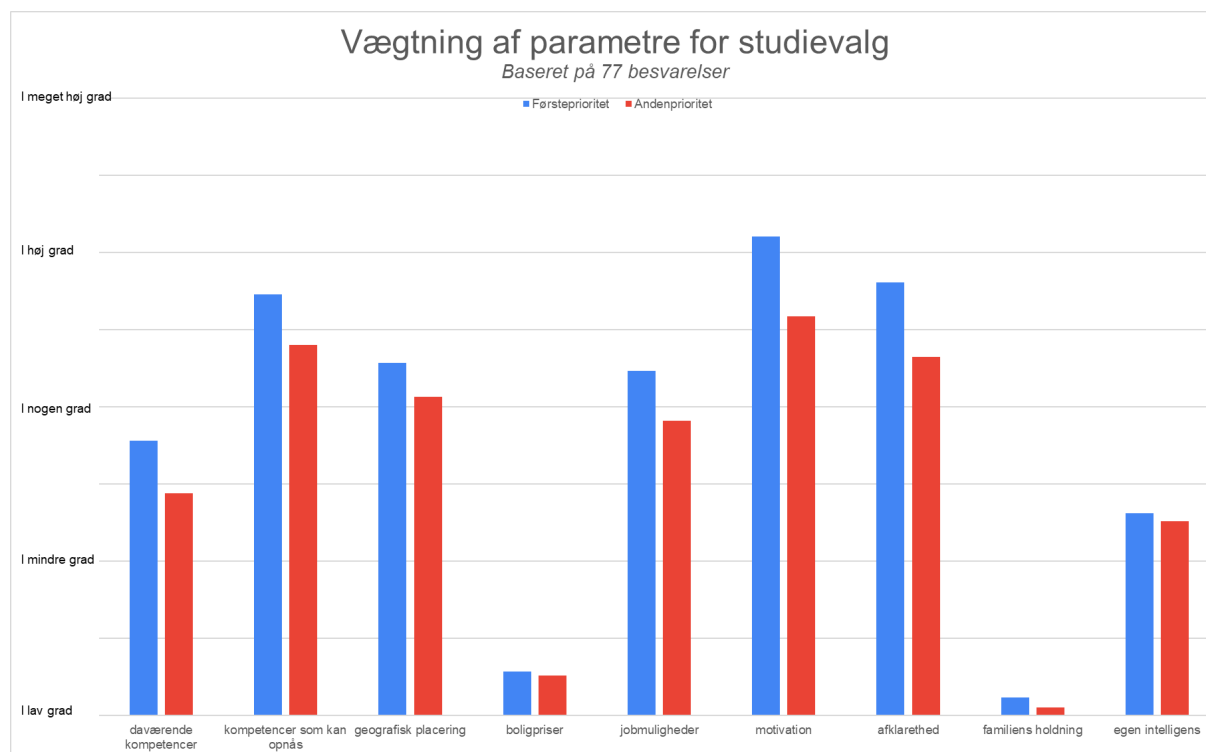
Manglende eller få muligheder for at få et job efter endt uddannelse blev, ifølge Uddannelsens- og Forskningsministeriet (2018) konkluderet som en betydelig forklaring på, hvorfor nogle studerende afbryder deres studier. I rapporten fra UFM blev der beskrevet at 44% af de frafaldne studerende angav, at de jobmuligheder som studiet førte til ikke var tiltalende for dem. Ifølge rapportens undersøgelse var det den anden største årsag til frafaldet blandt de studerende.

Dette område har også vist sig problematisk på de erhvervsfaglige uddannelser, hvor en kvalitativ undersøgelse lavet af Tanggaard (2013), baseret på interviews med 106 nuværende eller tidligere studerende, viste at studerende havde tendens til at falde fra som følge af mangel på praktikplads.

2.3.5 Boligmuligheder

En årsag som har vist sig, ikke at være særligt betydningsfuld for frafaldet blandt de studerende er boligmulighederne i forbindelse med studiet. Med boligmuligheder menes faktorer så som mulighederne for at få en permanent boligløsning, eller boligens generelle stand. Tal fra Uddannelsens- og Forskningsministeriet (2018) viste, at færre end hvert tiende studerende, som faldt fra, angav ufordelagtige boligomstændigheder som værende en betydende årsag til deres frafald. I spørgeskemaundersøgelsen, der er udført i forbindelse med dette projekt, kan man se at betydningen af boligpriser, i korrelation

med resultaterne fra Uddannelsens- og Forskningsministeriet (2018), har betydet meget lidt i de adspurgte studerendes studievalg, se figur 2.5.



Figur 2.5.

2.3.6 Kvaliteten af uddannelsen

40% angiver i undersøgelsen fra Uddannelsens- og Forskningsministeriet (2018), at kvaliteten af undervisningen var årsag til deres frafald. Det gør det til den femte mest betydningsfulde frafaldsårsag. Tanggaard (2013) nævner, at studerende har givet udtryk for læringsinstitutioner med manglende lederskab fra lærere, havde stor effekt på deres beslutning om at falde fra. Samtidig gav studerende udtryk for, at fordi lærere har forskellige læringsmetoder, er det ingen garanti for, at den brugte læringsmetode er bedst egnet til den enkelte studerende. Artiklen beskriver yderligere, at institutionens størrelse også har en nævneværdig indflydelse, da en for stor institution kan få en studerende til at føle sig overset og i den sammenhæng mindre vigtig for det store billede. Der var også beviser for, at den studerendes grundlag for at falde fra, var baseret på individuelle problemer og ikke på grund af institutionsproblemer. Dette postulat bliver understøttet af en ekstensiv undersøgelse lavet af Humlun og Jensen (2010). Denne kvantitative undersøgelse udspurgte 22.029 unge som var gået direkte til en erhvervsfaglig uddannelse, uden en gymnasial uddannelse. Undersøgelsen viser, at den enkelte persons baggrund, herunder køn, etnisk oprindelse, forældres uddannelsesniveau, mm., har stor indflydelse

på den studerendes chance til at komme igennem uddanneles-systemet.

2.3.7 Det individuelle ophav

De tidligere pointer tager udgangspunkt i de valg, som den enkelte studerende selv har haft mulighed for at bestemme over. Humlun og Jensen (2010) tager udgangspunkt i en anden vinkel: Den studerendes ophav og hvordan den har effekt på frafald. Som ovenstående nævner, var undersøgelsen baseret på 22.029 studerende indenfor erhvervsfaglige uddannelser. Ud fra deres data er Tabel 2.1 og Tabel 2.2 blevet lavet, til at fremvise deres fundne resultater. Tabel 2.1 giver et overblik over frafaldssansynligheder ved forskellige selektions-kriterier, hvor *køn*, *etnisk oprindelse*, *familietype*, *forældres indkomstniveau* og *forældres uddannelsesniveau* er i fokus.

Selektionskriterier	Frafaldsrate
Køn	
Mænd	36.9%
Kvinder	39.4%
Etnisk oprindelse	
Danskere og vestlige indvandrere	34.6%
Ikke-vestlige indvandrere	56.8%
Familietype som 15-årig	
Kernefamilie	29.2%
Sammenbragt	42.9%
Enlig forælder	46.9%
Barn bor alene	63.3%
Forældres indkomstniveau	
1. kvartil	51.2%
2. kvartil	37.9%
3. kvartil	30.2%
4. kvartil	26.3%
Mors højeste uddannelsesniveau	
Grundskole	39.8%
Over grundskole	31.1%
Fars højeste uddannelsesniveau	
Grundskole	40.5%
Over grundskole	31.0%

Tabel 2.1. Frafaldrater ved forskellige selektionskriterier (Humlun og Jensen, 2010)

I tabel 2.1 ses en stor procentuel forskel i frafaldsrate under *etnisk oprindelse*, hvor ikke-vestlige indvandrere har en frafaldsrate på 56.8%, mens danskere og vestlige indvandrere kun har en frafaldsrate på 34.6%. Et resultat som bakkes op af kvalitative resultater fra Lehmann (2007) og Sørensen og Ibsen (2017). Konklusionen om vigtigheden af det sociale miljø er stærkt sammenlignelig undersøgelseerne imellem.

Et andet område, hvor der ses stor difference i procentsatsen, er *familietype som 15-årig*. Her er frafaldsraten blevet undersøgt indenfor fire udvalgte familietyper: *Kernefamilie* med 29.2% frafald, *sammenbragt* med 42.9% frafald, *enlig forælder* med 46.9% frafald og *barn som bor alene* med 63.3% frafald. Resultaterne åbner op for spekulationer om, hvorvidt det kun er det sociale miljø på institutionen, som er af betydning, eller om miljøet på hjemmefronten også spiller en rolle. *Forældres indkomstniveau* viser også en effekt på frafaldsraten, da der observeres et signifikant spring fra 51.2% for frafald hos børn med forældre, som har en lav indkomst, til 26.3% for frafald hos børn med forældre, som har en høj indkomst. *Forældres uddannelsesniveau* viser en mindre, men dog betydningsfuld indflydelse på frafald. Der er cirka 10% forskel mellem et barn, som har en forælder med grundskoleuddannelsesniveau, og et barn som har forældre med et uddannelsesniveau over grundskoleniveau. Når kombineret, som set i Tabel 2.2, kan det ses, at der er en relation mellem den frafaldne studerendes køn og forældres uddannelsesniveau. For mænd havde det større betydning om faren havde en højere uddannelse, mens det modsatte gjorde sig gældende for kvinder.

Kombinerede sektionskriterier	Frafaldsrater
Mænd, ikke-vestlige indvandrere	59.4%
Mænd, ikke-vestlige indvandrere, ikke kernefamilie, grundskole far	72.3%
Mænd, ikke-vestlige indvandrere, ikke kernefamilie, grundskole mor	69.8%
Mænd, danskere, ikke kernefamilie, grundskole far	49.7%
Mænd, danskere, ikke kernefamilie, grundskole mor	48.8%
Kvinder, ikke-vestlige indvandrere	53.0%
Kvinder, ikke-vestlige indvandrere, ikke kernefamilie, grundskole far	49.2%
Kvinder, ikke-vestlige indvandrere, ikke kernefamilie, grundskole mor	54.8%
Kvinder, danskere, ikke kernefamilie, grundskole far	47.6%
Kvinder, danskere, ikke kernefamilie, grundskole mor	50.3%

Tabel 2.2. Frafallssansynligheder ved forskellige selektionskriterier når kombineret (Humlun og Jensen, 2010)

I tabel 2.2 ses det, at ikke-vestlige indvandrere generelt har langt større frafalds-procent, hvis de er mænd. Det ses også at køn som parameter har langt lavere betydning for danskere end for andre etniske grupper. Tabellen peger også på at en ikke-vestlig indvandrer muligvis er bedre til socialt at tilpasse sig, hvilket kunne være en proces som gennemgås forskelligt ud fra køn. Dette kunne betyde at danskere ikke behøver gennemgå

den sociale tilvænning og oplever dermed muligvis ikke processen som skaber det store mellemrum mellem mænd og kvinder.

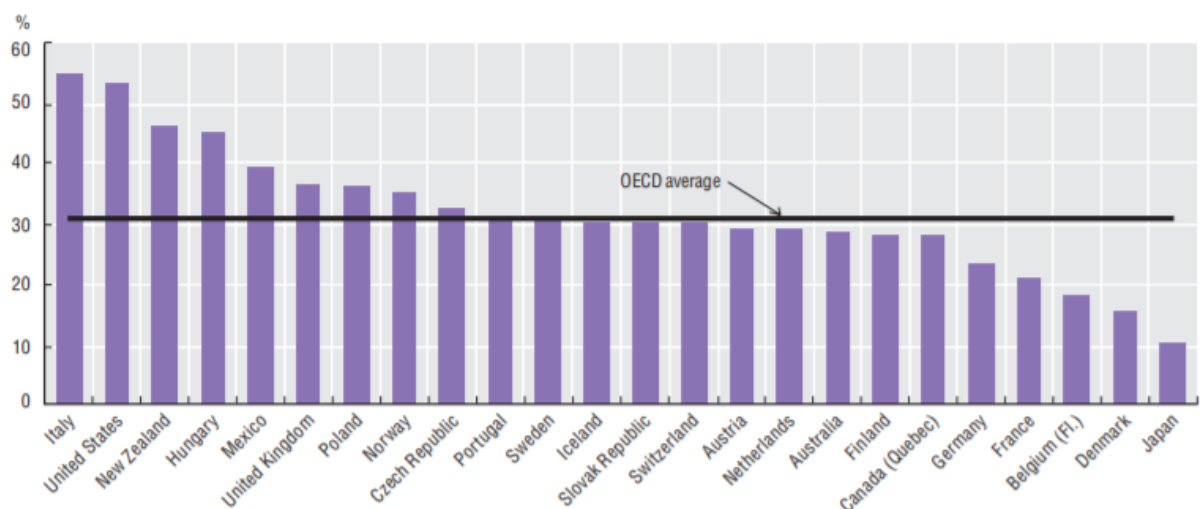
2.3.8 Delkonklusion

Efter at have analyseret hvilke årsager, der kan være skyld i studerendes frafald, kan det konstateres der er mange faktorer på spil. Selvom det er tydeliggjort, at områder som det individuelle ophav har direkte effekt på frafaldsrater, kan det i sammenhæng med denne rapport konkluderes, at det ikke kan påvirkes af et softwareprogram. Derfor mindskes dets relevans for dette projekt. Et område som kan påvirkes er tallene for den enkelte studerendes afklarethed og velovervejethed angående studie prioriteringer. Analysen viser, at mens studerende for det meste er velovervejede angående deres 1. prioritet, er der ofte foretaget færre overvejelser i forhold til 2. prioriteten. Analysen påpeger at det er vigtigt, at den studerende, som eksempel, gør sig tanker om efterfølgende jobmuligheder samt det sociale miljø på studiet. Rapporten vil fremadrettet følge det udgangspunkt, at det er vigtigt for fastholdelsen af den studerende, på en givet uddannelse, at der er foretaget et velovervejet valg, om at starte på netop den uddannelse.

2.4 Frafaldsrater i udlandet

Da systemet i Danmark ikke nødvendigvis fungerer på samme måde som i andre lande, bliver det også taget et kig på hvordan andre lande performer i dette område.

På figur 2.6 ses det at Danmark ligger som et af de lande hvor frafaldsraten er mindst på tertiære uddannelser. Derudover ses det at Japan er det eneste land som har en mindre frafaldsrate end Danmark, hvilket er interessant at kigge videre på. Sammenlignes de tidligere nævnte frafaldsrater for Danmark i denne rapport med nedenstående graf, ses en afvigelse. Nedenstående grafs data er fra 2007, og det formodes at fremdriftsreformen er skyld i større frafald siden da (Ritzau, 2018).



Figur 2.6. Viser frafaldsrater i procent for de pågældende lande(OECD, 2008)

Newby et al. (2009) har også gennemgået det japanske uddannelsessystem og ledt efter kilder til deres fortrinlige succes indenfor gennemførsel af studier. Ifølge rapporten har hele den japanske kultur stor indflydelse på hvordan den studerende klarer sig, og unge bliver fra en meget tidlig alder gjort opmærksom på hvor vigtig uddannelse er for samfundet og dets videreudvikling.

Ifølge Koichi (2014) fremhæver Japansk kultur at alle er i stand til at gennemføre en videregående uddannelse, så længe de arbejder hårdt nok. Hvis en studerende ikke gennemfører eksamen bliver skylden næsten aldrig pålagt læreren eller forældrene, som udgangspunkt er det altid den studerendes ansvar. Miljøet på gymnasiet og universitetet er dog skruet lidt anderledes sammen, sammenlignet med det danske. Mange unge frygter den gymnasiale eksamenstid i Japan, da det kræver exceptionelle huskeevner, men når man igennem til universitetet bliver fokuset mere på at gøre de studerende klar til

erhvervslivet med de rigtige kompetencer. Derfor er universitetet lettere at gennemføre, hvilket også afspejles i frafaldsraterne.

På trods af at forskellen mellem den japanske og danske frafaldsrate er omkring 5%, er de kulturelle forskelle for store til at den japanske model kan implementeres i det danske uddannelsesmiljø.

For at bekæmpe det problem der eksisterer omkring frafald i Danmark, er der således ikke mulighed for at finde den store inspiration i udlandet, hvor frafaldsraten kun er større.

2.5 Studievalgsprioritering

Baseret på den indsamlede information, kan det argumenteres, at uddannelsesprioriteter kan have direkte indflydelse på frafalds-rater. Derfor vil dette afsnit, give et indblik i hvordan valg af prioritering fungerer, samt hvordan systemet virker som helhed.

Ifølge UddannelsesGuiden (2019g) foregår ansøgning til videregående uddannelser i Danmark gennem webstedet optagelse.dk. UddannelsesGuiden skriver, at det er muligt at søge ind på en videregående uddannelse, efter ansøgeren har færdiggjort en ungdomsuddannelse eller en uddannelse på lignende niveau. Videre skrives det at ansøgeren har mulighed for at prioritere sine ønsker til videregående uddannelse. I tilfælde af at ansøgeren ikke bliver optaget på sin 1. prioritet vil pågældende i stedet blive taget i betragtning til optagelse på 2. prioritetsvalget. Ligeledes, hvis 2. prioriteten heller ikke er en mulighed rykkes der videre til 3. prioritet.

Ansøgeren vil blive tilbudt en studieplads sidst i juli, såfremt vedkommende er kommet ind (UddannelsesGuiden, 2019g). Uddannelsen påbegyndes typisk i august eller september. Derfor er det væsentligt både at overveje, hvilken uddannelse pågældende helst vil ind på, samt hvilke alternativer der eksisterer, hvis 1. prioriteten ikke er en mulighed. Dette kunne være en uddannelse der ligner, eller måske noget helt andet. Det kan lade sig gøre at have op til 8 uddannelser på optagelse.dk i prioriteret rækkefølge (UddannelsesGuiden, 2019g). Interfacet for prioritering af de op til 8 uddannelser fremgår af figur 2.7.



Søg videregående uddannelse [Skift til digital ansøgning](#)

1 Personoplysninger **2** Adgangsgrundlag **3** Uddannelsesvalg **4** Afslut ansøgning

1 Tilføj prioritet **2** Tilføj prioritet **3** Tilføj prioritet **4** Tilføj prioritet **5** Tilføj prioritet **6** Tilføj prioritet **7** Tilføj prioritet **8** Tilføj prioritet

Vælg op til 8 uddannelsesønsker

Når du har valgt - ved at trykke på knappen "Tilføj uddannelse" - vil ønsket blive vist i prioritetslisten til venstre. [?](#)
Der vil du med pilene kunne ændre den prioriterede rækkefølge.

Region Sprog Gruppér efter

Uddannelsesstype Uddannelsesområde

Søg efter [Vis uddannelser](#)

[Gå til forsiden](#) [Fortsæt](#)

Figur 2.7. Prioritering af uddannelser på optagelse.dk. (UddannelsesGuiden, 2019g)

Nogle vigtige ting at være opmærksomme på er blandt andet at alle ansøgninger behandles ens, uanset hvilket prioriteringsnummer ansøgningen har. Uddannelsesstederne vurderer ansøgningen udelukkende ud fra kvalifikationer. For kvote 1 vil det altså sige at vurderingen foretages alene ud fra det opnåede eksamensresultat fra sin tidligere ungdoms- eller erhvervsuddannelse. Da der kun tilbydes én studieplads, bliver der ikke tilbudt uddannelser af lavere prioritering end den tilbudte selvom ansøgeren er kvalificeret til optagelse her. (UddannelsesGuiden, 2019g).

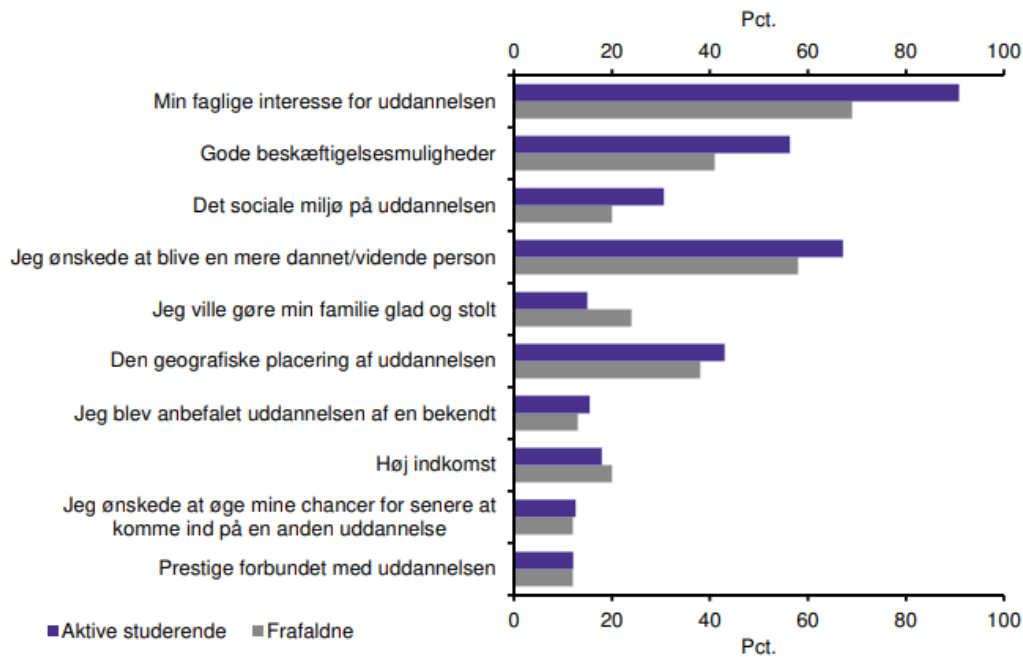
2.6 Parametre der indgår i det gode studievalg

I undersøgelsen foretaget af Danmarks Evalueringsinstitut (2018), beskrives studievalgsprocessen for de afklarede studerende. Med udgangspunkt i sådanne konklusioner bør det være muligt at udpege hvilke overvejelser der er vigtige når der skal foretages det gode uddannelsesvalg. Både 1. og 2. prioritet er som nævnt vigtigt at overveje, så de overvejelser der bliver gjort i forbindelse med 1. prioritet skulle også gerne gøres ift. 2. prioritet.

Undersøgelsen foretaget af (Danmarks Evalueringsinstitut, 2018), peger på, at de afklarede studerene foretager deres studievalg på et langt bedre grundlag end de studerende, som ikke er afklaret. Derudover er de afklarede studerene meget aktive ift. at indsamle informationer på flere forskellige måder. Dette er både at finde og læse litteratur om hvad uddannelsen indebærer, men også at have haft kontakt med ansatte og studerende på de studier de er interesserede i. Derpå konkluderer rapporten, at der er en korrelation imellem hvor meget viden man har indsamlet om studiet man søger ind på, og hvor afklaret valget er.

Mange faktorer indgår i valget af videre uddannelse efter gymnasiet eller tilsvarende uddannelse. Det kunne være den geografiske placering af diverse universiteter, forældres påvirkning, boligpriser, de efterfølgende jobmuligheder og løn efter endt uddannelse, kompetencerne man allerede besidder eller dem man kunne tænke sig at erhverve. Flere af disse ting må ansøgeren overveje.

Af Figur 2.8 vises en sammenligning af henholdsvis aktive og frafaldne studerende og i hvor høj grad forskellige årsager var af betydning for deres valg af studie. Det fremgår heraf at den mest betydningsfulde årsag er den faglige interesse for uddannelsen.



Figur 2.8. Årsager til studievalg. (Uddannelsens- og Forskningsministeriet, 2018)

En af ressourcerne som er udviklet for at hjælpe den studerende med at blive afklaret med sit studievalg, er en liste af gode overvejelser udviklet af UddannelsesGuiden (UG). I listen har UG skrevet de overvejelser, man som ansøger kan tage i betragtning, når man skal vælge studie. Her fokuseres primært på praktiske forhold, der rækker udover den faglige interesse i studiet (UddannelsesGuiden, 2019a).

Her er et resume af listen:

- **Uddannelsens beliggenhed** er et af de væsentlige punkter ansøgeren skal have med i overvejelserne. Mens nogle uddannelser kun kan tages på udvalgte uddannelsesinstitutioner, er der mange studier der udbydes flere steder. Nødvendigheden af at pendle, ønsket på at starte på frisk et nyt sted kan være nogle af de tanker ansøgeren gør sig.
- **Uddannelsens fokus på gruppearbejde** kunne være vigtigt for nogle, hvor forskellige uddannelser differentierer sig i hvor integreret en del gruppearbejdet er. Der er endda forskel på universiteters tilgang til gruppearbejdet helt overordnet.
- **Uddannelsens længde** varierer afhængigt af om det er en erhvervsakademiuddannelse (2 år), professionsbacheloruddannelse (3,5-4 år) eller en universitetsuddannelse (5 år). Ansøgeren bør overveje om det er vigtigt hvor længe man studerer.
- **Det sociale miljø på uddannelsen** kan have noget at sige for valget ift. hvordan sin dag er som studerende ser ud. Nogle uddannelser er store og har derved mange sociale arrangementer, mens andre jo ikke opfylder dette. Men samtidigt kan et mindre studie have et tættere sammenhold.

2.6.1 Generalisering af det gode studievalg

På trods af at UG har forsøgt at vejlede generelt om, hvilke parametre som danner grundlag for det gode studievalg, så kan det stadig være rigtig svært at træffe.

Bilag B er et uddrag af data fra spørgeskemaet som blev foretaget i forbindelse med rapporten. På Bilag B ses det, hvordan studerende, som succesfuldt afsluttede deres første studieår på en videregående uddannelse, vægter forskellige parametre for deres studievalg. I Bilag B observeres en relativt lige svarfordeling blandt svarpersonernes besvarelser. Hvis det gode studievalg defineres til, at man har gennemført første studieår, kan det ud fra data konkluderes at det gode studievalg er meget individuelt.

Fordi der findes så mange forskellige uddannelser, skal unge ifølge Bessie Rauff (2019) droppe idéen om drømmeuddannelsen. Ifølge artiklen findes der ikke blot et men mange gode valg. På den baggrund bør de fleste ansøgere kunne sætte mere end én prioritering på prioriteringslisten på optagelse.dk, og være tilfreds med at gennemføre uddannelsen uanset om det var deres 1. prioritet eller 2. prioritet.

2.6.2 Delkonklusion

Det kan konkluderes, at der er visse parametre, som kan være betydningsfulde for at træffe det gode studievalg, der ikke leder til frafald fra studiet. Især er den faglige interesse for uddannelsen vigtig, samt at den studerende har indsigt i, hvad uddannelsen indebærer. Dette gælder både for 1. og 2. prioriteten og så videre. Herudover er det også værd at bemærke, at der findes mange lignende uddannelser, og at det derfor bør være muligt at finde flere uddannelser som man vil være tilfreds med at gennemføre.

2.7 Eksisterende værktøjer til studievalg

Når kommende studerende skal vælge studie, tilbydes der på UG's hjemmeside (UddannelsesGuiden, 2019b) en række værktøjer, som hver forsøger at pege brugeren i den rigtige retning. Af de præsenterede værktøjer adresserer fire af dem sig til den studiesøgende:

- **Adgangskortet** kan bruges som det første værktøj til at pege den søgende i den rigtige retning. Den angiver alle muligheder indenfor den studie-søgendes gymnasiale fagkombination. Studierne er inddelt i kategorier, så brugeren bliver peget i retning

af bestemte uddannelses typer. Værktøjet kan dog være meget uoverskueligt, da der er uoverskueligt mange uddannelser, at kigge igennem UddannelsesGuiden (2019c).

- **Uddannelses-zoom** opstiller to til tre forskellige uddannelser og sammenligner derefter deres vurderinger i udvalgte kategorier. Den sammenligner ikke det faglige indhold, men kun elevernes holdning til det sociale miljø, faglige indhold og udbytte af studiet. Det sammenholder også uddannelsen med karrieremuligheder. Dette værktøj forudsætter derfor, at brugeren har afgrænset studievalget til maksimalt tre studier UddannelsesGuiden (2019f).
- **Studievælgeren** tager højde for interesser, ønsket uddannelse, geografi og forventninger. Derefter foreslår den en handlingsplan. Denne handlingsplan definerer fire punkter, brugeren skal igennem for at blive mere afklaret med sit valg. Dog er tre af disse fire punkter fastlagt som det samme for alle punkter. De tre fastlagte punkter er, at foreslå at læse videre om uddannelsen på UG.dk, tage til åbent hus eller kontakte en vejleder. Det sidste punkt sender dig videre til uddannelseszoom, hvor forventninger holdes op mod data fra virkeligheden. Derfor opstår der spildarbejde, da brugeren indtaster en masse data, som ikke bliver taget i betragtning. UddannelsesGuiden (2019e).
- **Jobkompasset** fremlægger en række forskellige temaer for uddannelser. Her vælges et tema og en række uddannelser vises frem. Alle uddannelserne er inddelt efter en faglig retning. Hvis titlen på uddannelsen fanger interesse, henvises vedkommende til at læse videre på UG.dk UddannelsesGuiden (2019d).

Den kommende studerende bliver gennem UG præsenteret for disse værktøjer, men en gennemgående tendens hos alle værktøjerne er uoverskuelighed. Denne uoverskuelighed kommer af for mange muligheder, og derfor gennemgår brugeren ikke alle muligheder, men mister derimod lysten til at bruge værktøjet. En anden gennemgående trend for alle værktøjerne er, at de alle henviser til UG.dk, hvor der yderligere kan læses om uddannelserne. Derved fremtræder værktøjet som ligegyldigt, når de alle giver det samme svar, trods forskellige metoder til at frembringe resultatet.

På nettet eksisterer der også andre værktøjer til hjælp med uddannelsesvalg. En af disse er hjemmesiden Studentum (2019). Her bliver brugeren stillet en række spørgsmål, som i sidste ende viser pågældende score blandt de forskellige fakulteter defineret af Studentum.

2.8 Metodeovervejelser

Ifølge Olsen (2006) så rent metodisk var spørgeskemaet teoretisk set udvalgt til et godt resultat indenfor de svar der skulle komme ud af det. Men en udfordring er en meget

tydelig alder som svarpersonerne har. omkring 80 % af svarpersonerne var under 25 år, hvilket viser, at spørgeskemaet har ramt mange fra den samme målgruppe.

Spørgeskemaet manglede i høj grad flere respondenter, som havde droppet ud af sit studie førhen. Dette er ærgeligt, da der kun var ca. 11,2 % der havde svaret, at de førhen havde droppet ud. Dette var meget få og det kan skyldes, at svarpersonerne har set det som et følsomt område. Det kan skyldes "betydningsvidden" af ordet, da ord forstås vildt forskelligt fra person til person. Nogle personer har måske anset ordet at droppe ud som negativt, hvis de blot mener at de har "skiftet" studie evt. denne betydningsvidde har muligvis været skyld i de få besvarelser fra folk der førhen har droppet ud af et studie (Olsen, 2006). Det kan dermed også skyldes at spørgeskemaet har ramt en forkert målgruppe, da spørgeskemaet i høj grad har ramt en meget snæver omgangskreds.

Spørgeskemaet havde derudover nogle klassiske og basale fejl, som at spørgeskemaet manglede i nogle situationer, at man som svarperson kunne svare "Jeg er ikke kommet ind på 1. eller 2. prioritet". Dette blev senere rettet, men kan evt. have ledt til fejldata.

2.8.1 Fordele og ulemper ved spørgeskema

Når man ser på fordelene ved spørgeskema, så er der meget, som rent teoretisk kunne have været komplimenterende for undersøgelsen af frafald, men mange af disse fordele løb ud i vandet i praksis (Olsen, 2006).

POST- SPØRGE- SKEMA	Spørgeskema	Spørgeskema
	<ul style="list-style-type: none"> • Kun hensyn til svarperson • Mulighed for åbne spørgsmål • Visuelle hjælpemidler • "Følsomme" spørgsmål <p><i>Andre fordele</i></p> <ul style="list-style-type: none"> • Tilsikret anonymitet • Tidsbesparende • Moderate omkostninger • Store stikprøver • Geografisk spredning • Valg af eget svar tempo • Ingen interviewer- eller samspilsvirkninger 	<ul style="list-style-type: none"> • Fuldstændigt selvforklarende • Korte spørgsmål • Sprogligt enkle spørgsmål • Få åbne spørgsmål • Filterspørgsmål problematiske • Ingen kundskabsspørgsmål • Særlig formgivning <p><i>Andre ulemper</i></p> <ul style="list-style-type: none"> • Risiko for manglende svar motivation • Relativt stort bortfald • Ingen uddybende sonderinger • Ingen kontrol af svarperson • Reduceret læsefærdighed, fx på grund af funktionel analfabetisme • Ingen mulig svarkontrol

Figur 2.9. Fordele og ulemper

For eksempel så er de "følsomme" spørgsmål nemmere at svare på. Det tyder på, at de følsomme spørgsmål har skræmt svarpersonerne rigeligt alligevel. Spørgeskemaets

anonymitet havde derudover tiltænkt, at det ikke ville påvirke besvarelsene.

Ulemperne var større end forventet ift. undersøgelsen. Svarmotivationen virker til at have været et problem, da der forekom en del useriøse besvarelser. Derudover så er det foretagne spørgeskemaundersøgelse ikke fulgt op med et interview, hvilket kan betyde at vores korte og meget lukkede spørgsmål ikke kan forklare tendensen, men blot bekræfte en tendens i samfundet. Det er derfor en ulempe, hvis der ikke er mulighed for at forklare sig i særlig høj grad i et spørgeskema (Olsen, 2006).

3 | Problemformulering

Som led i P1-forløbet, skal der som en væsentlig del af projektarbejdet produceres et mindre program af høj kvalitet, og dette skal gøres i programmeringssproget C. På baggrund af problemanalysen opstilles således et afgrænset problem, der ønskes taklet gennem et sådant program.

Frafald på videregående uddannelser er som beskrevet et større problem, som har konsekvenser for det enkelte individ, der står overfor et nederlag, for institutioner og for samfundet generelt. På verdensplan klarer Danmark sig dog godt, hvor kun Japan har en mindre frafaldsrate, så den store inspiration til forbedring kan ikke findes i udlandet. Et tal der potentielt kunne sænkes, er det procentuelle frafald af studerende der er kommet ind på en 2. prioritet, da dette tal er ca. 50% større end frafaldet for 1. prioritet. Med mere end 900 videregående uddannelser i Danmark, kan det antages at der eksisterer en lignende uddannelse, der også er tilfredsstillende som ansøgeren kan komme ind på. Særligt når begrebet om den 'perfekte uddannelse' ikke findes, bør en løsning på den høje frafaldsrate kunne realiseres. Udover det viser flere undersøgelser, heriblandt spørgeskemaundersøgelsen udført i forbindelse med denne rapport, at 2. prioritetsvalget er knap så velovervejet som 1. prioritetsvalget, hvilket kan skyldes et dårligt valg, der ikke har gennemgået de betydningsfulde overvejelser i samme grad fra som 1. prioriteten. Der findes allerede en del værktøjer til at afklare studievalget, men det er særligt rettet mod at finde ét studie.

Med henblik på at reducere frafaldsraten blandt studerende, der bliver optaget på deres 2. prioritet er følgende problemformulering blevet udformet:

Hvordan kan der udvikles et program i C, som anbefaler en 2. prioritet til ansøgeren af en videregående uddannelse, på baggrund af den valgte 1. prioritet?

4 | Problemløsning

Baseret på rapportens problemanalyse, er det muligt at tackle løsningen af problemet med op til flere tilgange. Det kommende afsnit vil gennemgå disse muligheder, samt hvorfor den endelige retning er blevet valgt. Dog vil teknologien ikke blive diskuteret, da den er blevet fastsat på forhånd af studieordningen. Der skal udvikles et program i programmeringssproget C.

Som det første i dette afsnit er der en klargjort kravspecifikation (Afsnit 4.1), som fungerer som den endelige tjekliste for et færdiggjort og fyldestgørende program. Som opfølgning på dette er der et designafsnit, (Afsnit 4.2). Dette afsnit vil beskrive opbygningen af programmet herunder filtreringsprocessen. Til sidst i afsnittet findes en algoritmisk kravspecifikation (Afsnit 4.2.3).

Et større implementeringsafsnit vil beskrive, og vise det udviklede program med forklaringer af de centrale datastrukturer, variabler og funktioner gennem udklip fra koden (Afsnit 4.3)

4.1 Kravspecifikationer

For at kunne udvikle en løsning som besvarer problemformuleringen, er der opstillet en række kravspecifikationer, som det færdige program ideelt set skal overholde. Disse kravspecifikationer er lavet ud fra problemformuleringen i kapitel 3, hvilket giver basis for de kommende afsnit; design (Afsnit 4.2) og implementering (Afsnit 4.3). Der vil også blive fremlagt algoritmiske krav i afsnit 4.2.3, som vil give et overblik over hvilke krav, som stilles for programmet i et teknisk perspektiv.

Programmet skal leve op til følgende krav:

- Have kendskab til alle bacheloruddannelser i Danmark.
- Have informationer om diverse parametre vedrørende uddannelser, så forskellige uddannelser kan sammenlignes med hinanden

- Studielokation: Forstået som uddannelsesby.
 - Arbejdsform: Forstået som individuelt- eller gruppearbejde.
 - Lignende uddannelser: Forstået som de uddannelser som UG har registreret som værende lignende i forhold til fagligt indhold og de kompetencer som opnås via uddannelsen.
 - Det generelle sociale miljø: Forstået ved brug af Likert-skalaen med faktorer, som "ensomhed, stress" osv.
- Vurdere om en brugerindtastet uddannelse eksisterer i databasen af uddannelser som programmet har kendskab til.
 - Udpege lignende uddannelser for enhver valgt uddannelse.
 - Filtrere uddannelser fra, som ikke passer sammen med det valgte bruger-input.
 - Sortere resterende uddannelser baseret på det generelle sociale miljø og fremvise det bedste mulige foreslag til en 2. prioritets uddannelse.

4.2 Design

Første punkt i kravspecifikationen konstaterer, at programmet skal have kendskab til alle bacheloruddannelser i Danmark, samt have adgang til udvalgte informationer om hver uddannelse. Dette er ikke blot en stor mængde data, men også data som gerne ændrer sig fra år til år. Til fordel for projektet og programmet har Uddannelses- og Forskningsministeriet offentliggjort en fil, som både giver en oversigt over alle uddannelser i Danmark samt specifik data omkring disse uddannelser. Dette gør projektets opgave markant mindre, eftersom det ikke er nødvendigt selv at udvikle en data-oversigt. I stedet kan en tilpasset version af filen fra Uddannelses og Forskningsministeriet anvendes. I afsnit 4.2.1 gives en oversigt over de parametre, der blev fremvist i problemanalysen som værende relevante for studerendes frafaldsrater, samt hvordan de vil blive håndteret, og hvor deres data kommer fra.

4.2.1 Parametre relevante for frafald

Tidligere i rapporten er det blevet beskrevet, at der er parametre med høj betydning for frafald, som ikke er mulige, i projektets sammenhæng, at påvirke eller måle på. Et eksempel herpå er parameteren *individuelt ophav*, som er beskrevet i afsnit 2.3.7. Da sådanne parametre ikke kan indgå i problemløsningen, er det vigtigt at danne en ramme om de parametre, som er inkluderbare. Baseret på den tidligere analyse kan en

parameters indflydelse deles op i to egenskaber: Dens direkte effekt på frafaldsrate samt dens betydning for individet. Overordnet er der følgende relevante parametre, som er baseret på resultaterne fra det fremstillede spørgeskema i rapporten: *Kompetencer som kan opnås, jobmuligheder* samt *motivation og afklarethed*. Spørgeskemaet undersøgte også:

- *Daværende kompetencer*: I hvilken grad den tilspurgte mente, at deres daværende kompetencer havde effekt på valg af studieretning
- *Boligpriser*: I hvilken grad den tilspurgte mente, at boligpriser spillede en rolle i deres valg af studieretning
- *Egen intelligens*: I hvilken grad de valgte studieretning, baseret på deres egen vurdering af deres intelligens

Disse parametre vil ikke blive brugt i kommende design, da de enten er meget lavt vægtet af studerende ifm. valg af første prioritet, eller har korrelation med høje frafaldsrater. Med andre ord: Hvis den kommende studerende prioriterede disse parametre højt, havde de større sandsynlighed for at falde fra. Derfor bliver disse parametre ikke brugt i det færdige program. Derudover er der på baggrund af analysen taget et valg om, at brugeren ikke skal have mulighed for at overveje dem.

På basis af kvalitative undersøgelser i problemanalysen, vil designet supplerende bruge parameteren *Studiemiljø*. Dette resulterer i følgende endelige oversigt over parametrene, der tages højde for:

- Kompetencer som kan opnås
- Geografisk placering
- Jobmuligheder
- Motivation
- Afklarethed
- Studiemiljø

Da dette er meget brede begreber, er hvert parameter blevet forsimplet til en mere håndgribelig størrelse, så det er muligt at arbejde med den i et program.

Kompetencer som kan opnås

Det kan være en stor udfordring at sætte rammer for hvilke kompetencer, der kan opnås på diverse uddannelser. En metode er at undersøge den individuelle uddannelses studieplan, hvilken kan sammenlignes med nærliggende uddannelser. Dette er dog en tidskrævende opgave, da dette ikke er information som ligger offentligt tilgængeligt i et overskueligt format. Alternativt har UddannelsesGuiden (UG) en sektion for hver af deres uddannelser,

kaldet *Lignende uddannelser*. Det er implicit, at denne sektion er baseret på, at de fremlagte uddannelser og den originale uddannelse har sammenlignelige kompetence-mål. Det er derfor besluttet for dette design, at bruge *Lignende uddannelser* modulet fra UG. Dette vil fungere som det første lag af programmets filtrering af uddannelser. Denne proces er yderligere forklaret i afsnit 4.2.2. Beslutningen beror på at UddannelsesGuiden selv har angivet nogle lignende uddannelser.

Geografisk placering

Hvis der skal tages højde for geografisk placering, ses der fire muligheder, *distance i kilometer*, *institut*, *region* og *kommune*. I et større program, kunne der potentielt laves en løsning, som tager højde for flere af disse parametre, men da kravene til dette program er, at man skal udarbejde et mindre program af høj kvalitet, tages der kun højde for én af disse parametre. *Distance i kilometer* bliver meget hurtigt et omfattende aspekt at måle på, da der skal tages højde for, om der måles i fugleflugt eller via vejnettet. Derudover kan distance i kilometer ikke oversættes direkte til den tid det tager at nå frem. *Institution* er meget bredt. Aalborg Universitet har flere institutioner i landet, hvilket betyder at denne information fra brugeren, ikke er nok til at vurdere, hvor brugeren gerne vil studere. *Region* er også meget bredt, da der er eksempler på regioner med flere universiteter. *Kommune* er til gengæld specifikt nok til, at det er muligt at lave en direkte vurdering af, præcist hvilket universitet den studerende ønskede, da de indtastede deres information. Dog kunne et større program tage højde for flere af disse parametre. Som eksempel kunne et scenarie være, at en ansøger gerne vil studere i en region, men er ligeglad med den specifikke kommune. For simplicitetens skyld er det besluttet, at dette design kun fokuserer på kommuner. Dataet omhandlende de enkelte uddannelsers beliggenhed bliver indlæst fra UFM's data-fil.

Jobmuligheder

Jobmuligheder dækker over en række emner. Herunder tilgængelige typer job, specifikke stillinger, arbejdspladser, og ledighedsprocenter. Da det er en ekstremt stor opgave at vurdere hvilke uddannelser ender ud i hvilke specifikke job, vælger dette design at fokusere på ledighedsprocenten.

Job er meget varierende når det kommer til navne, arbejdsområder, og kvalificering. En nyuddannelses uddannelse låser ikke nødvendigvis personen fast til ét job. Derimod er det i højere grad den enkeltes personlighed og kompetencer, som har effekt på, hvilke jobs der kan opnås. I modsætning til ovenstående er det nemt at benytte ledighedsprocenter

målt på færdiguddannede som parameter. Dette område viser sig dog uholdbart at vurdere i dette projekt, da programmet kun fokuserer på bacheloruddannelser, og der enten er lidt eller slet ingen data på jobmuligheder for bacheloruddannelser. På dette grundlag vil programmet i denne rapport ikke tage højde for denne parameter, men i et udvidet program, hvor der også bliver taget højde for kandidatuddannelser eller professionsbachelor, vil dette være en relevant parameter at inkludere.

Afklarethed og motivation

Undersøgelsen viser, at afklarethed og motivation er to af de større faktorer for frafaldsraten. Disse parametre er svære at få brugeren til at give et direkte input på. I dette projekt er afklarethed og motivation taget højde for som følge af, at brugeren har valgt en 1. prioritet, der giver motivation, og som brugeren er afklaret omkring. Hermed vil brugeren også have motivation for den 2. prioritet som programmet giver som output, da denne uddannelse er en af 1. prioritetens lignende uddannelser.

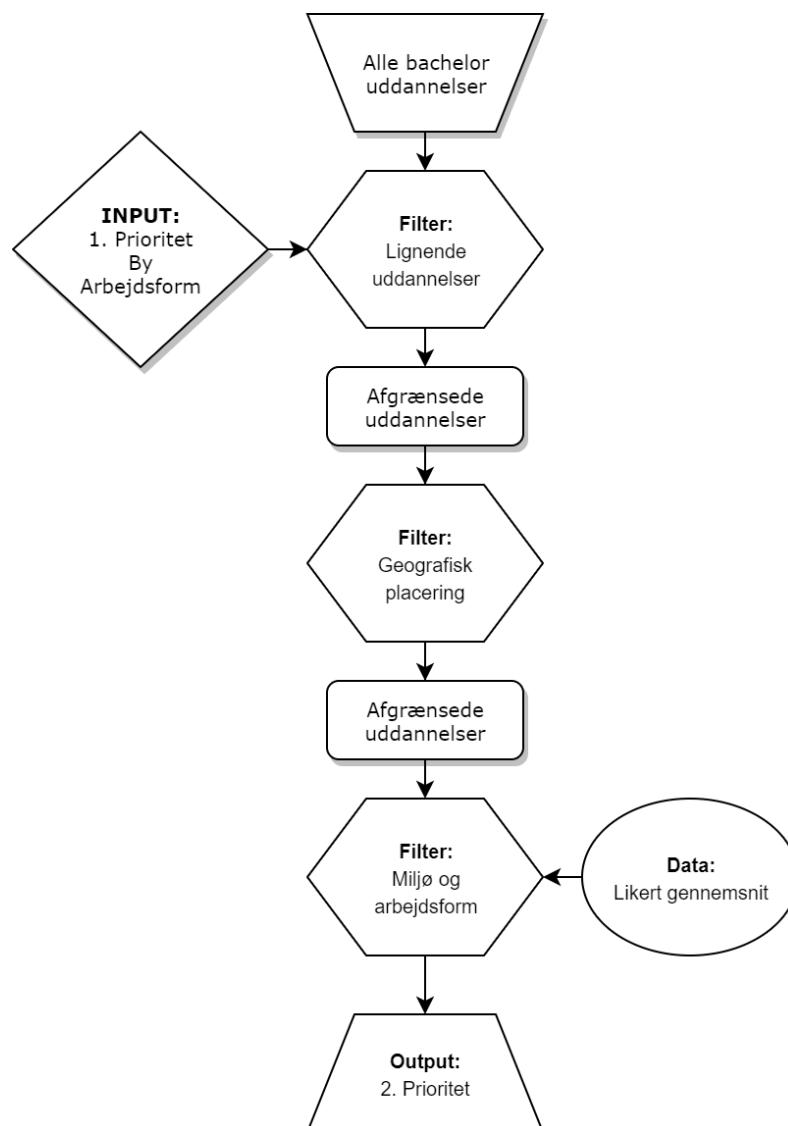
Studiemiljø

Begrebet studiemiljø kan være bredt for det enkelte individ, hvilket betyder, at det kan være svært at vægte hvad der skal tages højde for. Dataet fra UFM inkluderer information om hver enkelt uddannelse samt diverse undersøgelser. Disse undersøgelser har benyttet sig af Likert-skalaen som beskrevet i afsnit 2.1.1 og berører områderne: *Socialt miljø, engagement fra underviser, ensomhed, daglig stress* samt *hvor tilpas den studerende føler sig*. For at danne et generelt billede af studiemiljøet på uddannelserne er der blevet lavet en gennemsnitsscore for alle disse områder. Gennemsnittet bliver brugt i filtreringsprocessen i programmet. Da analysen viste, at dette er en vigtig parameter ift. studerendes frafaldsrate, vil brugeren ikke blive promptet for input angående ønske om studiemiljøet, i stedet vil data fra UFM's fil blive taget i brug.

Der er dog et emne, som hører til under studiemiljø, der er præferencebaseret: *individuel- eller gruppe-orienteret arbejde*. Denne information er ligeledes stillet til rådighed af UFM, og brugeren vil derfor kun blive spurgt om, hvilken arbejdsform de foretrækker, hvorefter programmet vil tage højde for valget i den fremtidige filtrering.

4.2.2 Proces og programflow

Med en oversigt over hvad de enkelte parametre indebærer, kan der dannes en samlet procesoversigt, som giver et forkromet overblik over, hvordan programmet designmæssigt ser ud fra start til slut. Som værktøj til at danne sig et visuelt overblik, er figur 4.1 udarbejdet. Figuren viser programflowet. Følgende afsnit vil uddybe hvert trin.



Figur 4.1. Flow af program proces

Modellens prioriteringsrækkefølge (figur 4.1) er baseret på den tidligere fremlagte analyse. Målt imod effekten på frafaldsrate, viste det sig vigtigst, at den studerende, grundigt, havde overvejet sit valg af *lignende uddannelser* samt *geografisk placering* i prioriteret rækkefølge. *Socialt miljø* er et filter, som er tilføjet baseret på kvalitative undersøgelser. Det er derfor svært at vurdere, hvor denne parameter ligger i prioriteringsrækkefølgen, men i dette design, hvor dets sociale miljø baseres på en 1-5 Likert-skala, som kan sammenlignes

på tværs af uddannelserne, er det blevet placeret i slutningen af processen. Dette sørger også for, at programmet med sikkerhed kun outputter én 2. prioritet, da det kan nøjes med at vælge den sidststående uddannelse med den højeste gennemsnitlige studiemiljøscore.

Programmet initialiseres med en liste over alle bacheloruddannelser i Danmark, hvorefter brugeren bliver promptet for at indtaste sin valgte 1. prioritet, derefter bliver de spurgt om hvilke kommuner de ønsker at studere i, og til sidst spørger programmet hvilken arbejdsform brugeren foretrækker, brugeren kan enten vælge individuelt- eller gruppearbejde eller begge dele. Når disse valg er foretaget, vil programmet finde lignende uddannelser baseret på den valgte 1. prioritet og påbegynde en filtreringsprocess, hvor Uddannelser som ikke matcher brugerens valg bliver filtreret fra. Til sidst tager programmet højde for områderne beskrevet i afsnit 4.2.1, hvilket fungerer ved brug af gennemsnitsværdien fra *sociale miljø, engagement fra underviser, ensomhed, daglig stress og om den studerende er tilpas* i form af Likert-skalaen. Dette ender ud i et output af den 2. prioritet med den bedste score.

4.2.3 Algoritmiske krav

Ift. de mere generelt stillede krav om programmets funktionalitet, laves et andet sæt af krav, der ligger tættere på koden. Følgende er udarbejdet ud fra det beskrevne programflow.

Programmet skal kunne:

- Oprette en database ud fra en fil med alle bacheloruddannelser.
- Indlæse uddannelsernes data ind i structs.
- Structs skal kunne indeholde al information om uddannelsen, såsom uddannelsesby, uddannelsens arbejdsmetode og lignende uddannelser mm.
- Tjekke om en bacheloruddannelse er indeholdt i databasen
- Tage brugerens 1. prioritet som input.
- Tage én eller flere kommuner fra en liste over kommuner, hvor der kan læses en akademisk bachelor i Danmark som input.
- Tage brugerens præference indenfor arbejds-orientering som input.
- Generere lignende uddannelser i en liste, A, ud fra bruger input.
- filtrer liste A baseret på brugerens input.
- Fremvise den bedste 2. prioritets uddannelse samt *URL* til uddannelsen på UG.

4.3 Implementering

Det følgende afsnit omhandler, hvordan det fremlagte design kan realiseres, ved hjælp af en teknisk løsning. Der vil løbende i afsnittet blive indsat snippets fra koden, men ikke alt kode vil være repræsenteret i afsnittet. Hvis der er yderligere interesse for koden, kan den findes i bilag D.

Som grundlag for at kunne organisere projektet, er der lavet en overordnet struktur både for koden, men også for den data som bliver opbevaret og bearbejdet. Dette er forklaret yderligere i afsnit 4.3.1 og afsnit 4.3.4. På trods af at projektet er af mindre størrelse, er det besluttet at arbejde i flere C filer, for nemmere at vedligeholde strukturen af selve programmet, samt danne et bedre overblik. Til at supplere dette, er der oprettet en **makefile**, som hjælper med at kompilere C filerne til ét samlet program, i modsætning til at skulle gøre dette manuelt hver gang koden skulle omdannes til en eksekverbar fil.

For at skabe et læsbart flow, fokuserer afsnittet ikke på de individuelle filer og deres indhold. I stedet vil implementeringen af løsningen blive forklaret i samme rækkefølge, som den forekommer i programmet (Bilag D).

4.3.1 Kode standard

For at skabe overblik samt en ensartethed og kontinuitet i koden, blev der sat regler for programmeringen umiddelbart før arbejdet på implementering blev påbegyndt. Det vil sige at der blev indgået aftaler imellem gruppemedlemmerne angående standarder for variabelnavngivning, indrykning mm.

Aftalerne blev som følger:

Alle gruppemedlemmer skulle...:

- ...gå metodisk til værks. Mere specifikt arbejdes der efter idéen om top-down programmering med trinvis forfinelse.
- ...altid benytte curlybrackets ({}) i forbindelse med if/else-sætninger, loops osv.
- ...bruge fire spaces som indrykning.
- ...skrive identifiere og kommentarer på engelsk.
- ...bruge camelCase til funktioner, snake_case til variabler, CAPS til symbolske konstanter og følgende typecastede form til structsnavne: "Struct_navn_S".
- ...gruppere funktioner og prototyper efter funktionalitet, så funktioner som er relevante til hinanden er nemme at overskue.
- ...vælge gode variabel- samt funktionsnavne.

Endvidere er det vigtigt i dokumentationen af koden, at der er kommenteret på en fornuftig måde. Dette indebærer en beskrivelse af, hvad hver funktion overordnet gør. Hvis input- og outputværdierne til en funktion kræver yderligere forklaring, kommenteres dette også. For at dette gøres så optimalt som muligt, er der benyttet et dokumentationsprogram; Doxygen. Denne dokumentation findes i bilag C.

4.3.2 Main

Vores overordnede struktur af programmet er indkapslet i funktionen **main()** som ligger i **Program.c**. I funktionen kan man danne sig et overordnet overblik over flowet i programmet.

De erklæringer som ses på linje 4-9 i **main()** på snippet 4.1 er uinitialiserede men vil få assignet værdier senere i programmet. Variablerne er alle selvdefinerede structs eller arrays af selvdefinerede structs. Definitionerne på disse ligger i **edu_data.h**.

Originalt lå disse arrays som globale variabler, men dette er ikke attraktivt af flere årsager: Selvom det gør det nemt at tilgå værdierne alle steder i programmet, vurderes det, at det er god programmering at parameteroverføre til de relevante funktioner, så man kan se, hvilke data der er nødvendigt for funktionerne. Ved at benytte parameteroverførsel undgås det også, at man utilsigtet tilgår variable forkerte steder i programmet og derved formindskes muligheden for fejl.

Efter erklæringerne vil flowet meget overordnet se ud som følger:

- Print af introduktion til brugeren
- Generér alle structs ud fra en CSV-fil med data fra UG.dk
- Prompt brugeren for input
- Sorter i data ud fra brugerens input
- Print det mest optimale valg af andenprioritet til brugeren

På snippet 4.1 ses det hvordan ovenstående ser ud i programmet.

```
1 int main(void)
2 {
3     /* Arrays for structs */
4     Region_S regions[NUMBER_OF_REGIONS];
5     City_S cities[NUMBER_OF_CITIES];
6     University_S universities_denmark[NUMBER_OF_UNIVERSITIES_DENMARK];
7     Edu_S *educations = mallocArr(NUMBER_OF_EDUCATIONS * sizeof(Edu_S)); /* Could not be
        allocated statically */
8     User_input_S input; /* An encapsulation of the users preferences */
9     Edu_S *second_prio; /* A pointer to the second priority choice that should be determined */
10
11     printIntro();
12     generateAllStructs(educations, universities_denmark, regions, cities);
```

```
13     input = runAllPrompts(educations, cities, universities_denmark);
14     second_prio = findSecondPrio(input);
15     printRes(second_prio);
16
17     free(educations);
18     freeStrArr(input.wanted_cities);
19
20     return EXIT_SUCCESS;
21 }
```

Snippet 4.1. Funktionen `main()`

4.3.3 Generelle funktioner

Mens mange funktioner i programmet bliver anvendt med specifikke formål, er der også lavet generelle funktioner, som har til formål at blive genbrugt i flere sammenhænge. Det er blevet valgt at samle disse funktioner i en separat C-fil kaldet **A320lib.c**, da funktionerne går igen i C-filerne **Program.c**, **prompts.c**, **edu_data.c** og **sort.c** eller blot kan anses som værende generelt brugbare. Selvom navnet bærer præg af det, er filen ikke oprettet som et rigtigt bibliotek til C, men dette kunne være en videreudviklings mulighed. Funktionerne har altså ikke direkte relevans til opgaverne, som løses af de andre C-filer, men de anses alligevel som værende yderst brugbare i forskellige situationer. For at have en forståelse for disse funktioner og deres brug i de kommende afsnit, vil dette afsnit forklare hvilke funktioner der er tale om, samt hvordan de fungerer.

char *callocStr(const int length)

Funktionen som ses i snippet 4.2 er en simpel funktion, der dynamisk allokerer hukommelse til en tekststreng. Den forventer at få længden af tekststrengen overført som parameter, hvorefter den benytter sig af stdlib-funktionen **calloc()** til at allokere nok hukommelse til strengen. Her bemærkes det, at der er sørget for at allokere en ekstra byte til arrayet, for at sikre plads til karakteren `'\0'`, som er C-konventionen for afslutningen af en tekststreng. For god ordens skyld, tjekker funktionen også, om der er plads i lageret til at allokere den ønskede hukommelse til strengen. Hvis der ikke er plads udskrives en passende fejlmeddelelse, hvorefter programmet afsluttes. Funktionen returnerer en pointer til det første element i arrayet.

Denne funktion ligner meget en anden af de oprettede funktioner i **A320lib.c**, **mallocArr()**, som også allokerer hukommelse til et array, forskellen er blot at dette gøres med stdlib-funktionen **malloc()** i stedet for **calloc()**, og bytes i dette array vil derved ikke blive nulstillede.


```
1 char *callocStr(const int length)
2 {
3     char *arr;
4
5     arr = calloc(length + 1, 1); /* +1 to make sure there's space for '\0' */
6     if (arr == NULL)
7     {
8         puts("FATAL ERROR! callocStr could not allocate the requested memory.\nThe program will
9             now exit.");
10        exit(EXIT_FAILURE);
11    }
12    return arr;
13 }
```

Snippet 4.2. Allokering af plads til en tekststreng med `calloc`

`int clearInputBuffer(void)`

Funktionen fra snippet 4.3 tømmer inputbufferen, hvilket betyder, at den skiller sig af med uønskede tegn, som brugeren kunne have indtastet. Et eksempel kunne være, at der er scannet for en `int`, men brugeren har indtastet en tekststreng i stedet. Denne tekststreng kunne f.eks. være "abe". Funktionen vil i denne forbindelse kunne benyttes til at skille sig af med strengen, således at det undgås, at der fortsættes i koden med uønsket data i inputbufferen. Funktionen returnerer 1, hvis den formår at køre while-loopet færdigt.

```
1 int clearInputBuffer(void)
2 {
3     while (getchar() != '\n');
4
5     return 1;
6 }
```

Snippet 4.3. Funktionen `clearInputBuffer()`

`char *promptForString(const int length)`

Formålet med funktionen, som ses i snippet 4.4, er at indlæse brugerens input som en tekststreng. Tekststrengen bliver lagret i et dynamisk allokeret `char`-array, med hjælp fra `callocStr()` funktionen. Pointeren til arrayet bliver returneret af funktionen, så den kan tilgås andre steder i programmet. Det er her værd at bemærke, at tekststrengen i virkeligheden er et dynamisk allokeret array. Derfor bør lageret frigøres igen med `stdlib`-funktionen `free()` når strengen ikke længere skal bruges.

```

1 char *promptForString(const int length)
2 {
3     char *string_to_return = callocStr(length);
4
5     scanf(" %[^\n]", string_to_return);
6     clearInputBuffer();
7
8     return string_to_return;
9 }

```

Snippet 4.4. Funktionen **promptForString()**

char **callocStrArr(const int str_amount, const int str_len)

Som det ses på linje 3, 4 og 5 i snippet 4.5 bliver der indledningsvis i funktionen erklæret tre lokale variable:

- Variablen *str_amount_freeable* er en heltalsvariabel, som sørger for at der bliver allokeret en ekstra plads i arrayet. Denne plads vil senere blive assignet en speciel char, som gør det muligt at frigøre den allokerede hukommelse.
- Variablen *i* er ligeledes en heltalsvariabel. Dette er blot en counter, som skal bruges i for-løkken på linje 7 i snippet 4.5.
- Variablen *arr* er det array som returneres, og skal ses som værende et array af tekststreng. Arrayet initialiseres med **mallocArr()**, så der bliver allokeret hukommelse til de pointere, som senere skal pege på tekststrengene i arrayet.

Efter erklæringerne bliver der på linje 7 i snippet 4.5 iterativt allokeret hukommelse til hver tekststreng i en for-løkke vha. **callocStr()**.

Det som gør, at *callocStrArr()* kan bruges sammen med **freeStrArr()**, er det, som sker på linje 12 i snippet 4.5. På linje 12 i funktionen bliver der allokeret hukommelse til en enkelt char på det sidste index i arrayet med **mallocArr()**. På efterfølgende linje bliver denne char sat til `'\b'`, som er en non-printable char der symboliserer et backspace. Denne char vil være den, som funktionen **freeStrArr()** kigger efter, for at vide at den er færdig med at frigøre alle tekststrengenes hukommelse. Denne char blev valgt, da det er usandsynligt at en tekststreng vil starte med denne char.

Funktionen vil returnere en pointer til det første element i arrayet, hvorfor man bl.a. vil kunne tilgå hver tekststreng i arrayet med notationen:

```
1 array_navn[index];
```

Hvis man foretrækker pointeraritmetik, kan følgende notation benyttes:

```
1 array_navn + index;
```

```
1 char **callocStrArr(const int str_amount, const int str_len)
2 {
3     int str_amount_freeable = str_amount + 1, /* +1 to be able to free the array again */
4         i;
5     char **arr = mallocArr(str_amount_freeable * sizeof(char *));
6
7     for (i = 0; i < str_amount; i++)
8     {
9         arr[i] = callocStr(str_len);
10    }
11    /* Readies array to get freed */
12    arr[i] = mallocArr(1);
13    arr[i][0] = END_OF_ARR;
14
15    return arr;
16 }
```

Snippet 4.5. allokering til 2D array med calloc

void freeStrArr(char **arr)

Denne funktion går hånd i hånd med **callocStrArr()** præcis ligesom stdlib-funktionerne **calloc()** og **free()** går hånd i hånd. Som det ses i snippet 4.6 frigør funktionen indledningsvis iterativt den hukommelse, som er allokeret til tekststrengene, indtil den møder '\b'. Så snart funktionen møder denne char bryder den ud af loopet. Afslutningsvis frigøres den hukommelse, som var allokeret til arrayet der pointede til tekststrengene.

```
1 void freeStrArr(char **arr)
2 {
3     int i = 0, done = FALSE;
4
5     while (!done)
6     {
7         if (arr[i][0] == END_OF_ARR)
8         {
9             done = 1;
10        }
11        free(arr[i]);
12        i++;
13    }
14    free(arr);
15 }
```

Snippet 4.6. Frigørelse af allokeret plads

4.3.4 Datastruktur

Hele programmet bygger på en lang række data omkring uddannelser, som skal behandles for at nå frem til en 2. prioritetsuddannelse. Dette afsnit vil beskæftige sig med at beskrive, hvordan denne data er organiseret (en større fil fra UFM), og grundlæggende hvordan en uddannelse modelleres til en passende datastruktur.

CSV

Størstedelen af programmets data er organiseret i en CSV-fil fra Uddannelses- og Forskningsministeriet (2018a), som indeholder: Titel på uddannelse, type uddannelse, hovedinstitutionen, institutionskommune, undervisningsform og lignende uddannelser.

Der kan ses et eksempel på ovenstående i figur 4.2

	A	B	C	D	E	F	G	H	I	J
1	Titel	displaydocclass	hovedinsttx	instkommunetx	fagligmiljo_likert	socialtmiljo_likert	undervisningsform_p1	undervisningsform_p2	lignende_udd1	lignende_udd2
2	Naturressourcer	Bacheloruddannelse	Copenhagen Business school	Frederiksberg	4,3	1,1	Selvstudie	Gruppearbejde	Miljøvidenskab	Environmental biology
3	Kemi	Bacheloruddannelse	Danmarks tekniske universitet Lyngby - Taarbæk		1	4,3	Gruppearbejde	Forelæsninger	Fysik og teknologi	Farmaci
4	Matematik	Bacheloruddannelse	Aalborg universitet	Aalborg	3,1	2,7	Forelæsninger	Selvstudie	Datalogi	forsikringsmatematik
5	Dansk	Bacheloruddannelse	Københavns Universitet	København	0,5	4,1	Projektarbejde	Gruppearbejde	Engelsk	Filosofi
6	Jura	Bacheloruddannelse	Aarhus universitet	Aarhus	2,1	4	Selvstudie	Gruppearbejde	Statskundskab	Samfundsvidenskab

Figur 4.2. En simpel udgave af CSV-filen.

Det er vigtigt at bemærke, at CSV-filen ikke ser ud som på figur 4.2, men at dette blot er en simplificering, da den originale fil indeholder for mange kolonner til at indsætte. Hele CSV-filen kan findes i bilag D. Datafilen fra UFM havde originalt ikke en række med lignende uddannelser, hvilket derfor er blevet skrevet ind manuelt. Det har skabt nogle komplikationer med muligheder for indtastningsfejl, hvilket skaber problemer, hvis der bruges en `strcpy()` eller `strcmp()` på en værdi, som er indlæst fra CSV-filen.

En uddannelse som datastruktur

Det væsentligste data i arbejdet med at foreslå en 2. prioritetsuddannelse er selve uddannelsen. Dette ønskes derfor modelleret i en passende datastruktur. En indkapsling af informationer, der alt sammen siger noget om en uddannelse. I C giver det, efter vores overbevisning, bedst mening at gøre dette i et struct, hvilket kan opfattes som et objekt, der indeholder data som kan være af forskellige typer. I programmet er datatypen blevet navngivet "`Edu_S`", som er en typecastet version af "`struct Edu_S`".

Edu_S

Dette struct indeholder en uddannelse og alt det data som er relevant for uddannelsen. Hvilke datapunkter, der er relevante er blevet fastlagt som følge af problemanalysens resultater. Det er navn, uni niveau, gruppearbejde/ selvstudie, studieby, universitet, region, miljø, lignende uddannelser, løn, jobmuligheder, gennemsnit, arbejdsløshed og URL til uddannelsen på UG.

Definitionen på structen kan ses i snippet 4.7, hvor den på linje 1 bliver typecastet, så den efterfølgende kan refereres til som *Edu_S*. Der bliver også refereret til *Edu_S* structet på denne måde i rapporten. Det kan ses at structet indeholder medlemmer af forskellige typer. Heriblandt medlemmer af typen int, float, tekststreng (char*) og pointere til andre structs (struct Edu_S*).

Som tidligere beskrevet i afsnit 4.3.2 skabes der et array af *Edu_S* som en lokal variabel i **main()**, da der eksisterer mange instanser af en *Edu_S*. Det ønskes at lave en unik *Edu_S* der modsvarer alle de uddannelser der er i Danmark. I projektet er der foretaget en afgrænsning i, at der kun er indlæst data om bacheloruddannelser, og dermed ikke f.eks. professionsbachelor.

I forbindelse med pointere til andre structs, er *similar_educations* lidt speciel, da det er et array af *Edu_S* pointere til *Edu_S*. Dog kræver det, at alle *Edu_S* er genereret før der kan sættes pointere til de rigtige adresser.

På linje 10 i snippet 4.7 ses et medlem af typen *Edu_S**. Dette medlem er tænkt som et array fyldt op med pointers til andre uddannelser, som ifølge UG.dk ligner den pågældende uddannelse.

```
1 typedef struct Edu_S
2 {
3     char name[EDU_NAME_MAX_LENGTH];
4     char uni_level[UNI_LEVEL_MAX_LENGTH];
5     int group;
6     City_S *city;
7     University_S *uni;
8     Region_S *region;
9     Environment_S Environment;
10    struct Edu_S *similar_educations[MAX_SIMILAR_EDUCATIONS];
11    int salary;
12    char jobs[JOBS_IN_CSV][JOB_NAME_MAX_LENGTH];
13    double avg_grade;
14    int unemployment;
15    char url_to_education[URL_MAX_LENGTH];
```

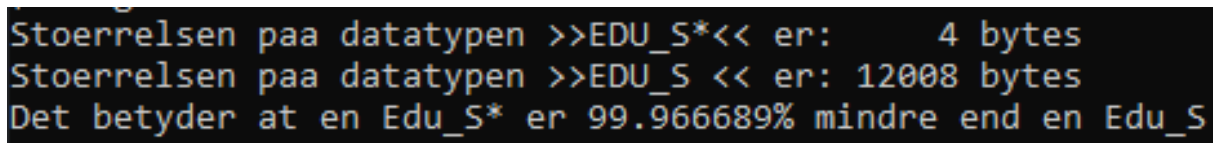
```
16 } Edu_S;
```

Snippet 4.7. Structet `Edu_S`

Der er flere begrundelser for at have et array af pointers til andre `Edu_S` structs, den første er at programmet ikke vil fungere, hvis et struct har felter af samme type som structet selv. Derudover kræver det mindre computerkraft at flytte et array af pointere rundt, frem for et array af vores `Edu_S` structs. Dette skyldes at en `Edu*` er størrelsesmæssigt 99,97% mindre end en hel `Edu_s`. Dette er illustreret på figur 4.3 med følgende print-statement som grundlag for outputtet:

```
1 printf("Stoerrelsen paa datatypen >>EDU_S*<< er: %5d bytes\n"
2       "Stoerrelsen paa datatypen >>EDU_S << er: %5d bytes\n"
3       "Det betyder at en Edu_S* er %lf%% mindre end en Edu_S\n\n",
4       sizeof(Edu_S*), sizeof(Edu_S), ((double)sizeof(Edu_S) - (double)sizeof(Edu_S*)) /
        sizeof(Edu_S) * 100);
```

Snippet 4.8. Print-statement som er grundlag for printet som ses på figur 4.3



```
Stoerrelsen paa datatypen >>EDU_S*<< er: 4 bytes
Stoerrelsen paa datatypen >>EDU_S << er: 12008 bytes
Det betyder at en Edu_S* er 99.966689% mindre end en Edu_S
```

Figur 4.3. Print som illustrerer størrelsesforskellen på `Edu_S*`ere sammenlignet med hele `Edu_S` structs

Udover `Edu_S` ses structet `Environment_S` som er en indkapsling af de forskellige Likert-skala værdier, som hvert enkelt uddannelse indeholder. Likert-skala værdien i programmet er taget fra CSV-filen i form af værdier fra 1-5. Disse værdier kommer fra spørgsmål med 5 svarmuligheder. Dette kunne for eksempel være spørgsmålet: "Har du oplevet at føle dig ensom på studiet?" Hvortil man får følgende svarmuligheder:

1. Altid
2. Ofte
3. Nogle gange
4. Sjældent
5. Aldrig

Hvilket `Environment_S` indeholder med flere forskellige spørgsmål. `Environment_S` indeholder derudover også structet `Teaching_methods_S`, som indeholder den foretrukne arbejdsmetode.

Her er et code snippet af `Environment_S`:

```
1 typedef struct Environment_S
2 {
```

```

3     double academic_env_likert;
4     double social_env_likert;
5     double engagement_teachers_likert;
6     Teaching_method_S teaching_method;
7     double loneliness_likert;
8     double stress_daily_likert;
9     double comfort_likert;
10 } Environment_S;

```

Snippet 4.9. Structet Environment_S

4.3.5 Dataindlæsning

Da alle elementerne i et **Edu_S** struct ikke skal indskrives manuelt i programmet, er der brug for at læse dataene fra Uddannelsens og forskningsministeriet direkte ind fra en CSV-fil. Dette kan gøres ved brug af standard library funktioner i C, **fgets()** og **sscanf()**, som indkapsles i en **getDataFromLine()** funktion, som kan ses på snippet 4.10.

```

1 void getDataFromLine(FILE *readFilePointer, char *url, char *name, char *uni_level, char *uni,
   char *city, char *region, Environment_S *Environment, int *unemployment)
2 {
3     char file_row[LINE_MAX_LENGTH];
4     double academic_env_likert, social_env_likert, engagement_teachers_likert;
5     char teaching_prio1[TEACHING_METHOD_MAX_LENGTH] = STR_NOT_SET,
6         teaching_prio2[TEACHING_METHOD_MAX_LENGTH] = STR_NOT_SET,
7         teaching_prio3[TEACHING_METHOD_MAX_LENGTH] = STR_NOT_SET,
8         teaching_prio4[TEACHING_METHOD_MAX_LENGTH] = STR_NOT_SET,
9         teaching_prio5[TEACHING_METHOD_MAX_LENGTH] = STR_NOT_SET;
10    double loneliness_likert, stress_daily_likert, comfort_likert;
11    academic_env_likert = 0.0;
12    social_env_likert = 0.0;
13    engagement_teachers_likert = 0.0;
14    loneliness_likert = 0.0;
15    stress_daily_likert = 0.0;
16    comfort_likert = 0.0;
17
18    fgets(file_row, LINE_MAX_LENGTH, readFilePointer);
19    sscanf(file_row, "%[^;]; %[^;]; %[^;]; %[^;]; %[^;]; %[^;]; %lf; %lf; %lf; %[^;]; %[^;];
   %[^;]; %[^;]; %lf; %lf; %lf; %d;",
20    url, name, uni_level, uni, city, region,
21    &academic_env_likert, &social_env_likert, &engagement_teachers_likert,
22    teaching_prio1, teaching_prio2, teaching_prio3, teaching_prio4, teaching_prio5,
23    &loneliness_likert, &stress_daily_likert, &comfort_likert, unemployment);
24    *Environment = updateEnvironment(academic_env_likert, social_env_likert,
25    engagement_teachers_likert, teaching_prio1, teaching_prio2, teaching_prio3,
26    teaching_prio4, teaching_prio5, loneliness_likert, stress_daily_likert, comfort_likert);
27 }

```

Snippet 4.10. Dataindlæsning fra CSV-fil

Da en csv-fil er en filtype, hvor alt data er semikolonsepareret, er det nemt at iterere igennem alle uddannelser og få dem indlæst i arrayet af *Edu_S*.

Funktionen **getDataFromLine()** (se snippet 4.10), der indlæser data fra den eksterne fil, leverer værdierne tilbage gennem outputparametre til funktionen **generateEducations()** (se snippet 4.11), som udover at kalde **getDataFromLine()** for hver linje i filen, også kalder **makeStruct()**. Denne assigner værdier til hver *Edu_S* i arrayet. Det er sådan, at værdierne i arrayet af bacheloruddannelser bliver initialiseret. I en bedre udgave af programmet ville det ikke være attraktivt at have et statement, som tog så mange forskellige input i et udtryk, som kan ses i **scanf()**. Derimod ville en løkke være ideel i dette tilfælde. Denne løkke kan, i dette tilfælde, kaldes 18 gange hvor den tilføjer den indlæste data til det korresponderende indeks. Endvidere kan det også være fordelagtigt at have en funktion, der tæller antallet af semikoloner i filen. Ved at overføre denne værdi til en variabel, kan løkken køres indtil dette tal er nået og derved behøves der ikke ændres i antallet af løkker, hver gang noget nyt skal indlæses.

generateEducations ser således ud:

```

1 void generateEducations(Edu_S educations[], University_S universities_denmark[], Region_S
   regions[], City_S cities[])
2 {
3     int number_of_lines, salary = 0, unemployment, i;
4     char name[EDU_NAME_MAX_LENGTH], uni[URL_MAX_LENGTH], city[CITY_MAX_LENGTH],
5         region[REGION_MAX_LENGTH], url[URL_MAX_LENGTH], uni_level[UNI_LEVEL_MAX_LENGTH];
6     double avg_grade = 0.0;
7     Environment_S Environment;
8
9     char *filename = "data_files/db_ALL_EDUCATIONS.csv";
10    FILE *readFilePointer = openFileAfterHeader(&number_of_lines, filename);
11
12    for (i = 0; i <= number_of_lines; i++)
13    {
14        getDataFromLine(readFilePointer, url, name, uni_level, uni, city, region, &Environment,
15                        &unemployment);
16        educations[i] = makeStruct(url, name, uni_level, uni, city, region, Environment, salary,
17                                avg_grade, unemployment, universities_denmark, regions, cities);
18    }
19    rewind(readFilePointer);
20    linkSimilarEducations(number_of_lines, readFilePointer, educations);
21    fclose(readFilePointer);
22 }
```

Snippet 4.11. Dannelsen af structs, der indeholder alle uddannelser

Altså er der en kobling mellem de lokale variabler i **generateEducations()** og outputparametrene til **getDataFromLine()**, da disse føres direkte videre i **makeStruct()** for at skabe uddannelserne. Denne funktion, der skaber en struct, har den enkle opgave at

assigne værdier til en *Edu_S* og returnere denne. Dette er en funktion der kunne effektiviseres, da der returneres en hel struct i stedet for at opdatere arrayet af uddannelser direkte.

Det er dog ikke al data, om en uddannelse, som indlæses fra CSV-filen i første omgang. De sidste kolonner indeholder de lignende uddannelser, og der skal derfor jf. definitionen på en *Edu_S* være pointere til andre structs af typen *Edu_S*. Dette kan dog ikke gøres før, alle *Edu_S* er skabt. Det betyder at disse pointere skal initialiseres med værdien NULL, men bliver opdateret ved kald af **linkSimilarEducations()** funktionen. I arbejdet med en fil skal det også huskes at nulstille filpointeren, og at indlæse fra det rigtige sted i filen - som for lignende uddannelser vil befinde sig senere på linjen.

```

1 void linkSimilarEducations(int number_of_lines, FILE *readFilePointer, Edu_S educations[])
2 {
3     char similar_educations_2D[SIMILAR_EDUS_IN_CSV][EDU_NAME_MAX_LENGTH];
4     int i;
5
6     ignoreLine(readFilePointer);
7
8     /* For every line (education) the similar educations are found */
9     /* Pointers to the other educations are inserted */
10    for (i = 0; i <= (number_of_lines - IGNORE_LINES); i++)
11    {
12        findSimilarEducations(readFilePointer, similar_educations_2D);
13        insertPointersToEducations(i, number_of_lines, similar_educations_2D, educations);
14    }
15 }
```

Snippet 4.12. Funktionen linkSimilarEducations

Det første der sker i funktionen i snippet 4.12 er, at der bliver ignoreret en linje fra filen, som er headeren. Efterfølgende bliver der itereret over de forskellige linjer med henblik på, at indlæse navnene på de lignende uddannelser i et array af strings, som funktionen **findSimilarEducations()** sørger for. Dernæst skal der opdateres NULL-pointere for en given *Edu_S* vha. **insertPointersToEducations()**.

For hver uddannelse i CSV-filen bliver alle lignende uddannelser, som er fundet via **findSimilarEducations()**, iterativt gennemgået, hvor strengen bliver sammenlignet mod alle uddannelses navne i *Edu_S educations* arrayet. Hvis **strcmp()** returnerer '0', indsættes en *Edu_S* pointer i *similar_educations* arrayet på den pågældende uddannelse.

Det er muligt at have flere uddannelser med samme navn, derfor bliver *similar_edu_index* talt op, hver gang der bliver fundet et match.

Der bliver brugt en dobbelt for-løkke, da studiegruppen vurderede at dette er mere

overskueligt at læse i forhold til parameteroverførsel i en ny funktion, som iterativt skulle kaldes med samme parametre. Man kunne dog opnå et større abstraktionsniveau ved at indkapsle funktionaliteten af det inderste loop i en funktion, og dermed undgået den dobbelte løkke.

```
1 void insertPointersToEducations(int current_edu, int number_of_educations, char
    similar_educations_2D[][EDU_NAME_MAX_LENGTH], Edu_S educations[])
2 {
3     int current_similar_edu, loop_edu_index, similar_edu_index = 0;
4     for (current_similar_edu = 0; current_similar_edu < SIMILAR_EDUS_IN_CSV;
        current_similar_edu++)
5     {
6         for (loop_edu_index = 0; loop_edu_index < number_of_educations; loop_edu_index++)
7         {
8             if (strcmp(similar_educations_2D[current_similar_edu],
                educations[loop_edu_index].name) == 0)
9             {
10                educations[current_edu].similar_educations[similar_edu_index] =
                    &educations[loop_edu_index];
11                similar_edu_index++;
12            }
13        }
14    }
15 }
```

Snippet 4.13. Indsættelse af pointere til lignende uddannelser

4.3.6 Håndtering af brugerinput

Al interaktion med brugeren er indkapslet i filen **prompts.c**. Formålet med filens funktioner er at bede brugeren om en række præferencer, hvorefter bearbejdelsen af disse sker i en anden fil (se 4.3.7). Dataen vil blive indlæst i et struct kaldet *input* (se snippet 4.14), denne struct returneres til **main()** for at dataen kan bruges af resten af programmet. Filen skal derudover tjekke om den angivne uddannelse er indeholdt i databasen, samt om brugerens indlæste data er valid.

Al interaktion med brugeren foregår i denne fil, da det derved er muligt at tilføje et nyt parameter at måle på, uden at det ændrer noget ved resten af programmet. Dette er også gjort for at minimere brugerens involvering i programmet under beregninger, men at brugerinput først er indlæst og bearbejdet bagefter.

Struct User_input_S

Structet *User_input_S* har tre medlemmer.

- En pointer til det *Edu_S* struct som korresponderer med brugerens ønskede førsteprioritet
- Et array af strings som indeholder de byer som brugeren ønsker at studere i
- Brugerens foretrukne arbejdsmetode (gruppearbejde eller individuelt arbejde)

Structet kan ses på snippet 4.14.

```
1 typedef struct User_input_S
2 {
3     Edu_S *first_prio;
4     char **wanted_cities;
5     work_type work_type;
6 } User_input_S;
```

Snippet 4.14. Structet *User_input_S*

Det er valgt at structet skal indeholde en *Edu_S** i stedet for en hel *Edu_S* af samme årsager som beskrevet i afsnit 4.3.4. Udover det er det valgt at lade funktionen **runAllPrompts()** returnere en hel *User_input_S* struct i stedet for en pointer, da der ikke er den store størrelsesforskel på de to datatyper. Dette er illustreret med print-statementet på snippet 4.15.

```
1 printf("Størrelsen paa variabelen >> input<< af typen >>User_input_S << er: %5d bytes\n"
2       "Størrelsen paa variabelen >>*input<< af typen >>User_input_S*<< er: %5d bytes\n",
3       sizeof(input), sizeof(&input));
```

Snippet 4.15. Print statement som illustrerer størrelsesforskellen på en *User_input_S* og en *User_input_S**

Outputtet fra print-statementet fra snippet 4.15 kan ses på figur 4.4.

```
Størrelsen paa variabelen >> input<< af typen >>User_input_S << er:    12 bytes
Størrelsen paa variabelen >>*input<< af typen >>User_input_S*<< er:     4 bytes
```

Figur 4.4. Output fra print-statementet som ses i snippet 4.15

Det er vurderet at en størrelsesforskel på 8 bytes ikke er nok til, at der ville udarbejdes en løsning med at returnere en pointer i denne situation. Ulempen ved denne beslutning er dog, at hvis det i fremtiden bliver aktuelt at tilføje et medlem af stor størrelse til structen, så vil man være nødsaget til at revurdere denne beslutning.

**Edu_S *getAndCheckFirstPrio(Edu_S educations[], City_S cities[],
University_S universities_denmark[])**

Funktionen **getAndCheckFirstPrio()** er blandt promptfilens hovedfunktioner. Funktionen tager blandt andre en input parameter: *Edu_S educations[]*. Denne parameter med-

tages, da funktionen i sidste ende skal returnere en *Edu_S* pointer til adressen på første-prioritetsstructet. I funktionen bliver der kaldt yderligere hjælpefunktioner som **prompt-ForEdu()**, som indlæser tre tekststrengene, **wrongInput()**, der blot verificerer et tal og til sidst har den **findEducationIndex()**, der løber igennem alle uddannelser indtil den finder et match. Hvis den ikke finder et match returnerer den *WRONG_INPUT*, som afleder en fejlbesked.

Funktionen **getAndCheckFirstPrio()** kan ses på snippet 4.16.

```

1  Edu_S *getAndCheckFirstPrio(Edu_S educations[], City_S cities[], University_S
    universities_denmark[])
2  {
3      char edu_name[EDU_NAME_MAX_LENGTH], edu_city[CITY_MAX_LENGTH], edu_uni[UNI_MAX_LENGTH];
4      int education_index = WRONG_INPUT, done = FALSE;
5      Pos_mistake_S pos_mistakes;
6      int number_of_lines = 0, enough_data = FALSE;
7
8      char *filename = "data_files/db_ALL_EDUCATIONS.csv";
9      openFileAfterHeader(&number_of_lines, filename);
10
11     /* validates input. If input is invalid prompt repeats */
12     do
13     {
14         pos_mistakes.edu_name = TRUE;
15         pos_mistakes.edu_city = TRUE;
16         pos_mistakes.edu_tag = TRUE;
17         promptForEdu(edu_name, edu_city, edu_uni);
18         education_index = findEducationIndex(edu_name, edu_city, edu_uni, educations,
            &pos_mistakes, number_of_lines);
19         if (education_index != WRONG_INPUT)
20         {
21             done = TRUE;
22         }
23         else
24         {
25             printErrorEduIndex(pos_mistakes, cities, universities_denmark);
26         }
27     } while (done == FALSE);
28     puts("OK. Education found.");
29
30     enough_data = checkIfEnoughData(&educations[education_index]);
31     if (enough_data == FALSE)
32     {
33         printf("\nWe don't have enough data on the education, sorry!\n");
34         exit(0);
35     }
36     printArrow();
37     return &educations[education_index];
38 }

```

Snippet 4.16. Indlæsning af førsteprioritet

work_type getAndCheckGroupWork(void)

Det sidste brugeren bliver spurgt om, er hvilken arbejdsform de foretrækker. Dette sker via funktionen **getAndCheckGroupwork()**. Her indtaster brugeren sin præference som en tekststreng. Herefter bliver denne tekststreng tildelt en enumeration som værdi. Dette sker, da det er nemmere at sortere/sammenligne med data fra CSV-filen. Funktionen tager ikke nogen parametre, men returnerer en enumeration type: -1, 0 eller 1, alt efter svaret. Der benyttes enumerations i stedet for heltal da det simplificere læsbarheden i koden. Først i funktionen køres en do-while()-løkke, som tjekker om brugerens input er korrekt. Dette tjek sker ved at benytte **strcmp()** funktionen. Herefter indlæses brugerens input, og når der findes et match tildeles variabelen *res* værdien for den valgte præference. Til sidst frigøres lagerpladsen for *answer* og værdien af *res* returneres til structet *input*:

```
1 work_type getAndCheckGroupWork(void)
2 {
3     char *answer;
4     work_type res;
5
6     do
7     {
8         puts("Do you prefer groupwork or individual work?\n"
9             "Enter 'group' for groupwork, 'individual' for individual work or 'both' if you
10             aren't fussed:");
11         answer = promptForString(WORK_TYPE_MAX_LENGTH);
12     } while (!correctGroupInput(answer));
13
14     if (strcmp(answer, "group") == 0)
15     {
16         res = group;
17     }
18     else if (strcmp(answer, "individual") == 0)
19     {
20         res = individual;
21     }
22     else if (strcmp(answer, "both") == 0)
23     {
24         res = both;
25     }
26     else
27     { /* Should not happen */
28         exit(EXIT_FAILURE);
29     }
30     printf("OK. All educations with work type \"%s\" will be taken into consideration.\n",
31         answer);
32     printArrow();
33     free(answer);
34     return res;
```

34 }

Snippet 4.17. Behandling af arbejdsform

4.3.7 Sortering og filtrering

Det første skridt, der skal tages, for at finde den bedste andenprioritet til brugeren er at samle de uddannelser, som betegnes af UG som værende lignende uddannelser. Dette håndteres ved at lave et array af pointers til 1. prioritetens lignende uddannelser, der vel at mærke, hver især er af typen *Edu_S*. Funktionen **makeEditableSimilarEduArray()** udfører denne opgave.

```

1 Edu_S **makeEditableSimilarEduArray(User_input_S input)
2 {
3     int i;
4     Edu_S **similar_edu_array = mallocArr(MAX_SIMILAR_EDUCATIONS * sizeof(Edu_S *));
5
6     for (i = 0; i < MAX_SIMILAR_EDUCATIONS; i++)
7     {
8         similar_edu_array[i] = input.first_prio->similar_educations[i];
9     }
10
11     return similar_edu_array;
12 }
```

Snippet 4.18. Dannelse af array af lignende uddannelser

Pladsen til arrayet af *Edu_S* pointere bliver allokeret inde i funktionen, det gøres for at overlade hele opgaven om at skabe arrayet af lignende uddannelser til denne funktion. Brugen af **mallocArr()** gør det vigtigt at frigøre den allokerede hukommelse igen, når den ikke længere skal anvendes, hvilket der også bliver taget højde for i programmet.

Filtrering baseret på brugerens valg af studieby

Den første reelle filtrering foretages via brugerens valg af studiebyer. Funktionen **filterSimilarEdusByCities()** tager brugerens *input*-struct ind som parameter, og derigennem har funktionen adgang til de byer som brugeren har angivet.

```

1 void filterSimilarEdusByCities(Edu_S **similar_edu_array, User_input_S input)
2 {
3     int i, similar_edu_is_in_wanted_cities = FALSE;
4
5     for (i = 0; i < MAX_SIMILAR_EDUCATIONS; i++)
6     {
```

```

7      if (similar_edu_array[i] != NULL) /* If there's a similar education logged on the index
      */
8      {
9          similar_edu_is_in_wanted_cities = checkIfSimilarEduIsInWantedCities(i,
          similar_edu_array, input);
10         if (!similar_edu_is_in_wanted_cities) /* If the similar education is not part of the
          wanted cities, remove it from array */
11         {
12             similar_edu_array[i] = NULL;
13         }
14     }
15 }
16 }

```

Snippet 4.19. Filtrering baseret på studiebyer

Selve filtreringen foregår ved, at de pointers, som peger på lignende uddannelser, der ikke findes i de ønskede byer, bliver sat til *NULL*. Hermed bliver de "fjernet" fra arrayet af lignende uddannelser, senere vil der blive taget højde for *NULL* pointerne. Det er en fordel blot at manipulere det allerede eksisterende array af pointere til lignende uddannelser, da det ikke er nødvendigt at optage yderligere plads til et nyt array.

Likert data

For endeligt at finde frem til den bedste 2. prioritet for brugeren, så sorteres arrayet via funktionen **qsort()**, således at den bedste uddannelse ligger på det første index. **qsort()** modtager **compareByNullAndLikert()** (snippet 4.20) som sammenligningsfunktion, og den funktion kalder i visse tilfælde funktionen **compareByLikertAverage()**, (se snippet 4.21).

```

1  int compareByNullAndLikert(const void *a, const void *b)
2  {
3      Edu_S *Edu_a = *(Edu_S **)a;
4      Edu_S *Edu_b = *(Edu_S **)b;
5
6      if (Edu_a == NULL)
7      {
8          return 1;
9      }
10     else if (Edu_b == NULL)
11     {
12         return -1;
13     }
14     else
15     {
16         return compareByLikertAvg(Edu_a, Edu_b);
17     }

```

18 }

Snippet 4.20. Sortering af NULL pointere og Likert værdier

```

1  int compareByLikertAvg(const Edu_S *Edu_a, const Edu_S *Edu_b)
2  {
3      double average_a = getEnvironmentAverage(Edu_a),
4          average_b = getEnvironmentAverage(Edu_b);
5
6      if (average_a < average_b)
7      {
8          return 1;
9      }
10     else if (average_a > average_b)
11     {
12         return -1;
13     }
14     else
15     {
16         return 0;
17     }
18 }
```

Snippet 4.21. Sammenligning på likertværdier.

Arrayet indeholder *NULL*-pointere som følge af de forudgående filtrerings funktioner, og de bliver sorteret bagerst i arrayet. Hvis to uddannelser, der skal sorteres, ikke er *NULL*, bliver de sammenlignet via gennemsnitsværdierne fra deres likert værdier. Fordelen ved at sortere arrayet, hvilket kræver meget computerkraft, er at det er mere fremtidssikret, i og med at der til enhver tid kan tilføjes flere sammenligningsfunktioner. Det kan blive aktuelt i tilfælde af, at der ønskes at sortere uddannelserne ud fra flere parametre end dem, som tages højde for i det nuværende program. En alternativ måde at finde frem til den bedste uddannelse, kunne være at gennemløbe de resterende uddannelser lineært, for at finde den uddannelse, der har den højeste gennemsnitlige likert værdi. Det ville være en hurtigere løsning, i forhold til computerkraft, men det vil ikke være ligeså fremtidssikret som den fremgangsmåde, der er blevet valgt til dette program.

De to første linjer i funktionen **compareByNULLAndLikert()**, (se snippet 4.9), deklarerer to *Edu_S* struct pointere, og initialiserer dem til værdierne fra parametrene *a* og *b*. Både *a* og *b* typecastes til typen *Edu_S** struct* og denne pointer-pointer bliver dereferenced for at nå ind til den pointer, som den yderste pointer peger på. Det er denne, inderste, pointer som bliver gemt i variablerne *Edu_a* og *Edu_b*. Formålet med at lave de to variable er, at det gør funktionen mere overskuelig, og man slipper for at typecaste *a* og *b*, hver gang de bruges, og i den følgende funktion, **compareByLikertAvg()**, hvor de bliver parameteroverført til.

```

1  int compareByNULLAndLikert(const void *a, const void *b){
```



```

2     Edu_S *Edu_a = *(Edu_S **)
3     Edu_S *Edu_b = *(Edu_S **)b;

```

Snippet 4.22. pointercast til en `Edu_S**`

På trods af, at det array som bliver sorteret allerede er af typen `Edu_S**`, når det bliver anvendt i `qsort()` funktionen, er det stadig nødvendigt at foretage et typecast på de to elementer, som bliver sammenlignet af sammenligningsfunktionen. Dette skyldes den måde `qsort()` er programmeret på, hvilket gør det nødvendigt for dens hjælpefunktion at modtage to parametre af typen `const void*`.

Håndtering af uddannelser med mangelfuld data

Desværre er dataen i den CSV-fil, hvorfra data om alle uddannelserne hentes, mangelfuld. Den mangelfulde data drejer sig om uddannelsens primære arbejdsmetoder eller Likert gennemsnittene omhandlende uddannelsens miljø. Disse uddannelser bliver stadig indlæst i programmet, men så snart programmet når til sorterings stadiet vil de blive filtreret fra. Det sker som følge af, at uddannelserne bliver fjernet, da den pointer, som peger på den pågældende uddannelse, bliver sat til `NULL`, hvis ikke uddannelsen indholder den arbejdsmetode som brugeren har indtastet. Alle uddannelser i filen kan altså genkendes og findes i det første stadie.

4.3.8 Output

Da formålet med programmet er at finde den bedst mulige 2. prioritet for brugeren, outputter programmet kun en enkelt uddannelse. Dette er med antagelsen, at programmet stadig har uddannelser tilbage i det endelige sorteringsarray, som bliver ændret baseret på data om det generelle studiemiljø. Det kan dog ske, at brugeren har valgt input som resulterer i, at ingen uddannelser er tilbagestående. Dette er en situation som er uundgåelig med nuværende design, da der kan forekomme uddannelser med meget få lignende uddannelser, som potentielt ikke passer til brugerens lokation og arbejdsorienterings præferencer. Som supplement er det ikke alle uddannelser i CSV-filen, som indeholder al den nødvendige data. Dette betyder at selvom uddannelser kan være i kategorien *lignende uddannelser*, bliver de ikke taget højde for. For at imødekomme et sådant scenario, har programmet to outputmuligheder, den bedst mulige 2. prioritet eller en fejlmeddelelse.

```

1 void printRes(Edu_S *second_prio)

```

```
2 {
3     if (second_prio != NULL)
4     {
5         printf("The best fit for your second priority is: %s in the great city of %s!\n"
6             "Before making your final decision please consider reading more about the
7             education on the following link:\n%s\n",
8             second_prio->name, second_prio->city->name, second_prio->url_to_education);
9     }
10    else
11    {
12        puts("No education was found using your chosen input.");
13    }
```

Snippet 4.23. Program output

Koden som printer resultatet til brugeren er vist i snippet 4.23, hvor funktionen **printRes()** tager en uddannelse som input, hvorefter den eftertjekker om variabelen er *NULL*. Med andre ord tjekker den om der er blevet fundet en endelig uddannelse. Hvis funktionen får overført en uddannelse og ikke en *NULL*-pointer, informeres brugeren om navnet på uddannelsen, beliggenhed, samt hvor man kan skaffe yderligere information omkring uddannelsen. Hvis en uddannelse ikke bliver fundet, informeres brugeren om, at der ikke kunne findes en uddannelse, baseret på det input brugeren har indtastet. Dette er et bevidst ordvalg, da det gerne skal undgås at påvirke brugerens beslutninger. Baseret på problemanalysen er de parametre programmet tjekker for vigtige i valget af 2. prioritet, hvilket betyder det ville være at modarbejde programmets formål, hvis programmet opfordrede brugeren til at prøve programmet igen, men denne gang med et bredere valg. Dette ville potentielt tvinge brugeren ud i, at vælge en uddannelse som ikke er optimal. Derfor blev det vurderet, at det ville være fordelagtigt, at programmet ikke foreslog, eller tog beslutninger, om noget der ligger uden for dets formål. Som opsummering er formålet med programmet at give den bedste mulige 2. prioritet, hvis det er muligt. Det er ikke formålet at give en 2. prioritet uanset hvad.

En sekundær vinkel på output er mængden af uddannelser, som bliver givet til brugeren. Samtidig med at det kan være muligt, at det endelige array af uddannelser er tom, er det også muligt at den indeholder flere uddannelser end en. Som nævnt i afsnit 4.3.7 bliver dette håndteret ved at sortere det endelige array. Dette er baseret på en score med 5 decimaler, hvilket gør det ekstremt usandsynligt for to uddannelser at have den samme score. Det kan dog forekomme og der er derfor i den sammenhæng gjort nogle overvejelser. Hvis to eller flere uddannelser ender med samme score, vil den outputtede uddannelse blive valgt tilfældigt. Et alternativ til dette ville være, at programmet outputter alle uddannelser der deler den bedste score, men formålet med programmet er, at outputte én enkelt 2. prioritet, hvilket betyder, at dette er en mulig fejl i programmet, på trods af at

risikoen for at den opstår er meget lille.

5 | Test

Test af programmet og dets funktioner er foregået løbende i takt med programmets udvikling. Oftest er der indsat en **printf()** funktion i de funktioner der blev arbejdet med. På den måde var det muligt at se, hvor langt i programmet man nåede inden den potentielle fejl, samt at se hvilket output eller input en pågældende variabel arbejdede med.

Det blev også besluttet efter ganske kort tid at der skulle benyttes en **makefile**, og sikre at der ikke var fejl i programmet inden der blev foretaget en overskrivelse/tilføjelse til programmet i den fælles kildekode. **Makefilen** har følgende compileroptions '-ansi', '-pedantic', '-Wall' og '-Wextra', som sikrer at der leveres så mange compile-time errors og warnings som muligt.

Årsagen til, at ovenstående blev besluttet var for at undgå fejl i vores kildekode.

5.1 Test af programmet

Der er blevet observeret en del fejl under udviklingen af programmet, herunder er to af de største 'bugs', som er blevet løst.

1. Alle uddannelser indeholdte pointere til lignende uddannelser, som ikke hørte til i den pågældende uddannelses '*similar_educations*' array. Softwarestudiet på AAU i Aalborg havde for eksempel pointere til samtlige lignende uddannelser som var helt forkerte, de bestod af Naturressourcer, Molekylærbiologi og Erhvervsøkonomi. Det kunne tænkes at dette var en fejl i CSV-filen men disse lignende uddannelser var ikke til at finde under Softwarestudiet.

Der blev lavet en **printf()** inde i funktionen, som læste CSV-filen, og her kunne det ses at variablerne medtog værdier fra tidligere indlæste uddannelser, årsagen til dette var, at variablerne blot var deklareret men ikke *NULL*/nul-initialiseret hver gang funktionen blev kaldt.

2. En anden problemstilling var at uddannelser uden læringsprioriteter og Likert-skalaer, blev medtaget trods filtreringen af lignende uddannelser. Et eksempel var softwarestudiet på AAU i Aalborg, som har Datalogi i Aalborg som en lignende uddannelse. Her blev Datalogi inkluderet som lignende uddannelse på trods af, at der ikke eksisterer data om arbejdsformen i datafilen fra Uddannelses og forskningsministeriet.

Årsagen til problemet var, at der aldrig blev kontrolleret om de lignende uddannelser specifikt havde *NULL*-værdier. Dette blev igen fundet ved at anvende funktionen **printf()** før og efter filtrerings-processen.

I programkoden findes også samtlige debugging funktioner, som hovedsageligt består af **print()**-funktioner hvor et specifikt element eller indeks bliver printet. Et eksempel kan ses på snippet 5.1, hvor funktionen har til opgave at udskrive en bestemt uddannelse, hvoriblandt alle structets members indgår.

```

1 void printEduStruct(Edu_S *education)
2 {
3     int j;
4
5     printf("The education is called: %s\n", education->name);
6     printf("Uni is: %s", education->uni->name);
7     printf(" (%s)\n", education->uni->tag);
8     printf("Uni level is: %s\n", education->uni_level);
9     printf("City is: %s\n", education->city->name);
10    printf("Region is: %s\n", education->region->name);
11    printf("Salary is: %d\n", education->salary);
12    printf("Group is: %d\n", education->group);
13    printf("Average Grade is: %f\n", education->avg_grade);
14    printf("The URL is: %s \n", education->url_to_education);
15    printf("Likert result for academic environment is: %f \n",
16           education->Environment.academic_env_likert);
17    printf("Likert result for social environment is: %f \n",
18           education->Environment.social_env_likert);
19    printf("Likert result for engagement of teachers is: %f \n",
20           education->Environment.engagement_teachers_likert);
21    printf("Method of teaching prio1: %s\n", education->Environment.teaching_method.Prio1);
22    printf("Method of teaching prio2: %s\n", education->Environment.teaching_method.Prio2);
23    printf("Method of teaching prio3: %s\n", education->Environment.teaching_method.Prio3);
24    printf("Method of teaching prio4: %s\n", education->Environment.teaching_method.Prio4);
25    printf("Method of teaching prio5: %s\n", education->Environment.teaching_method.Prio5);
26    printf("Loneliness likert: %f \n", education->Environment.loneliness_likert);
27    printf("Stress daily likert: %f \n", education->Environment.stress_daily_likert);
28    printf("Comfort likert: %f \n", education->Environment.comfort_likert);
29    printf("Amount of unemployment is: %d \n\n", education->unemployment);
30
31    printLine(50);
32}

```

```
30     printf("HERE ARE SOME SIMILAR EDUCATIONS TO %s:\n", education->name);
31     for (j = 0; j < 20; j++)
32     {
33         if ((education->similar_educations[j]) != NULL)
34         {
35             printf("Index is %d \n", j);
36             printf("Name is %s \n", education->similar_educations[j]->name);
37             printf("It is in %s \n", education->similar_educations[j]->city->name);
38             printf("You can take it at %s \n\n", education->similar_educations[j]->uni->name);
39         }
40     }
41 }
```

Snippet 5.1. lokalisering af fejl via printstatements

5.2 Alternative fremgangsmåder

At skrive en række **printf()**-statements indtil man finder fejlen, kan resultere i et meget ineffektivt resultat. Det ville i nogle tilfælde, have været langt mere effektivt, undervejs at have anvendt eksempelvis CuTest som testværktøj. Generelt er det god praksis at foretage systematiske test undervejs og dermed sørge for at funktionerne gør det rigtige, før de bliver implementeret i programmet. Optimalt set vil det betyde at fejlfinding på funktionniveau bliver unødvendigt. Dette indså projektgruppen først imod slutningen af projektet og er derfor ikke blevet implementeret, da det ikke ville være meningsgivende at igangsætte ved projektets afslutning.

6 | Diskussion

I udarbejdelsen af dette projekt og det dertilhørende C program har det været nødvendigt at træffe en række valg. Disse valg omhandler analysen af problemet som projektet bygger på, den programmeringsmæssige opbygning af programmet, samt hvilke funktionaliteter som programmet skulle inkludere. Det følgende afsnit er en diskussion af udvalgte overvejelser, der har ført til den endelige udformning af projektet og programmet.

6.1 Ét output i programmet

Et yderst betydningsfuldt aspekt af programmets funktionalitet er dets output. Hvilket er den uddannelse som programmet outputter som den bedste 2. prioritet for brugeren. Outputtet gjorde anlæg til en diskussion om, hvorvidt der blot skulle gives et uddannelsesforslag til brugeren, eller om der skulle gives flere uddannelsesforslag i tilfælde af at der var flere, som kvalificerede sig.

Som følge af at programmet sorterer i arrayet af lignende uddannelser, i sidste trin af filtreringsprocessen, frem for at filtrere dem fra, er det meget relevant at overveje, om programmet også skal levere foreslag til et potentielt 3. eller 4. prioritetsvalg. Det blev besluttet at programmet kun skal producere ét uddannelsesforslag til brugeren. Begrundelsen herfor er, at der ønskedes en vis simplicitet i programmet. Projektgruppen ønskede at lave et program, der løste ét problem, netop at finde én velovervejet og god uddannelse til brugerens 2. prioritet. Derfor skal brugeren ikke forvirres ved at blive foreslået flere uddannelser. I tilfælde af at programmet skal tjene et andet formål en dag, og f.eks. bruges til at finde flere uddannelsesforslag til brugeren, er programmet designet således, at det nemt kan omskrives til at give flere uddannelsesforslag som output. Hvis man prioriterer programmets effektivitet højere end fremtidssikringen af programmet, kan det være fordelagtigt ikke at sortere arrayet. I stedet kan man lineært søge igennem arrayet af uddannelser for at finde den uddannelse, der har den højeste Likert-skala værdi.

6.2 Sortering og filtrering

Baseret på problemanalysen er der fundet frem til de parametre, der har størst betydning for det gode studievalg. I det ideelle scenarie vil projektets program give et output baseret på alle de parametre. Desværre har projektets størrelse og ressourcermængder betydet en begrænsning for, hvilke data studiegruppen kunne få adgang til. Som følge heraf er der ikke taget højde for alle de aspekter. Ud fra den data som har været tilgængelig i projektet, er der taget en beslutning om at fokusere på uddannelsens indhold og de kompetencer som opnås gennem uddannelsen. Det vil være fordelagtigt at gå mere i dybden med uddannelsernes indhold i et projekt af større karakter. En ideel udvidelse af sorteringsalgoritmen ville være, at sammenligne uddannelsens kurser og studieplan. Herudover vil det også være meget relevant at anvende data angående beskæftigelse og jobmuligheder for de enkelte uddannelser, men det har dog ikke været muligt i dette projekt. På grund af projektets tidsramme er der truffet et valg om kun at filtrere uddannelserne baseret på enten arbejdsform, geografisk placering og lignende uddannelser. I de inkluderede CSV-filer er der yderligere data om uddannelserne, som også er indlæst i programmet, for at gøre det nemmere at sortere på flere elementer i en fremtidig udvikling af programmet.

6.3 “Nice-to-haves”

Da projektet har en bagkant, opstår der elementer, som ønskes implementeret, men som ikke kan nås. Derfor vil der i dette afsnit blive belyst nogle ændringer og forbedringer, som kunne have været attraktive, men som er blevet nedprioriteret.

Hashtabel

En hashtabel kunne effektivisere brugen af arrays ift. en langt mere effektiv og hurtig indeksering. En hashtabel kunne derfor have været en god måde at højne kvaliteten af programmet, men det burde have været implementeret i starten af projektet, da implementering af dette i slutningen af projektet ville kræve en større omstrukturering af koden.

Foreslå en uddannelse ved fejl i bruger input

For at kunne udvide vores prompt til eksempelvis at foreslå uddannelser som "Software", hvis brugeren f.eks. indtaster "soft", så burde der være lavet tre separate `scanf()`, som ville muliggøre at kalde andre funktioner imellem hvert prompt. Dette ville muliggøre, at man nemmere kunne fejlfinde problemet i hver enkel `scanf()` samt give foreslag til, hvad brugeren istedet skal indtaste. Som det ser ud i det færdige program, er brugerens input af førsteprioritet indkapslet i en enkelt `scanf()`, hvilket gør, at der kun kan fejlsøges på helheden efter prompten. At kunne foreslå korrekte navne på uddannelser som følge af fejlagtigt input, vil være en god brugervenlig feature i programmet. Et af programmets interaktionsmæssige faldgrupper er, at programmet er "case-sensitive", som følge af at karakterene *a* og *A* er leksikografisk forskellige. Da der benyttes `strcmp()` til at sammenligne programmets uddannelsesdatabase med brugerens input er det afgørende, om der anvendes store eller små bogstaver. At kunne indformere brugeren om, hvor der er fejl i det indtastede input vil gøre programmet mere brugervenligt.

Gøre groupwork mere nuanceret

Det færdige program sorterer kun ud fra om brugeren vil have gruppearbejde eller individuelt arbejde. Det kunne derfor være en mulighed at give brugeren lov til at vælge projektarbejde, forelæsninger mm. Dette er muligt, da denne data er indeholdt i CSV-filen fra UFM. Men dette data bliver ikke udnyttet i programmet på nuværende. Det kan til gengæld diskuteres, om dette er bedre for brugeren, siden det kan gøre processen mere forvirrende og mindre simpel.

Filtrere på jobmuligheder

Det var i starten tiltænkt, at programmet skulle kunne filtrere ud fra jobmuligheder og arbejdsløshedsrater efter endt studie. Det endte med, at programmet kun skulle håndtere bacheloruddannelser, hvilket gjorde at jobmuligheds- og arbejdsløshedsdata stort set ikke var tilgængeligt. En realisering af denne funktion var derfor så godt som umulig, men ud fra analysen kan det ses at det ellers er et vigtigt aspekt for det gode studievalg. Dette var blot ikke en mulighed pga. tiden der er sat af til projektet, da det ville have krævet en manuel indtastning af al data.

7 | Perspektivering

Studieprojektet har resulteret i et færdigt program, der løser sin opgave med ganske få mangler. Mangel på brugertest og de enkelte faldgrupper som programmet stadig indebærer gør dog, at programmet ikke er helt klar til at blive implementeret i det danske uddannelsessystem. I tilfælde af at der bliver lagt mere tid og flere ressourcer i udviklingen af programmet, antager studiegruppen, at programmet kan blive et supplerende værktøj til studievalgsprocessen. Det kan især antages, at programmet vil passe godt ind som en udvidelse til optagelse.dk, da denne platform allerede har indbygget data omkring uddannelserne i Danmark foruden information om ansøgeren. Den direkte adgang til data om både studierne og ansøgeren vil formentlig gøre, at programkoden kan optimeres, samt at det kan udelades at prompte brugeren for en 1. prioritet. Dette eliminerer den indledende fase i programmet, hvor brugeren indtaster et uddannelsesnavn og uddannelsessted, og hvor der foretages sammenligninger. I analysen blev det slået fast at mængden af værktøjer stillet til rådighed af UG godt kan virke uoverskuelig. Det er derfor ikke i studiegruppens interesse blot at tilføje endnu et værktøj til denne liste, i stedet vil det være at foretrække, hvis programmet kan blive en udvidelse til et allerede eksisterende værktøj.

8 | Konklusion

8.1 Opsamling

Gennem afgrænsninger i problemanalysen blev der i projektet sat et mål om at undersøge studerendes frafald på videregående uddannelser med særligt fokus på studerende, som havde fået tildelt deres 2. prioritetsvalg. Frafaldsraten viste sig at være højere på 2. prioriteten, selvom at mange studier har adskillige lignende uddannelser at vælge imellem. Så der burde kunne findes en 2. prioritet der ligner 1. prioriteten tilstrækkeligt. Resultatet af undersøgelsen var, at valget af 2. prioritet var knap så velovervejet som valget af 1. prioritet. Med et ønske om at reducere frafaldsraten blandt studerende der bliver optaget på deres 2. prioritet, blev følgende problemformulering udformet:

Hvordan kan der udvikles et program i C, som anbefaler en 2. prioritet til ansøgeren af en videregående uddannelse, på baggrund af den valgte 1. prioritet?

Dette lagde op til en løsning, der ville indebære ansøgerens 1. prioritet som input, samt et foreslag til 2. prioritet som output, hvor en uddannelse givetvis er en struct, der indkapsler en række data om uddannelsen. Problematikken vedrørende foreslag af en 2. prioritet ud fra inputtet krævede en præcisering og nedskalering af de betydningsfulde parametre, der indgår i studievalg. Parametre som studiemiljø og geografisk placering skulle afgrænses for at indgå i programmet, hvilket eksempelvis foregik ved at anvende Likert-skaladata til at sortere i uddannelserne.

Datagrundlaget i projektet har været en større CSV-fil (Uddannelses- og Forskningsministeriet, 2018a), som rummer informationer om videregående uddannelser i Danmark. De parametre der sorteres efter, er delvist begrænset af denne data. Det kunne være attraktivt at sortere på fagligt indhold i uddannelsen, men det kunne desværre ikke lade sig gøre i dette projekt. En mangel på data ved nogle uddannelser gør også, at der eksisterer uddannelser, hvortil der ikke kan findes et 2. prioritetsvalg i dette program.

8.2 Indfrielse af kravspecifikationer

Hvis man kigger på de opstillede kravspecifikationer (se afsnit 4.1), så overholder programmet det meste af det, som blev fastsat. Programmet kan finde alle bacheloruddannelser i Danmark, da disse indlæses fra CSV-filen. Der kunne sagtens udvides til at medtage professionsbacheloruddannelser også. De parametre der sorteres på baggrund af, stammer også fra data filen. Brugerinput bliver korrekt holdt op imod dataet fra databasen for at se om uddannelsen eksisterer. Forskellige uddannelsers studielokation og studiemiljøet bliver der også taget hensyn til i overensstemmelse med kravspecifikationerne, når uddannelser skal sammenlignes.

Et andet krav udformer sig i, at for enhver uddannelse skal der genereres det bedst mulige foreslag til 2. prioritetsvalget. Der er ikke blevet lavet brugertest i en skala, der kan sige noget omkring, hvorvidt programmet rent faktisk outputter nogle gode foreslag, da dette ville være for omfattende at måle på i dette projekts sammenhæng. Der er dog blevet sorteret i uddannelserne vha. projektets indsigt i, hvilke parametre, der menes at være vigtige for at give et kvalificeret foreslag til en 2. prioritet. Udover det så er det ikke alle uddannelser, som programmet har mulighed for at give foreslag til, da der er mangel på data ved visse uddannelser.

8.3 Opfyldelse af de faglige mål

Som et studieprojekt er der fastsat nogle mål for, hvad projektet fagligt skal give af udbytte. Et af de centrale mål er at analysere sig frem til et godt og velafgrænset problem, der kan løses vha. software. Det er projektgruppens vurdering, at projektets definerede problem opfylder det mål. At arbejde med frafaldsraten, særligt på 2. prioriteten, og årsagen til dette, er et velafgrænset problem.

Derudover er målet med projektet at udvikle et mindre program af høj kvalitet. Igennem implementationsafsnittet er der blevet vist nogle interessante funktioner, og der argumenteres for de egenskaber af programmet, der bidrager til kvaliteten. F.eks. overvejelserne omhandlende parametreoverførsel af hele structs i modsætning til pointere til structs. I afsnit 4.3 bliver det diskuteret, hvor effektive og kostbare løsningerne er, med henblik på forbedringsmuligheder. Herudover at programmet skrevet ud fra idéen om top-down programmering ved trinvis forfinelse, hvilket har resulteret i et overskueligt program, der er systematisk opbygget.

8.4 Forbedringsmuligheder

Der er blevet arbejdet på programmet op til deadline for projektafleveringen, men som beskrevet i diskussionen kunne både programeffektivitet og brugerinteraktion forbedres. Dette er i høj grad relateret til **strcmp()**-funktionen, som sammenligner to tekststrengene og ser, om de er ens. Denne funktion er krævende at skulle kalde mange gange, hvor en hash funktion kunne have været en god mulighed for at strukturere arrayet anderledes og derved gennemse arrayet ved hjælp af en unik indikator frem for navn. At tekststrengene skal skrives perfekt ind for at der findes et match, kan gøre, at brugeren må indtaste af flere omgange. Derudover er **strcmp()** case-sensitive og grammatisk kritisk, derfor kræves det, at brugeren indtaster sit input på perfekt vis, hvilket kan være problematisk for brugervenligheden. Dette kunne have været løst med nogle flere funktioner, som inputvaliderer og kan finde mere specifikke sammenhænge mellem input og databasens uddannelser, end der er i det nuværende program.

8.5 Implementeringskontekst

Hvis det udviklede program skulle bruges af ansøgere af videregående uddannelser, kunne det være en mulighed at tilføje programmet som en ny udvidelse til optagelse.dk. Dette eliminerer den indledende fase hvor brugeren indtaster et uddannelsesnavn og uddannelsessted, og hvor der i programmet foretages sammenligninger. I stedet vil dataet fra 1. prioriteten blive tilgængeligt med det samme uden dialog med brugeren. Optagelse.dk har formodentlig data om uddannelserne, samt om ansøgeren selv. Det kunne f.eks være ansøgerens gennemsnit, gennemførte fag mm. Dette kunne samtidigt sørge for, at programmet ikke blev endnu et "nyt værktøj" i den store samling af muligheder, hvilket studiegruppen ikke ønsker. I analysens afsnit 2.7 blev det iagttaget at UGs mange værktøjer kan virke uoverskuelige, og derfor kan der argumenteres for, at det vil være godt, hvis dette program kan fungere sammen med et allerede eksisterende værktøj.

Litteratur

Aarhus Universitet, 2019. Aarhus Universitet. *Skalatyper*.

<https://metodeguiden.au.dk/skalaer/skalatyper/>, 2019. Besøgt 05-11-2019.

Bessie Rauff, 2019. Bessie Rauff. *Findes det perfekte studie?*

<https://krifa.dk/inspiration/uddannelse/findes-det-perfekte-studie>, 2019.

Besøgt 12-11-2019.

Danmarks Evalueringsinstitut, 2018. Danmarks Evalueringsinstitut. *Studievalg og frafald på de videregående uddannelser*, Danmarks Evalueringsinstitut, 2018.

Edwards et al., 1990. Mike Edwards, Joseph P. Cangemi og Cash J. Kowalski. *The college dropout and institutional responsibility*. EBSCOhost, 111:1, s. 108–109, 1990.

Humlun og Jensen, 2010. Maria Knoth Humlun og Torben Pilegaard Jensen. *Frafald på de erhvervsfaglige uddannelser – Hvad karakteriserer de frafaldstruede unge?* AKF, 05, s. 35–47, 2010.

Koichi, 2014. Koichi. *WHY JAPANESE EDUCATION SUCCEEDS*.

<https://www.tofugu.com/japan/japanese-education/>, 2014. Besøgt 31-10-2019.

Lehmann, 2007. Wolfgang Lehmann. *“I just didn’t feel like I fit in”: The role of habitus in university drop-out decisions*. Canadian Journal of Higher Education, 37, s. 89–110, 2007. DOI: 10.1016/S0360-1315(02)00072-6.

Madsen, 2016. Tom Vindbæk Madsen. *Årsager til frafald*, Aarhus Universitet - Science and Technology, 2016.

Newby et al., 2009. Howard Newby, Thomas Weko, David Breneman, Thomas Johanneson og Peter Maassen. *OECD Reviews of Tertiary Education JAPAN*, OECD, 2009.

OECD, 2008. OECD. *How many students drop out of tertiary education*, OECD, 2008.

- Olsen, 2006.** Henning Olsen. *Guide til gode spørgeskema*. ISBN: 87-7487-812-3, Paperback. BookPartnerMedia A/S, 2006.
- Ritzau, 2018.** Ritzau. *20 procent flere falder fra på universitetet*. <https://www.dr.dk/nyheder/indland/20-procent-flere-falder-fra-pa-universitetet>, 2018. Besøgt 28-10-2019.
- Studenterrådet ved Aarhus universitet, 2000.** Studenterrådet ved Aarhus universitet. *DEL I: FRAFALD: DEFINITIONER OG TEORI*, Studenterrådet ved Aarhus universitet, 2000.
- Studentum, 2019.** Studentum. *Uddannelsestest/karrieretest - Hvad kan jeg blive?* <https://www.studentum.dk/EducationTest/Start/17>, 2019. Besøgt 08-11-2019.
- Sørensen og Ibsen, 2017.** Karen-Kathrine Vedel Sørensen og Lene Askholm Ibsen. *Frafaldsundersøgelse på Det Samfundsvidenskabelige Fakultet*, Antropologisk Analyse ved Institut for Antropologi, 2017.
- Tanggaard, 2013.** Lene Tanggaard. *An exploration of students' own explanations about dropout in vocational education in a Danish context*. Vocational Education & Training, 65:3, s. 422–439, 2013. DOI: 10.1080/13636820.2013.834956.
- Uddannelsens- og Forskningsministeriet, 2018.** Uddannelsens- og Forskningsministeriet. *Frafald og studieskift*. <https://www.altinget.dk/misc/Frafald%20og%20studieskift.pdf>, 2018. Besøgt 05-11-2019.
- Uddannelses- og Forskningsministeriet, 2018a.** Uddannelses- og Forskningsministeriet. *CSV-fil (Uddannelseszoom)*. <https://ufm.dk/uddannelse/statistik-og-analyser/uddannelseszoom>, 2018. Besøgt 09-12-2019.
- Uddannelses- og Forskningsministeriet, 2018b.** Uddannelses- og Forskningsministeriet. *Frafald blandt akademiske bachelorstuderende fra universiteterne inden for 4. år efter studiestart, jf. Danske Universiteters opgørelse*, Uddannelses- og Forskningsministeriet, 2018b.
- Uddannelses- og forskningsministeriet, 2017.** Uddannelses- og forskningsministeriet. *Statistik om SU til videregående uddannelser*. <https://ufm.dk/uddannelse/statistik-og-analyser/su/statistik-om-su-til-videregaende-uddannelser>, 2017. Besøgt 27-10-2019.

Uddannelses- og Forskningsministeriet, 2019. Uddannelses- og Forskningsministeriet. *Tilskud til uddannelse*. <https://ufm.dk/uddannelse/videregaende-uddannelse/universiteter/okonomi/uddannelsesstilskud>, 2019. Besøgt 08-11-2019.

Uddannelses- og forskningsministeriet, 2019. Uddannelses- og forskningsministeriet. *Tal og fakta om søgning og optag på de videregående uddannelser*. <https://ufm.dk/uddannelse/statistik-og-analyser/sogning-og-optag-pa-videregaende-uddannelser>, 2019. Besøgt 28-10-2019.

UddannelsesGuiden, 2019a. UddannelsesGuiden. *Overvejelser til dit studievalg*. <https://www.ug.dk/studievalg/studievalgoestjylland/10-overvejelser-til-dit-studievalg>, 2019. Besøgt 28-10-2019.

UddannelsesGuiden, 2019b. UddannelsesGuiden. *Alle værktøjer*. <https://www.ug.dk/inspiration/vejledningsvaerktoejer-0>, 2019. Besøgt 30-10-2019.

UddannelsesGuiden, 2019c. UddannelsesGuiden. *adgangskortet*. <https://www.ug.dk/videregaaendeuddannelse/adgangskortet>, 2019. Besøgt 31-10-2019.

UddannelsesGuiden, 2019d. UddannelsesGuiden. *jobkompasset*. <https://www.ug.dk/flereomraader/inspiration/miniartikler/jobkompasset>, 2019. Besøgt 31-10-2019.

UddannelsesGuiden, 2019e. UddannelsesGuiden. *Studievælgeren*. <https://www.ug.dk/videregaaendeuddannelse/studievaelgeren>, 2019. Besøgt 31-10-2019.

UddannelsesGuiden, 2019f. UddannelsesGuiden. *uddannelseszoom*. [https://ug.dk/vaerktoej/uddannelseszoom/#!/,](https://ug.dk/vaerktoej/uddannelseszoom/#!/) 2019. Besøgt 31-10-2019.

UddannelsesGuiden, 2019g. UddannelsesGuiden. *Prioritering*. <https://www.ug.dk/videregaaendeuddannelse/altomoptagelse/prioritering>, 2019. Besøgt 30-10-2019.

Bilag

Bilag A: SoTA-tabel

Se elektronisk bilag.

Bilag B: Det gode studievalg spørgeskemadata

Se elektronisk bilag.

Bilag C: Doxygen

Se elektronisk bilag.

Bilag D: Programmets kildekode

Al data forbundet med programmet er indeholdt heri.

Se elektronisk bilag.