



SUBMISSION OF WRITTEN WORK

Class code: BSGRPRO1KU

Name of course: Grundlæggende Programmering

Course manager: Claus Brabrand

Course e-portfolio:

Thesis or project title: MAX STREAM

Supervisor: Anders Clausen

BASTJAN ROSGAARD SEJBERG (base@itu.dk)

PERNILLE KREJBERG PETERSEN (pekp@itu.dk)

NICKLAS KRISTOFFER BRICHS JENSEN (nkrj@itu.dk)



GRUPPE 15

BASTJAN ROSGAARD SEJBERG [base@itu.dk]
NICKLAS KRISTOFFER BRICHS JENSEN [nkrj@itu.dk]
PERNILLE KREJBERG PETERSEN [pekp@itu.dk]

Afleveringsdato: 17. december 2021

Indholdsfortegnelse

| | |
|--|-----------|
| 1. Indledning | 4 |
| 2. Baggrund & Problemstilling | 4 |
| 2.1 Problemområde | 4 |
| 2.2 Brugermæssige krav | 4 |
| 2.3 Systemmæssige krav | 5 |
| 2.4 Datahåndtering i programmet | 5 |
| 3. Problemanalyse | 6 |
| 3.1 Designbeslutninger vedrørende programmets interne struktur | 6 |
| 3.2 Designbeslutninger vedrørende databasens struktur | 7 |
| 3.3 Designbeslutninger vedrørende brugergrænsefladen | 7 |
| 3.4 Udvidelser & forbedringer | 9 |
| 4. Brugervejledning | 11 |
| 4.1 Fejlmeddelelser og exceptions | 20 |
| 5. Teknisk beskrivelse af programmet | 20 |
| 5.1 Interne datastrukturer | 20 |
| 5.1.1 Objekt-klasser | 20 |
| 5.2 Interne algoritmer | 21 |
| 5.2.1 VideoDB | 21 |
| 5.2.2 ImageLoader | 21 |
| 5.3 Brugergrænsefladens struktur | 21 |
| 5.3.1 Profile | 21 |
| 5.3.2 Add Profile | 22 |
| 5.3.3 Main | 23 |
| 5.3.4 Main Movies / Series | 24 |
| 5.3.5 My List | 25 |
| 5.3.6 Show Movies / Series | 25 |
| 6. Afprøvning | 26 |
| 6.1 Afprøvning af programmet under udvikling | 26 |
| 6.2 Tidskrævende fejl opdaget gennem afprøvning | 27 |
| 7. Refleksion over arbejdsprocessen | 27 |
| 8. Konklusion | 27 |
| 9. Litteratur | 28 |
| 10. Bilag | 28 |
| 10.1 MOSCOW metode | 28 |
| 10.2 Verb/Noun metode | 29 |
| 10.3 Første mock-up i AdobeXD | 30 |
| 10.4 Kildekode til ImageLoader findPath-metode | 35 |
| 10.5 Unit Test af ImageLoaders findPath-metode | 36 |

1. Indledning

Denne rapport er udarbejdet i december 2021 i forbindelse med et 3-ugers projekt på baggrund af den givne projektbeskrivelse i forbindelse med kurset, Grundlæggende Programmering, på 1. semester af bacheloruddannelsen i Softwareudvikling på IT-Universitetet i København. I dette projekt har gruppen udarbejdet programmet 'Max Stream' der fungerer som et softwaresystem til en streamingtjeneste. Projektet er udarbejdet under vejledning af Anders Clausen.

2. Baggrund & Problemstilling

2.1 Problemområde

Streaming betyder at der hentes strømme af data, samtidig med det bliver afspillet disse bliver afspillet, så disse data ikke bliver lagret på enheden. En streamingtjeneste er en platform der kommunikere denne data videre til bruger. I dette tilfælde dækker denne data over film og serier, hvoraf serier er videre inddelt i sæsoner og episoder. Denne streamingtjeneste har til formål at danne en nemt tilgængelig grafisk brugergrænseflade, hvorfra en bruger kan tilgå en ønsket film eller serie. Brugeren kan tilgå siden, og oprette brugerprofiler. Herfra er det muligt at tilgå de to forskellige medietyper, film og serie. Hver især har disse 20 genretyper.

2.2 Brugermæssige krav

De brugermæssige krav til programmet fokusere på brugerens oplevelse og interaktion med programmet. De brugermæssige krav er hovedsageligt på baggrund af projektbeskrivelsen, og gruppens egne erfaringer med en række streamingtjenester.

| ID | Brugermæssige krav |
|----|---|
| 1 | Brugeren skal nemt og hurtigt kunne danne sig et visuelt overblik, og have forståelse for programmets funktioner. |
| 2 | Brugeren skal kunne finde informationer om en ønsket film eller serie i form af genre, IMDB-rating og årstal for udgivelse. |
| 3 | Brugeren kan oprette en brugerprofil, og skifte mellem eksisterende brugerprofiler. |
| 4 | Brugeren skal kunne søge efter en specifik film eller serie. |
| 5 | Brugeren kan filtrer indhold både efter film eller serie, men også inden for genre. |
| 6 | Brugeren skal kunne tilføje/slette en film eller serie til en liste på baggrund af brugerprofil. |
| 7 | Brugeren skal kunne danne sig et overblik sin brugerspecifikke liste. |
| 8 | Brugeren kan se et overblik over sæsoner og episoder for en given serie. |

Tabel 1 - Brugermæssige krav til programmet

Ud fra de ovenstående krav, hæves der tre fokuspunkter. Det første essentielle fokuspunkt i projektet er en intuitiv brugergrænseflade der giver brugeren forståelse for programmets funktionaliteter ud fra det visuelle aspekt. Årsagen til dette ligger til dels i, at der i projektbeskrivelsen bliver forklaret at det færdige produkt ikke skal kunne afspille medier. Det andet fokuspunkt er et ubesværet brugerflow gennem programmet, med transparente funktionaliteter. Det tredje, og sidste fokuspunkt er at give brugeren en oplevelse af en personlig platform med mulighed for brugerprofil, og en dertilhørende brugerspecifik liste.

2.3 Systemmæssige krav

De systemmæssige krav til programmet er valgt på baggrund af projektbeskrivelsen, og de givne retningslinjer i denne.

| ID | Systemmæssige krav |
|----|---|
| 1 | Programmet skal skrives i Java. |
| 2 | Programmet skal have lav kobling og høj sammenhæng. |
| 3 | Programmet skal have en grafisk brugergrænseflade. |
| 4 | Programmets brugergrænseflade skal skrives i Swing eller JavaFX. |
| 5 | Programmet skal kunne indlæse data de givne film og serier, som den skal behandle og håndtere i selve programmet. |
| 6 | Programmets vigtigste komponenter skal testes, og dokumenteres undervejs, evt. ved brug af unit test ¹ . |
| 7 | Programmet skal kunne køres på én computer. |

Tabel 2 - Systemmæssige krav til programmet

2.4 Datahåndtering i programmet

De data programmet skal håndtere er beskrevet nedenfor i tabel 3. Programmet læser al nødvendig data fra en database, hvorefter databehandlingen sker i selve programmet. Det er derfor et klienttungt system.

| Data | Beskrivelse |
|--------------|---|
| Film | 1. .jpg forsidebillede af alle film 2. .txt-fil med {navn; udgivelsesår; genre; IMDB-rating} |
| Serie | 1. .jpg forsidebillede af alle serier. 2. .txt-fil med {navn; kørselstid; genre; sæson, episode antal} |

Tabel 3 - Data der håndteres i programmet

¹ Barnes & Költing - Objects first with Java (2016) - side 325

3. Problemanalyse

Det første element af analysefasen var at tage stilling til de brugermæssige- og systemmæssige krav til programmet, herunder også forslag til ekstra features. Både disse givet i projektbeskrivelsen, og gruppens egne forslag. Med disse opstillede gruppen en tabel ved brug af MoSCoW-metoden² for at danne overblik over prioritering af elementer og features i projektet (Bilag 10.1). På baggrund af denne diskuterede gruppen applikations domænet. Dette blev gjort ved at identificere specifikke fænomener ved streamingtjenester, og abstrahere til koncepter. Her spillede verb/noun-metoden en betydelig rolle (Bilag 10.2). Denne er brugt som inspiration til klasser og metoder, men gruppen har dog videreudviklet op disse og kreeret egne termer for gennemgående elementer. Eftersom verb/noun-metoden kun er brugt til inspiration, er det valgt ikke at gå i dybden med Class-Responsibilities-Collaborator (CRC)-cards³.

3.1 Designbeslutninger vedrørende programmets interne struktur

Max Stream er gruppens løsningsforslag til dette projekt problemstilling. Programmet består af 17 klasser udviklet i Java, herunder en Unit-testklasse, 6 event-håndteringsklasser, 10 generiske klasser og en abstract klasse. Programmet indeholder også en brugergrænseflade, udviklet i JavaFX Scene Builder. Dette er bygget op omkring projektets brugermæssige krav (Tabel 1).

Projektmaterialerne der er givet til dette projekt bestod af en zip-fil med to .txt-filer, den ene med data og information om film, og den anden med data og information om serier, samt 100 .jpg-filer til hvert medie af film- og serieforside plakater (Tabel 3).

Det har under hele udviklingen af programmet været et mål om at opbygge programmets grundstruktur efter et Model-View-Controller (MVC) designmønster. Dette var for at opnå en så vidt mulig effektiv kode, for at have lav kobling og høj sammenhæng i fokus. Hvad omfatter redegørelse af model-klasser, har vi separeret opbevaring af kritiske og nyttige oplysninger gennem programmets køretid (lagt vægt på *køretid*; programmet gemmer derfor intet permanent efter nedlukning).

Blandt disse model-klasser er den mest åbenlyse *Profile*-klassen, komplementeret af *ProfileDB* (profildatabase): Her gemmes nødvendige oplysninger, bl.a. brugerens navn, farvevalg for profilbillede, alder og personlig liste af film og serier. Disse oplysninger er brugerspecifikke og tilhøre kun den bestemte profil. Ud over denne er der en abstract-klassen *Video*, hvori funktionelle metoder deklarereres, men også abstrakte metoder defineres. Herfra nedarver både model-klasserne *Series* og *Movie*, hvilket tillader bekvem organisering og håndtering af data. Afdelingen bag frontenden, altså *view*, er udelukkende bestående af fxml-filer for hver scene programmet indeholder. Disse definere udseendet, elementernes dimensioner, ID, og controlleren. Controller-elementet i programmet er knyttet op, til hver enkel fxml-fil, så der opnås minimal kobling i programmet og effektiv modularitet.⁴

² <https://www.productplan.com/glossary/moscow-prioritization/>

³ Barnes & Kölling - Objects first with Java (2016) - side 560

⁴ Næsten alle controllers nedarver fra *SceneController*, dog faciliterer dette navigeringen mellem scener i ens stage og har minimal kobling.

3.2 Designbeslutninger vedrørende databasens struktur

I programmet anvendes der ikke en ekstern database til opbevaring af nyttige data. Selvom en databasestruktur fremmer stærkere sikkerhed og effektiv adgang fra eksterne klienter, ville en database være overflødig for dette projekt. Dette er eftersom der er sat fokus på *must-haves* (efter MoSCoW-metoden) grundet begrænset tid.

I et scenarie hvor der er oprettet en database, og implementeret en lignende løsning i programmet, ville mere følsomme, private data opbevares på en database. Det er dog ikke tilfældet i den nuværende, klientbaserede løsning, hvor alle kan indhente vedkommendes brugerdata med minimalt besvær. Sikkerhed var dog ikke en prioritet i dette program, hvilket bestemt også er synligt med en manglende login-funktion.

I programmet opbevares data i såkaldte *DB-klasser* (databaseklasser) som f.eks. *VideoDB* og *ProfileDB*. Grundet deres natur som klasser, opbevarer de kun data i køretid. Det vil sige, at hvis programmet lukkes, mistes al data og skal dermed begynde på ny. Dette designvalg er taget eftersom gruppen ikke mente det var et afgørende element af projektet og krævede yderligere arbejde, med den begrænsede mængde tid.

Oprindeligt da projektet blev påbegyndt blev der undersøgt opbevaring af data i et dokument, hvorefter oplysningerne kunne indlæses derfra, men siden dette emne var dækket nogenlunde af med bl.a. "Scanner", blev dette hurtigt skrottet. Årsagen til dette var bl.a. også at dette ikke var et fastslået krav eller specifiseret i det udleverede materiale.

3.3 Designbeslutninger vedrørende brugergrænsefladen

For at danne et visuelt overblik over den ønskede løsning, blev der hurtigt udviklet et mock-up af brugergrænsefladen, kreeret i AdobeXD (Bilag 3). Dette mock-up havde til formål at give et indsig i hvordan brugergrænsefladen gerne skulle se ud for en kommende bruger. Derudover havde det til formål at diskutere hvordan gruppen på mest effektiv vis kunne implementere de besluttede elementer for at opfylde de brugermæssige krav (Tabel 1). Dette mock-up havde også til formål at danne rammer for opbygningen af GUI ved brug af Scene Builder via frameworkt, JavaFX.

I den generelle opbygning var der hovedsageligt fokus på 2 ting. Det første var at brugeren hurtigt skulle kunne danne sig et overblik over programmets funktioner på baggrund af det visuelle aspekt. Det andet var at opbygge streamingtjenesten med så få scener som muligt. Dette var både for at holde "low coupling, high cohesion" anbefalingeren, og for at skabe mindst mulig forvirring for en ny bruger af programmet.

For at opnå den første af de to ovennævnte fokuspunkter, er brugergrænsefladen udviklet med en fixet menulinje der altid ligger øverst i vinduet. Denne menu indeholder 5 knapper. Den første knap indeholder et .png med Max Stream logoet, og er også en 'Hjem'-knap, der tager brugeren tilbage til startsiden, hvorpå alle 200 film, og serier kan findes. Efter 'Hjem'-knappen følger tre knapper 'Movies', 'Series', 'My List', et søgefelt og en knap, formet som brugerprofil-farven, der tager brugeren til oversigten over brugerprofiler. Det tekniske aspekt af menulinjen vil blive yderligere gennemgået i afsnit 5.3.3.

Under menulinjen vises en anbefaling til brugeren. Anbefalingerne indeholder et billede fra den valgte film, en overskrift, og et kort resume af filmen.

Under anbefalingerne øverst på siden er der sat et tile pane op til visning af film- eller serieforside plakater der repræsenterer hvert medie i programmet. Dette er for at give et simpelt og symmetrisk overblik over tjenestens indhold. Dette er vist med en scroll-funktion, så der på main siden er mulighed for at scrollle ned gennem indholdet. Denne tile pane-funktion kan dog også skabe en forvirring hos brugeren, da det kan være svært at lokalisere den ønskede film eller serie, og dertil kan søgefunktionen i menulinjen, beskrevet ovenfor, benyttes. Denne tile pane-funktion bliver redegjort yderligere i afsnit 5.3.3.

Hver af de repræsenterede medier i form af film- eller serieforside plakater er en knap brugeren vil have mulighed for at klikke på, for at hente yderligere information om den valgte film. Denne side viser i tilfældet af en film informationer om navn, udgivelsesår, genre og IMDB-rating. I tilfælde af brugeren klikker på en serie viser den informationer om navn, kørsels år, genre, IMDB-rating og antallet af sæsoner og episoder. Visningen af medier bliver vist i den rækkefølge de står i txt-filen. En evt. mulighed for at sortere visningen af disse, f.eks. fra højeste til laveste IMDB-rating, er en rigtig god mulighed for udvidelse af dette program. Dette vil blive gennemgået yderligere i afsnit 3.4.

‘Movies’ og ‘Series’ scenerne er næsten identiske med ‘Hjem’. Dette er bl.a. for at give brugeren en følelse af en gennemgående brugergrænseflade, og for at undgå unødvendig forvirring. Forskellen ligger i at der under ‘Movies’, udelukkende befinner sig de 100 film, og at der under ‘Series’, udelukkende befinner sig de 100 serier. Herudover er der i disse scener også mulighed for at sortere i genre via. en dropdown menu. Altså kan der først vælges film, hvorefter der kan vælges om der ønskes animation, biography, drama eller en af de 17 andre genre, for at få et specifikt udvalg frem.

Visningen af genre var oprindeligt tiltænkt en hel scene med 20 genre-bokse der hver repræsenterer en genre, som en genvej til at få en specifik liste af enten film eller serie frem.



Figur 1 - Illustration af valg mellem genre i første mockup

Gruppen fandt hurtigt ud af denne scene var forholdsvis overflødig, og nemt kunne erstattes med en simpel dropdown menu, i kombination med visning af alle medier inden for den valgte kategori.

3.4 Udvidelser & forbedringer

Mulighed for udvidelser af programmet kan nemt hentes bl.a. fra tabellen udviklet med MoSCoW metoden (Bilag 1). Must-have kategorien redegør for de helt basale brugermæssige krav, angivet i projektbeskrivelsen. Under *should-have* og *could-have* kategorien er der gode muligheder for at finde forslag til forbedring. Det første forslag til forbedring under *should-have* er at brugeren har mulighed for at fortsætte på den film eller serie de sidst lukkede ned. Dette ville under optimale omstændigheder inkludere hvilken episode og sæson, hvis der f.eks. er tale om en serie. Derudover også gerne inkludere antallet af minutter og sekunder der er set af det valgte medie, så brugeren har mulighed for at hoppe direkte tilbage.

Et andet godt forslag til udvidelse ville være etablering af børneprofiler. Under oprettelsen af brugerprofiler på Max Stream bedes brugeren indtaste en alder. Det kunne derfor nemt udvides med en registrering af denne alder. Et eksempel kunne være at hvis en alder på 12 år eller derunder bliver indtastet, vil tjeneste kun tilbyde film og serier i kategorien ‘Family’.

Der er også mange gode forslag til forbedringer i form af specifikke søgninger, eller mulighed for at sortere i rækkefølgen medierne bliver vist. Dette kunne bl.a. være efter IMDB-rating, eller efter udgivelsesår (I serie-tilfælde ville dette være første år i kørselstid). Det kunne også være muligt at søge efter medier med en rating på 7-9 eller udgivelsesår fra 1980-1989. En sorteringsmetode som dette ville kunne implementeres i klassen VideoDB.

En hel anden form for udvidelse af programmet kunne være at tilføje en log-in, og dermed også log-out funktion. Dette ville give en sikkerhed til siden, og bl.a. gøre at der kun ville være adgang til programmet med det korrekte login.

Endnu en form for udvidelse af programmet kunne være en søgefunktion til at søge på en specifik film eller serie. Grundet tidspres er denne funktionalitet desværre ikke en del af projektet. Måden en søgefunktion kunne integreres ind i projektet var eksempelvis at bruge en EventListener, som ville opdatere hver gang der forekommer en ændring i hvad der står i søgerfeltet. Denne funktion ville derfor tage udgangspunkt i hvad brugeren har skrevet i søgerfunktionen, og derefter filtrere efter hvad der er blevet skrevet. Nemlig lignende filtrering med genrer, dog mere dynamisk frem for statisk, hvor man kunne basere inklusionen og eksklusionen af visse film og serier. Dette ville gøres vha. 2 separate HashMaps med film/serier, som stemte overens med genren og en anden, som ikke befandt sig i denne genre. Der blev eksperimenteret med denne løsning i det sidste øjeblik, dog var løsningen ganske suboptimal, og det blev derfor valgt at efterlade denne funktion i et ufuldendt stadie.

Profilsiden kunne godt have flere sikkerhedsforanstaltninger, og mere nøje tjek for hvilke kombinationer af navne og tal var tilladt. Brugeren bliver advaret ved oprettelse af konto, hvis noget ikke stemmer overens med de forudsatte regler, eksempelvis ingen tal i navnefeltet, eller omvendt for den sags skyld. Dog kan der ikke skrives noget med store bogstaver uden det anses som fejl, hvilket skyldes vores unøjagtige regex-udtryk, som skulle verificere inputtet.

Hvad angår profil, anvendes brugernes profiloplysninger minimalt i programmet. Det vigtigste er profilfarven og den brugerspecifikke liste. Som programmet er nu, er navn og alder overflødig, og der burde ikke indsamles mere data end nødvendigt. Generelt er der mange valgmuligheder for indhentning af data i vores program, som sjældent anvendes.

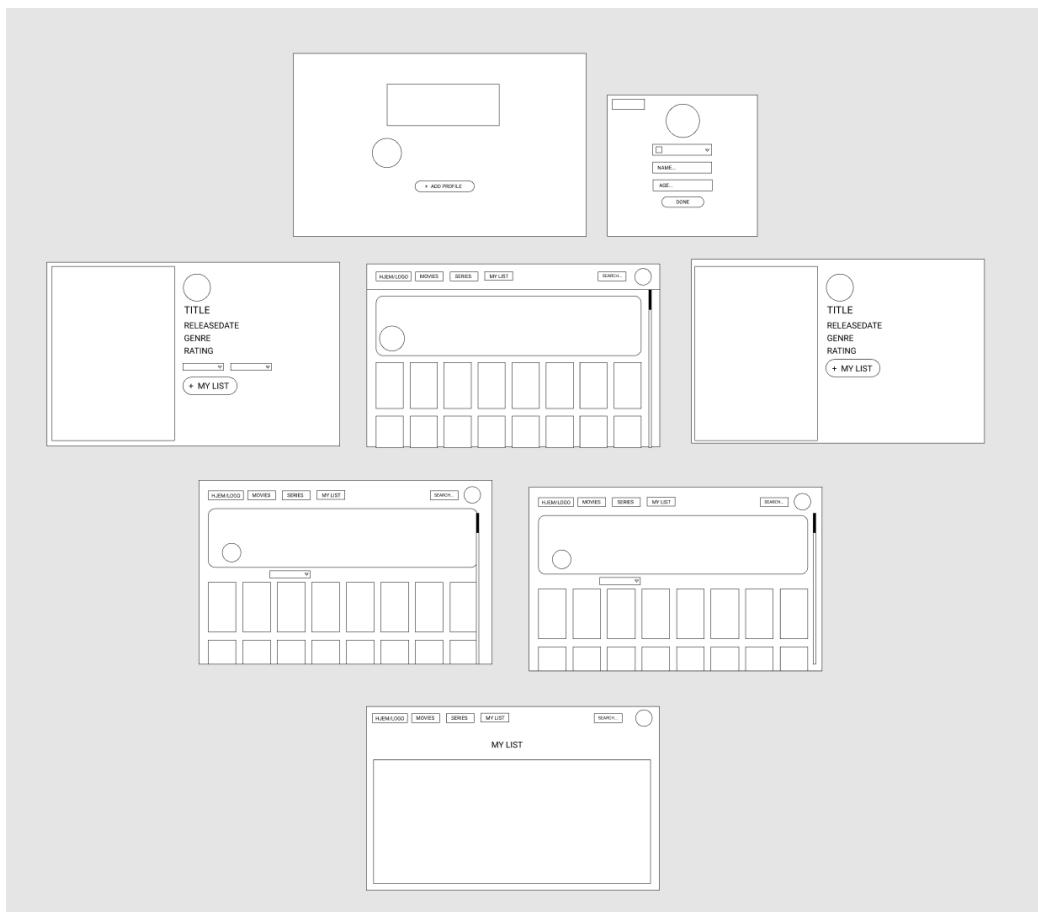
Organisering og oprydning af kode ville have været fordelagtigt under udvikling, men også til fremtidig vedligeholdelse af systemet, hvori der ønskes at opdatere systemet med ny funktionalitet og minimal forvirring bag kodens indforstået natur blandt de/den oprindelige udvikler/udviklere. Under udvikling oplevede gruppen ligeledes at sidde fast et par gange grundet de pludselige løsninger uden hensyn til kvaliteten af denne.

Under projektet havde gruppen minimal forståelse for bygningsautomatiseringværktøjet *Maven*. Selvom dette blev opsat, var gruppen meget usikre på hvordan dette blev konfigureret ordentligt, hvilket højest sandsynligt kunne have gavnnet projektet. Gruppens manglende selvsikkerhed tilbageholdte os i visse tilfælde fra at opsætte bedre struktur i andre packages.

For versionskontrol blev der anvendt værktøjet GitHub. Værktøjet blev dog ikke til alle tider anvendt optimalt, f.eks. blev der sjældent delt filer, og oftest blev det glemt at pushe eller committe i lange intervaller. Opgaverne blev delt op i branches, dog endte det typisk med at koden befandt sig et enkelt sted og på den måde udgjorde en risiko for fejl, eller noget vigtigt blev slettet, hvor det ikke kunne hentes tilbage ved roll back.

4. Brugervejledning

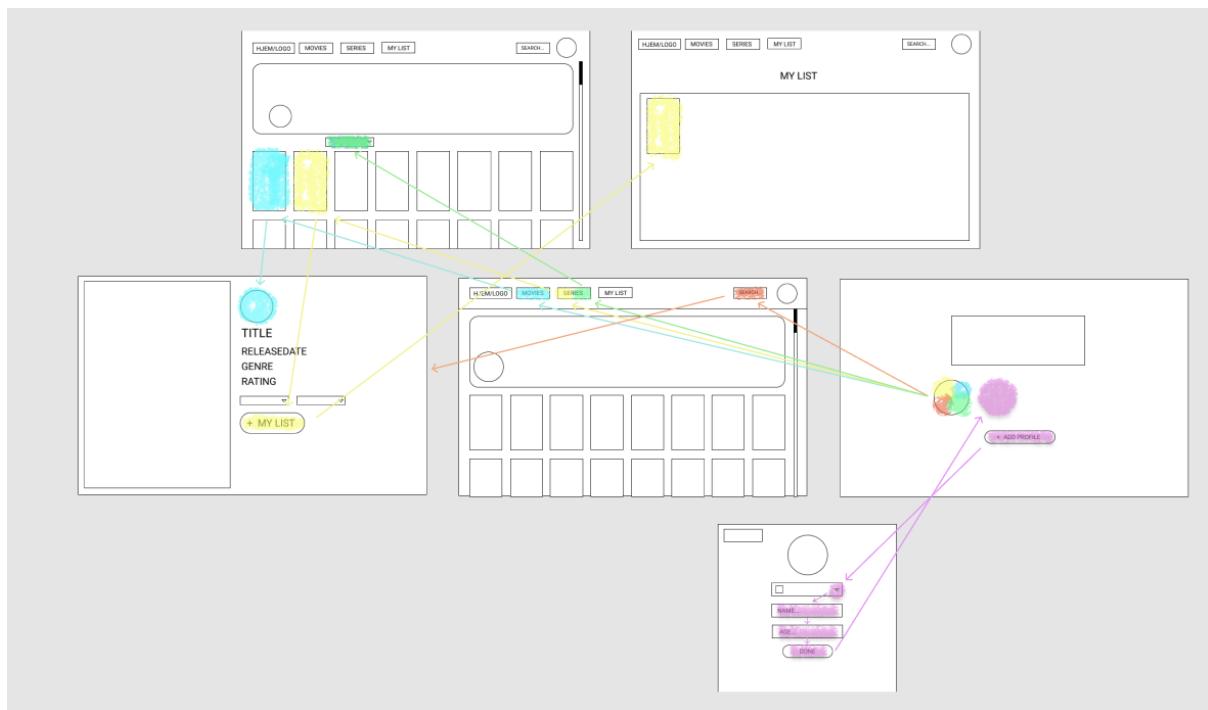
Når Max Stream åbnes, er det første brugeren møder en top profilsiden. Her er mulighed for at klikke ‘Add Profile’, og oprette sidens første profil. Her kan der vælges en ønsket farve, så brugeren har mulighed for at adskille sin profil, fra andre. Herunder skal der også tastes et navn (bogstaver), og en alder (tal). Derefter kan brugeren nu vælge sin profil ud fra den valgte farve, og ville blive sendt til start-siden. Øverst på startsiden findes menulinjen, der kan benyttes til forskellige brugerflows.



Model 1 - Komplet model over brugergrænsefladen

Nedenfor vil der blive gennemgået en række brugerscenarier, og mulige brugerflows.

1. Brugervejledning til at tilføje brugerprofil
2. Brugervejledning at trykke afspil på film/serie
3. Brugervejledning til at tilføje element til ‘My List’
4. Brugervejledning til at få bestemt genre frem af enten film eller serie
5. Brugervejledning til søgning efter film eller serie

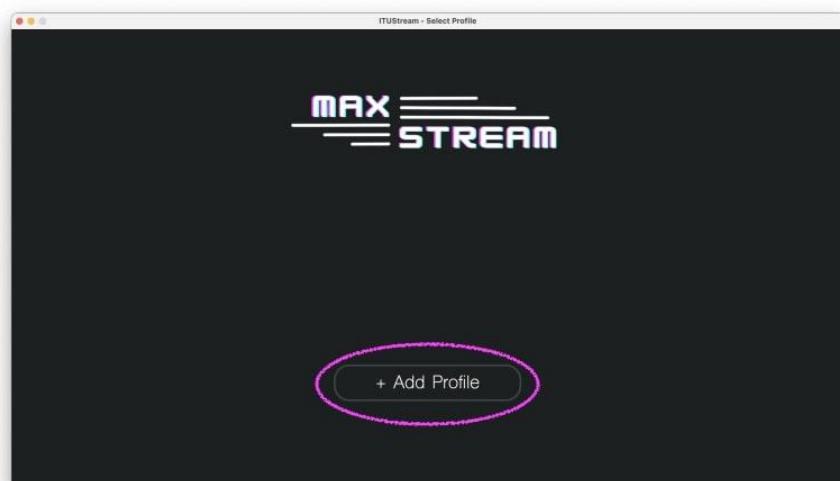


Figur 2 - Komplet model over brugergrænsefladen inklusiv 5 cases i brugerflow med farvekoordinering

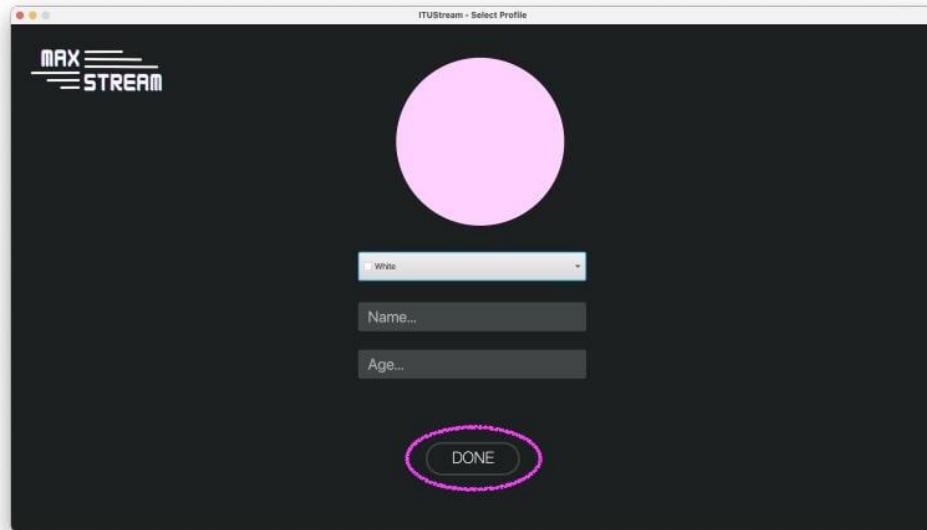
De to mest brugte brugerflow vil formentlig være brugerflow 1 og 2. Da det er nødvendigt for de resterende brugerflows at have en profil på Max Stream, vil brugerflow 1 være et uundgåeligt step. Når dette er udført, vil den mest benyttes funktion, højst sandsynligt være at trykke på afspil af en film eller serie. I figur 2 er det illustreret hvordan brugergrænsefladen, og brugerflows hænger sammen.

1. Brugervejledning til at tilføje en brugerprofil

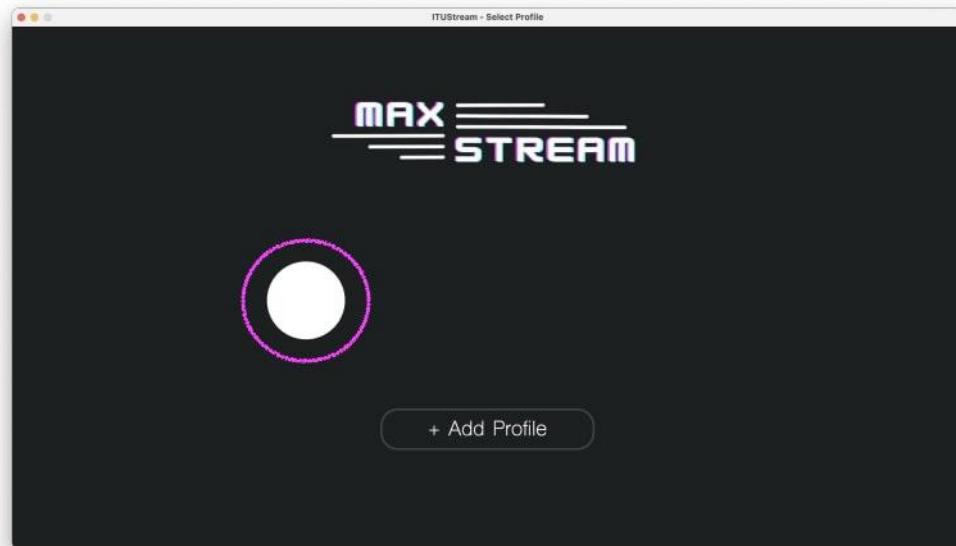
- Ved programmets opstart vises Max Streams logo, efterfulgt af en knap med teksten '+ Add Profile'. Start ved at klikke på denne.



2. Der ses nu en ny side med mulighed for at indtaste brugeroplysninger, på den profil der ønskes at oprettet. Her vælges en farve, så brugerprofilen er genkendelig, og herefter indtastes et navn og alder. Når dette er gjort trykkes der på 'Done'.
Når dette er gjort er brugeren oprettet, og vil nu fremtræde på programmets startside, som følgende:

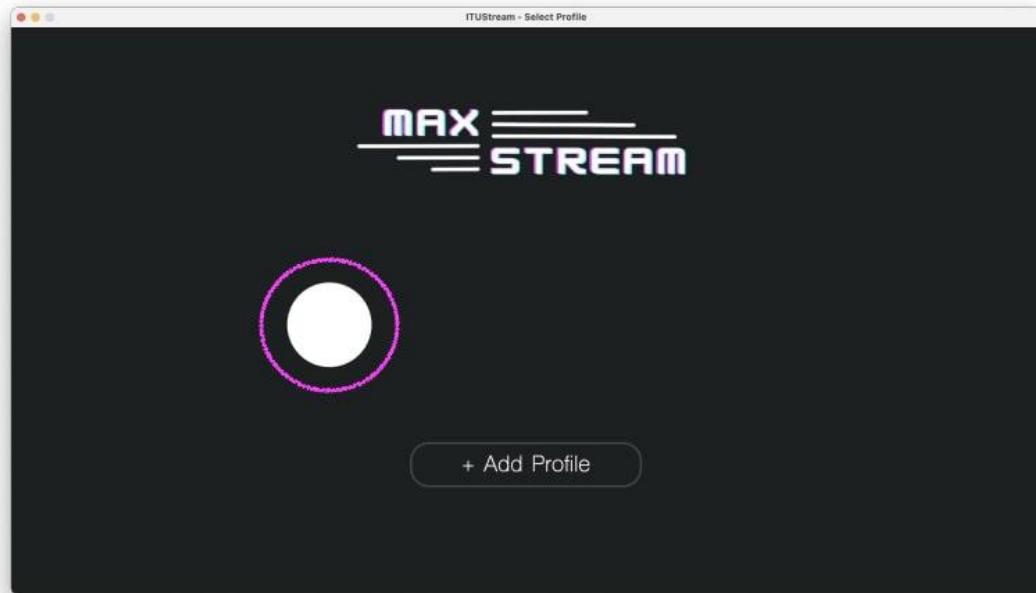


3. Brugeren er nu oprettet. Klik på cirklen med din valgte bruger og kom i gang med at stremme!

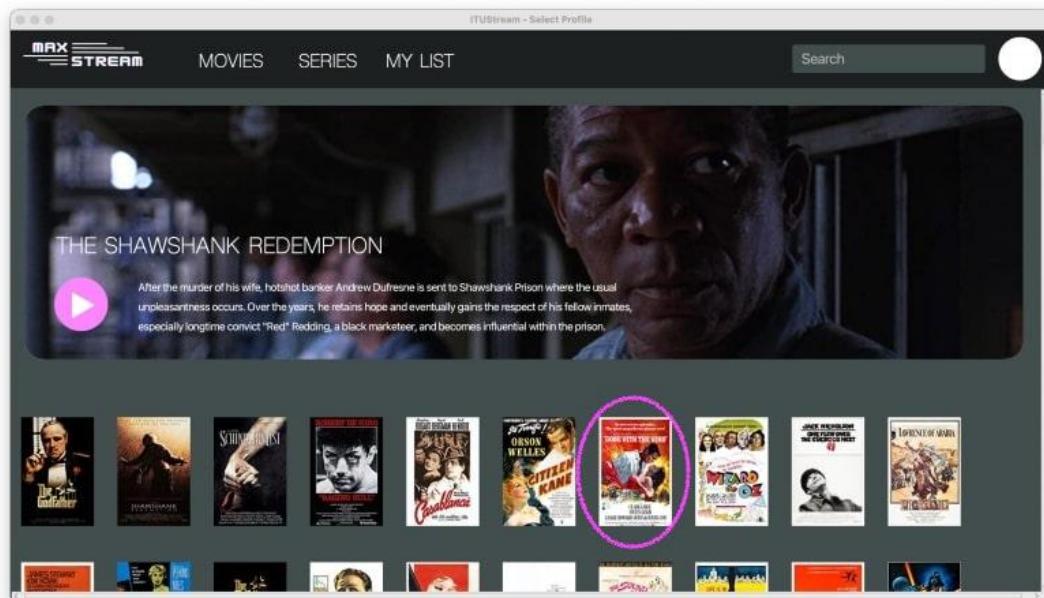


2. Brugervejledning til at afspille en film eller serie

1. Først vælges den brugerprofil, der ønskes at stremme fra.



2. Hjem-siden vises nu. På denne er alle medier vist, og der kan vælges en ønsket film eller serie.

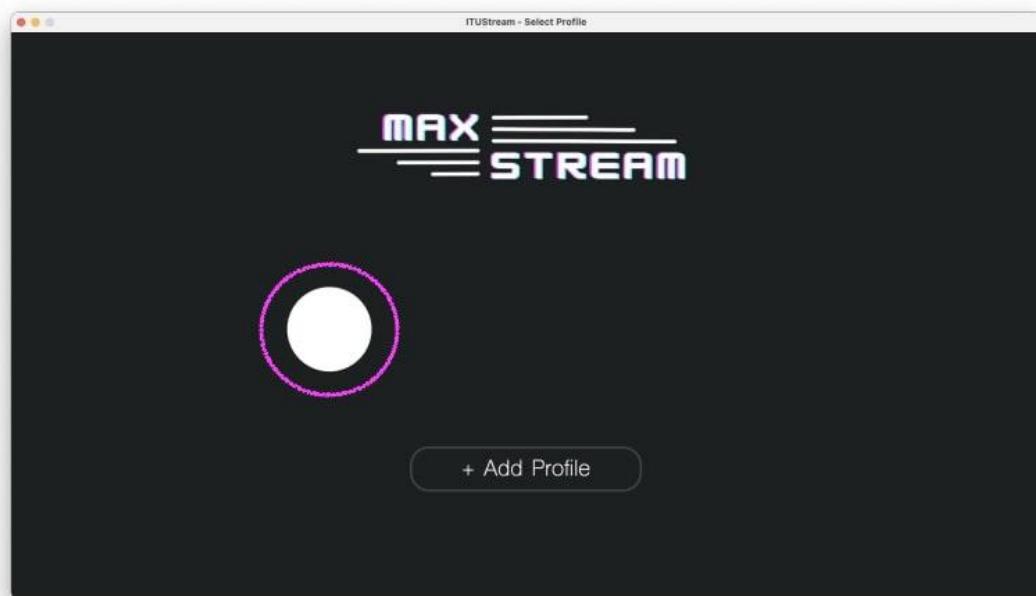


3. Når den ønskede film eller serie er valgt, klikkes der på play-knappen og afspil!

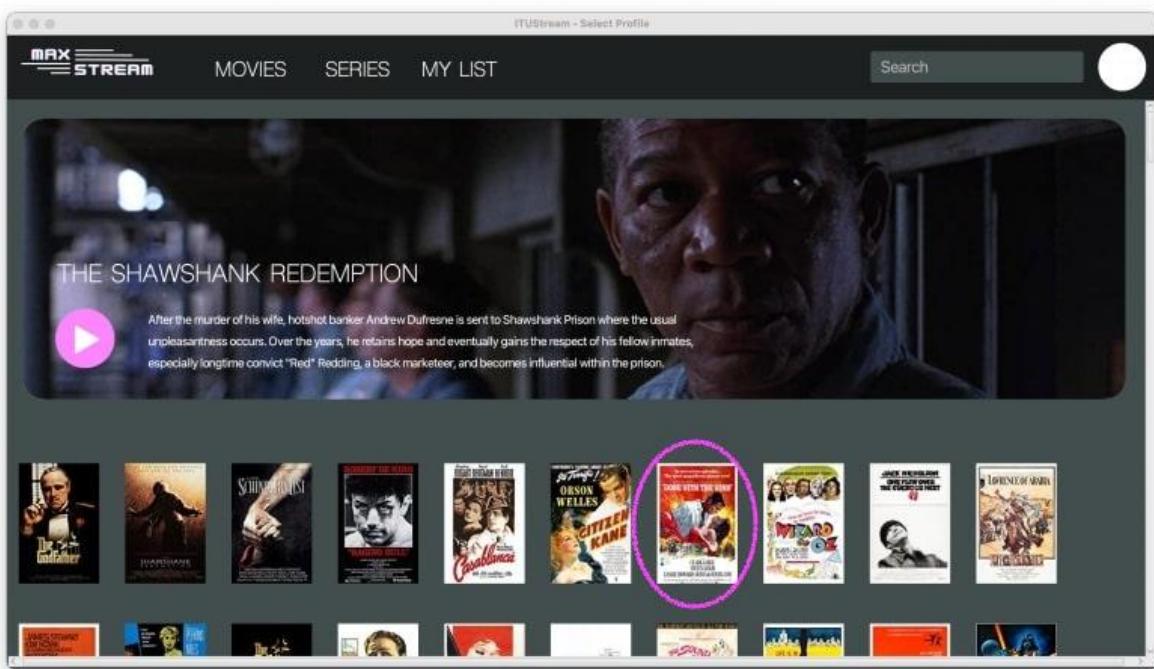


3. Brugervejledning til at tilføje et element til 'My List'

1. Først vælges den brugerprofil, hvis liste der skal tilføjes til.



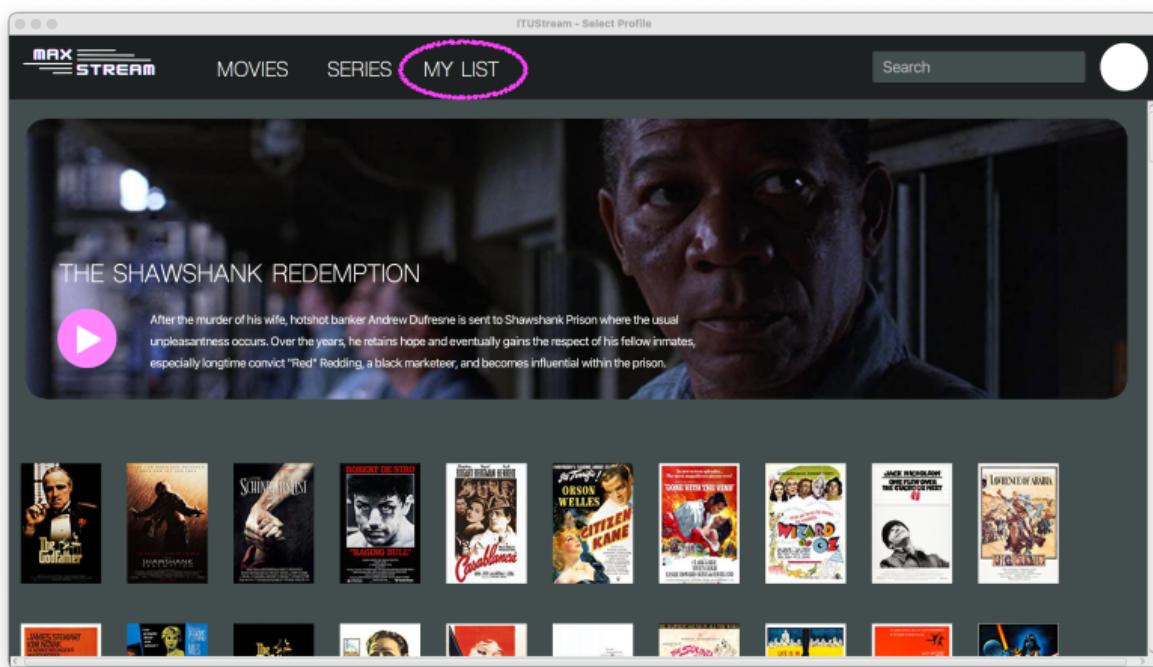
2. Herefter kan der vælges den film eller serie der ønskes tilføjet til listen. Klik på denne.



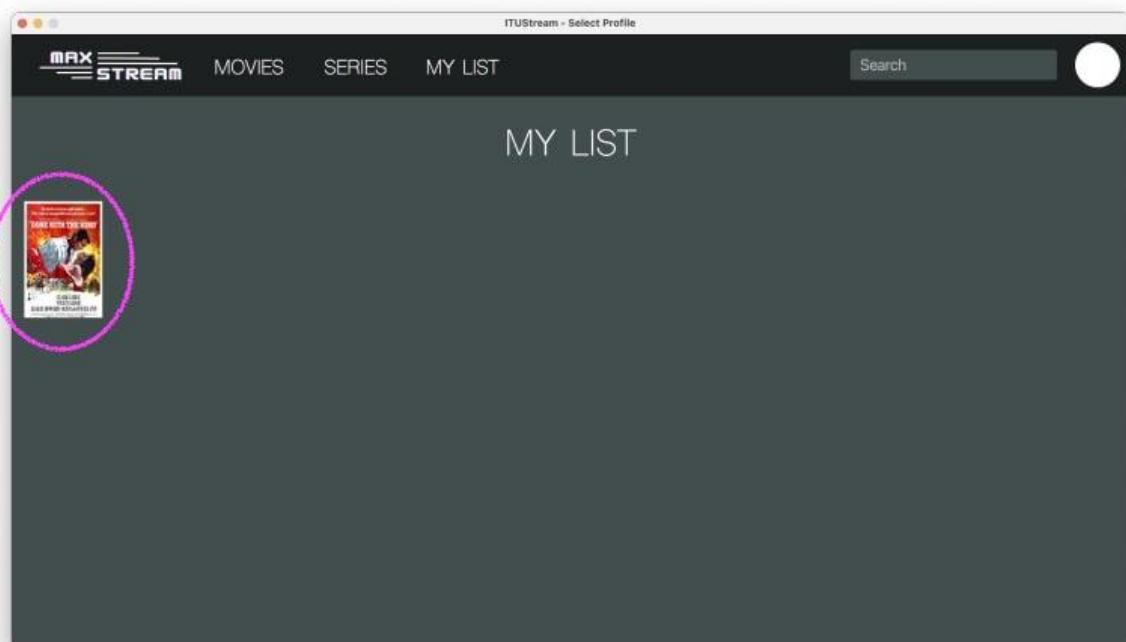
3. Når filmen er valgt kan der klikkes på '+ My List'- knappen. Herefter lukkes vinduet ved at klikke på det lille kryds i øverste højre hjørne.



4. Når denne er lukket, ses nu hjem-siden. Klik nu på 'My List' øverst i midten af menulinjen.

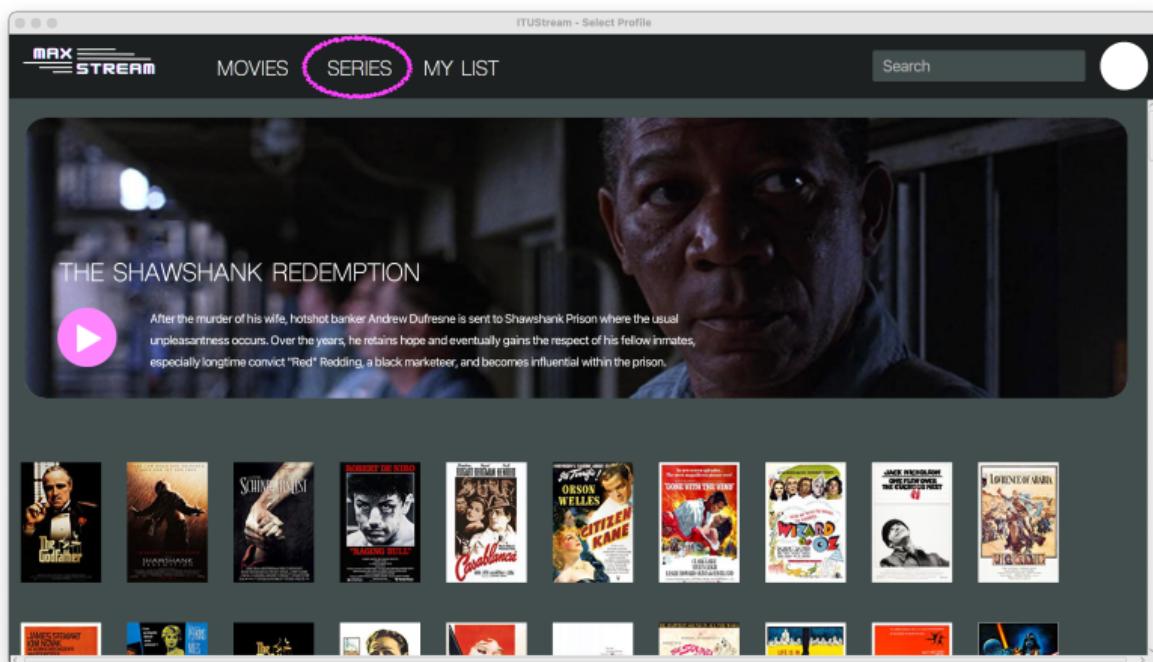


5. Den valgte film eller serie kan nu ses under 'My List'

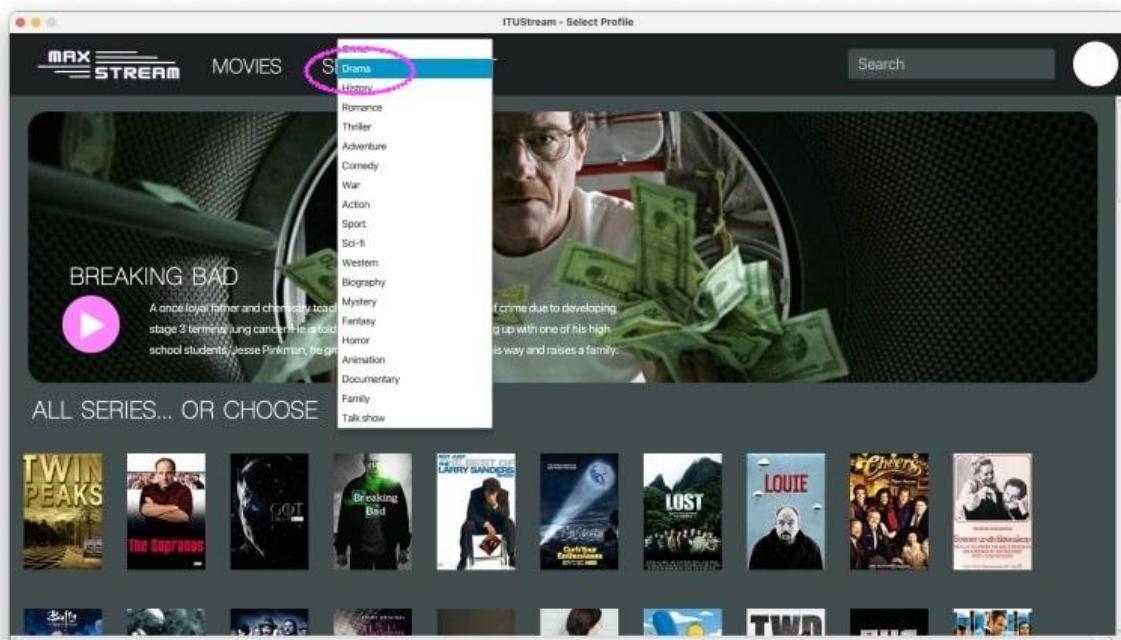


4. Brugervejledning til at finde en genre af enten film eller serier

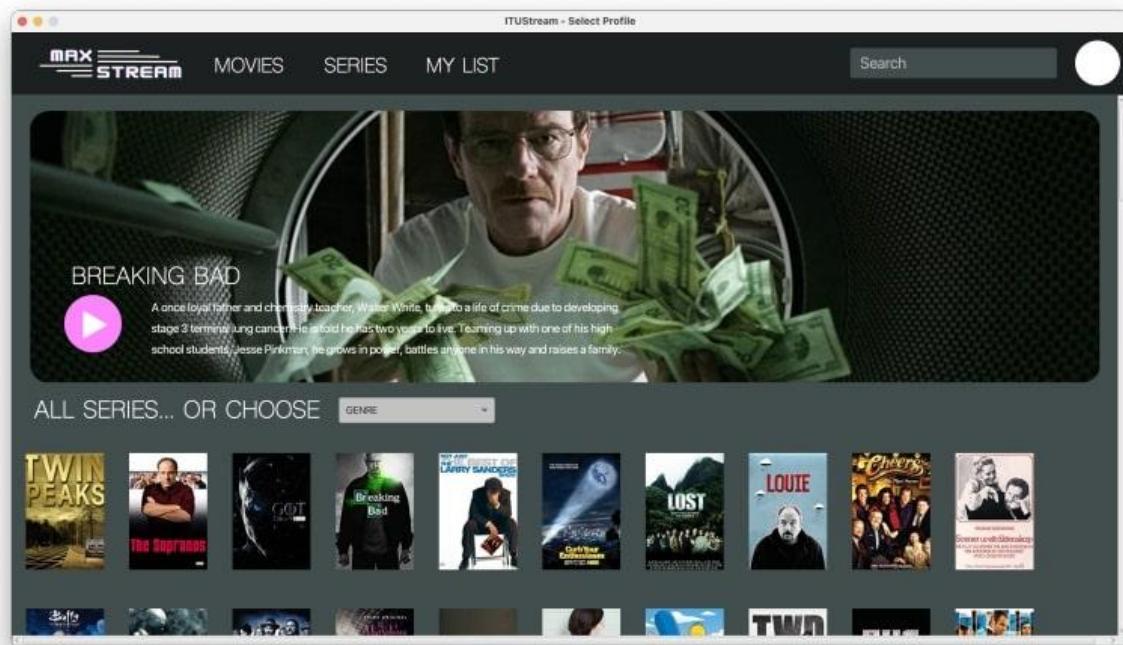
1. Først vælges der om der ønskes en genre af enten film eller serier. I følgende eksempel findes serier i genren, drama.



2. Herefter klikkes der på drop-down-menuen, under serier og den ønskede genre vælges

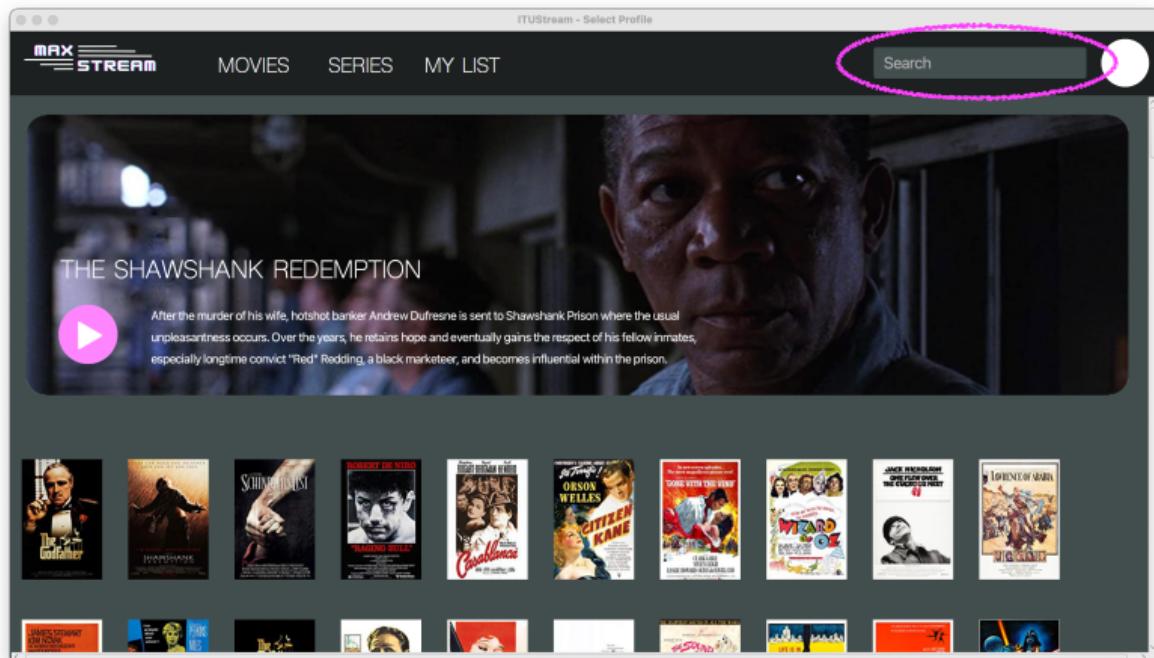


3. Herefter vises kun den valgte medieform i den ønskede genre



5. Brugervejledning til søgning efter film eller serie

Grundet tidspress har det desværre ikke været muligt for gruppen at få søgefunktionen til at fungere, og det 5. step i brugervejledningen er derfor ikke relevant i samme omfang, man ikke kan teste funktionerne i programmet selv. Dog er der et hurtigt redegørende billede af søgefunktionens placering på hjem-siden.



4.1 Fejlmeddelelser og exceptions

I programmet er der ikke direkte opbygget egne exceptions, selvom det i visse tilfælde - bl.a. under oprettelse af profil - ville være bekvemt til at undgå unødvendige problematikker. På trods af dette, anvendes der i stort set alle tilfælde, Exceptions, hvorend det er muligt, eksempelvis i model-klassen *VideoDBs* metode *buildSeriesList*. Hvor der modtages en sti og output forekommer. Her kunne en forkert sti resultere i intet output, således har man en exception til I/O.

5. Teknisk beskrivelse af programmet

5.1 Interne datastrukturer

5.1.1 Objekt-klasser

Der er sat fokus på at have få, men essentielle klasser egnet til objekter som *Video*, *Series* og *Profile*. Disse egnes til at indeholde vigtige oplysninger til f.eks. hver brugers liste og en udvalgt farve til deres brugerprofil.

Når et objekt instantieres for klassen tages der hensyn til de typiske kendeteogn, der bliver benyttet til at beskrive film, f.eks. *title*, *release*, *genres* og *rating*. Eftersom både *Series* og *Movie* nedarver fra abstract-klassen *Series* har de fælles egenskaber: *Series* har alle de samme egenskaber som film, bortset fra den også består af *yearStart* (startåret af sæson), *yearEnd* (slutåret af sæsonen) og *seasons*.

For at få fat i disse oplysninger er der getter-metoder, eksempelvis *getTitle*, *getRating* osv til at bl.a. finde filstien for en bestemt fil, eller udfylde siderne tilhørende filmene eller serierne med oplysninger:

```
@FXML
public void initialize() throws IOException {
    Image image = ImageLoader.imageFinder(VideoDB.getMovieList().get(VideoDB.currentlyShownVideo));
    coverImageView.setImage(image);

    movieTitle.setText(VideoDB.getMovieList().get(VideoDB.currentlyShownVideo).getTitle());
    inputDate.setText(VideoDB.getMovieList().get(VideoDB.currentlyShownVideo).getReleaseYear().toString());

    inputGenre.setText(Arrays.toString(VideoDB.getMovieList().get(VideoDB.currentlyShownVideo).getGenres())
        .replace("[", "")
        .replace("]", ""));
    inputRating.setText(VideoDB.getMovieList().get(VideoDB.currentlyShownVideo).getRating().toString());
```

showMovieController.java - Her benyttes alle oplysninger tilgængelige fra et *Movie*-objekt til at sætte teksten og billedet for detaljesiden af den bestemte film.

Objekt-klasserne er et centralt element til denne streamingtjeneste og bruges i alverdens tilfælde.

5.2 Interne algoritmer

Vores program opstartes i *ITUStreamApplication*, hvori ens *Stage* og første Scene, altså Profile-scenen, køres. I øvrigt kører vi vores 4 interne algoritmer fra *VideoDB*: *buildMovieList()*, *buildSeriesList()* og *buildVideoList()* og *buildGenreList*. Efter dette er udført kan programmet køres, så længe ingen uventede fejl opstår.

Brugereren skal derefter oprette en profil med et navn (String), alder (Integer) og en farve til profilen. For at oprette en profil, bliver brugerden videresendt til en ny scene, hvor profilen oprettes. Når profilen er blevet oprettet og brugerden klikker på den, bliver brugerden sendt videre til *main view*. I *main view* bliver brugerden vist alle film og serier, og bliver givet mulighed for at bevæge sig rundt i programmet.

5.2.1 VideoDB

Klassen *videoDB* er den klasse der står for at indlæse dataen i de to .txt-filer der som tidligere nævnt er blevet tildelt projektet. *videoDB* starter med at oprette 3 *ArrayList*, en liste der består af film, en liste der består af serier og til sidst en liste der består af både film og serier. Til at starte med initialisere klassen listen med film og listen med serier. Dette foregår ved at .txt-filen bliver læst ved brug af en *scanner*, som bliver ved med at køre indtil scanneren har indlæst hele filen. For hver linje som scanneren læser, bliver linjen delt op i mindre linjer som er separeret med semikolon. Disse linjer bliver initialiseret som den rette variabeltype som til sidst bliver brugt til at instantiere et nyt objekt af enten film eller serie. Disse objekter bliver tilføjet til listerne med film og serier, afhængig af om det er en film eller serie. Til slut bliver den sidste *ArrayList* initialiseret som tilføjer alle instanser film og serier fra de to tidligere *ArrayList*.

5.2.2 ImageLoader

I programmet, findes der en klasse, *ImageLoader*, (Bilag 10.4) som har til opgave at tjekke om et givent String-input har en gyldig sti der leder til en jpg-fil med samme navn som String-inputtet. Dette gør den først ved at lave en liste med filer der er i den specificerede mappe, eksempelvis *movie*-mappen. Der oprettes derefter en liste af alle filer i den specificerede mappe og kører derefter igennem alle filerne ved brug af et *foreach-loop*. *foreach-loopet* tjekker for hver fil i listen med filer, om der en fil, hvis navn svarer til det String-input metoden er blevet givet. Hvis dette er tilfældet returnerer metoden denne fils gyldige sti. Hvis der ikke kunne findes nogen gyldig sti vil metoden dog returnere *No filename found*.

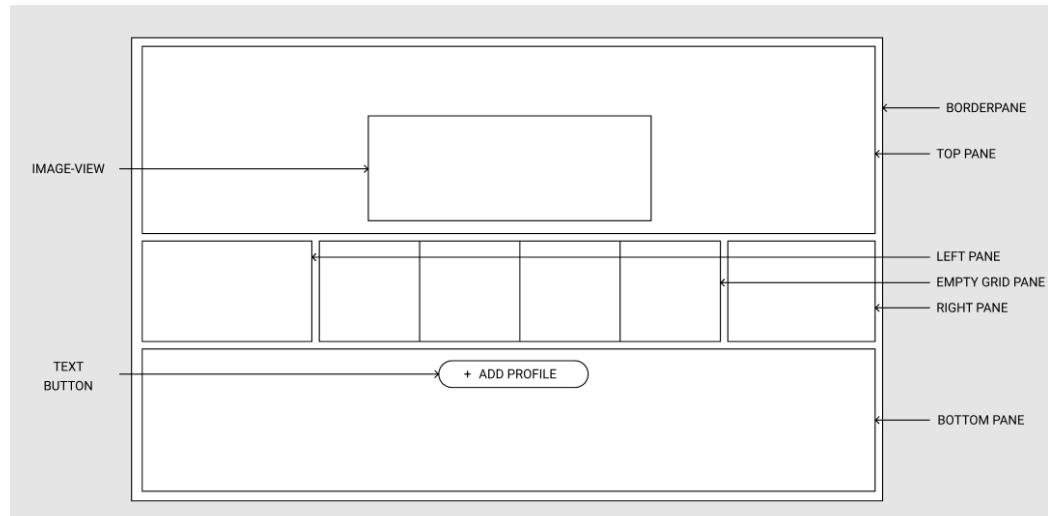
5.3 Brugergrænsefladens struktur

Opbygningen af den grafiske brugergrænseflade er lavet i *SceneBuilder* i frameworkt JavaFX. Hver scene der benyttes i programmet er herigennem oprettet som en FXML-fil. Hver af disse fxml-filer er beskrevet i dybden nedenfor.

5.3.1 Profile

Profile-view bygger på en borderpane, med simple panes som både top, bottom, left og right. Dette er hovedsagligt for at danne ramme for den tomme gridpane der er placeret i center pane. Denne

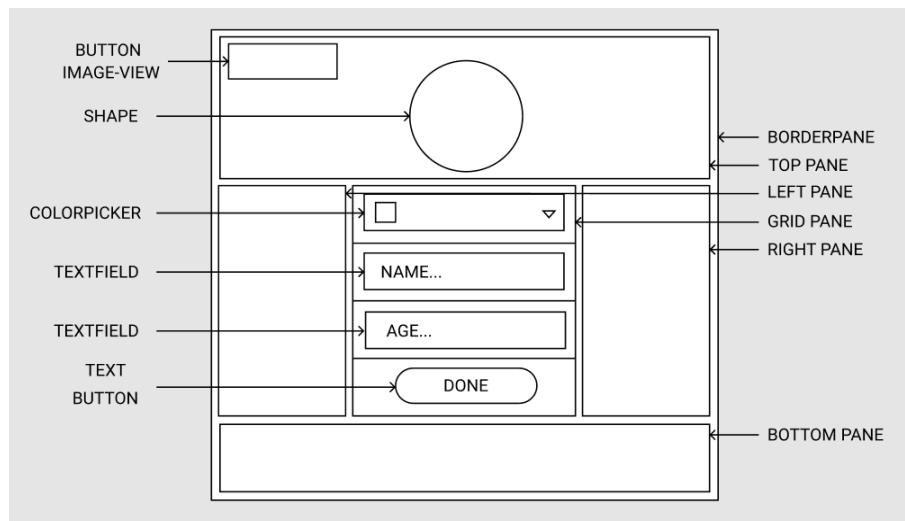
gridpane danner ramme for en kommende user-button. Derudover er der placeret et image-view i top pane, der viser et Max Stream logo. Dette billede har ingen yderligere funktion. Til slut er der en knap i bottom pane med tekst. Denne knap er den eneste mulige interaktion på denne side, før brugeren er oprettet..



Figur 1 - Illustration af hierarkiet i profile-view med tom grid pane

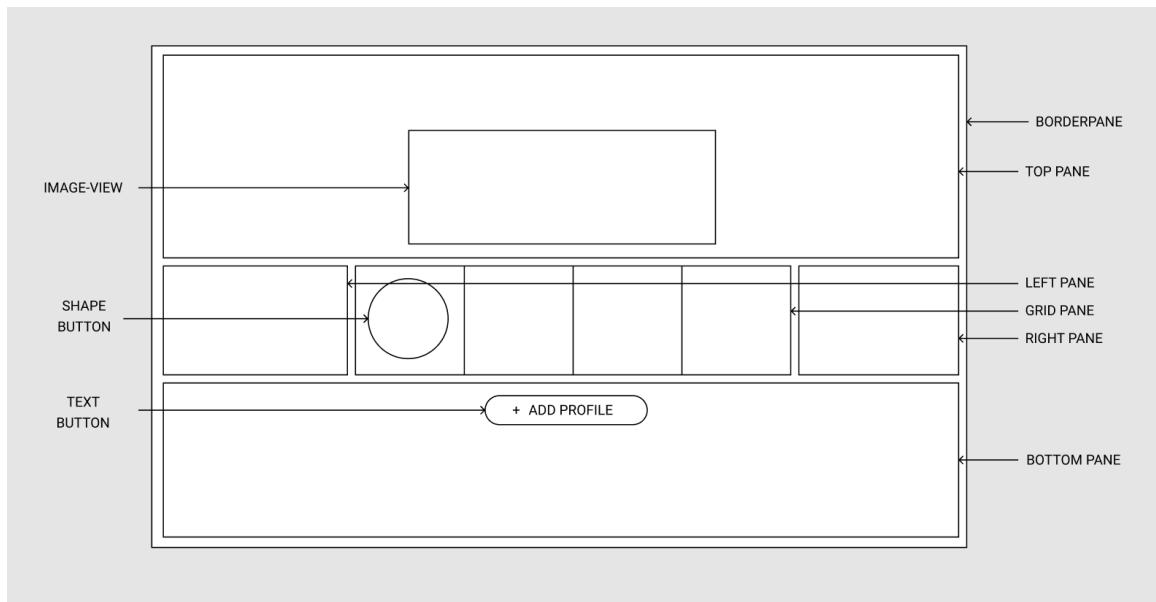
5.3.2 Add Profile

addProfile-view bygger igen på en borderpane, med panes der danner en ramme om et grid pane. I top pane findes en shape som illustrerer den brugerprofil der er i gang med at blive oprettet, og en button med image-view der illustrerer et Max Stream-logo, og samtidig fungere som en genvej tilbage til profileView. I center pane er der, ligesom ovenfor, placeret et grid pane. I den øverste grid er en colorpicker, der skifter farve på den illustrerede brugerprofil. I de to følgende grids er placeret et textfield i hver med en prompt text der henviser til navn og alder. Til slut er det sidste grid en button med text, 'Done', for at indikere at brugeren nu er oprettet. Denne sender brugeren tilbage til profile-view. Left-, right, og bottom pane er alle tomme panes.



Figur 2 - Illustration af hierarkiet i addProfile-view

Når en brugerprofil er oprettet vil addProfile lukkes ned, og der vil nu på profile-siden fremtræde en button med samme shape og farve som den illustrerede i addProfile. Det gør det muligt at klikke på den oprettede profil, og vil tage brugeren til main siden. Hierarkiet i denne scene er nu illustreret som følgende.



Figur 3 - Illustration af hierarkiet i profile-view med knap i gridpane.

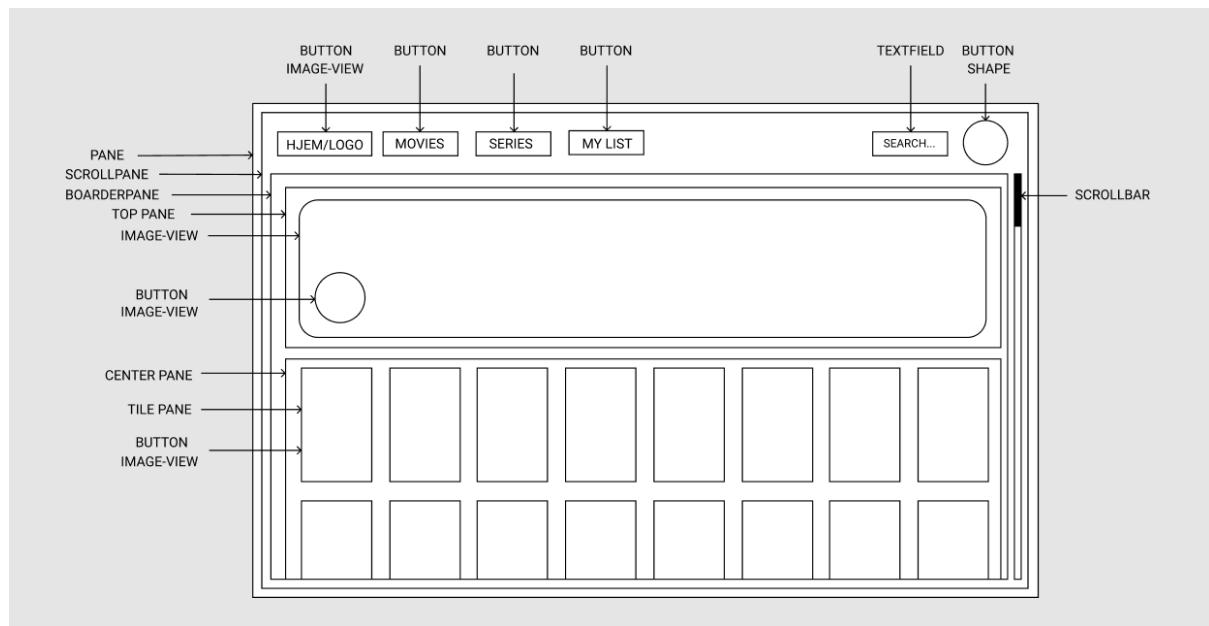
5.3.3 Main

Main view er bygget på en simpel pane. Oven på denne pane er lagt en scroll pane for at opnå funktionen af at kunne scrolle ned gennem alle medier på siden. Oven på denne scroll pane er lagt en borderpane for at have mulighed for at inddæle scenen yderligere, og skabe et brugervenligt design.

Den bagerste pane danner både ramme for resten af elementerne på siden, men det er også i denne menu-linjen befinner sig. Årsagen til dette er at menulinjen på denne måde ligger i fast position, og ikke rykker sig, når brugeren scroller ned igennem siden. I denne menulinje befinner sig først 4 knapper, hvoraf den første indeholder et image-view med et Max Stream Logo. Denne knap har både funktion i at agere som logo, men også at fungere som en 'Start'-knap, til at sende brugeren tilbage til main-scenen, skulle de befinde sig i en anden scene. De følgende tre knapper tager brugeren til den beskrevne side. I højre side af menulinjen indsæt et textfield som søgefelt, og en user-button, efter samme illustration som nævnt i de tidligere afsnit.

Under borderpane findes der en top- og center pane. Top pane danner ramme for en anbefaling af enten film eller serie. Herunder en titel på filmen, et kort resume af filmen og en play-button. Center pane er et tile pane, med det formål at hente data, og vise de tilgængelige film og serier. Denne tile pane fyldes ud af en knap med et image-view for hvert medie tilføjet på siden.

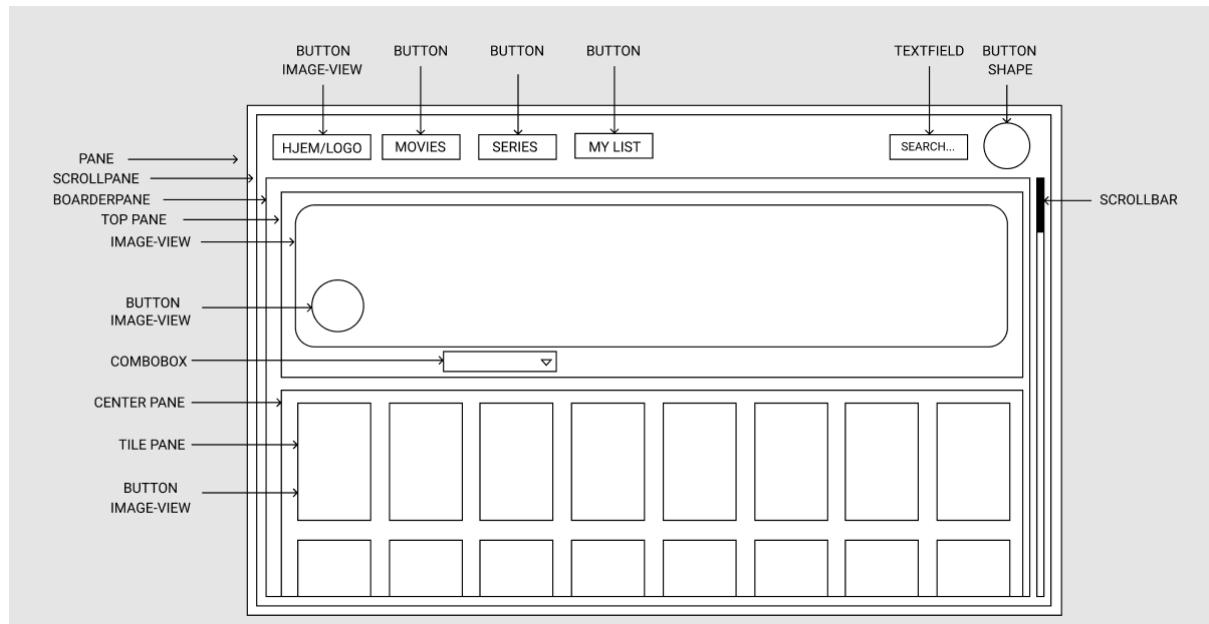
Helt til højre på siden findes en scrollbar for at illustrerer muligheden for rulle ned gennem tjenestens tilgængelige medier.



Figur 4 - Illustration af hierarkiet i main-view.

5.3.4 Main Movies / Series

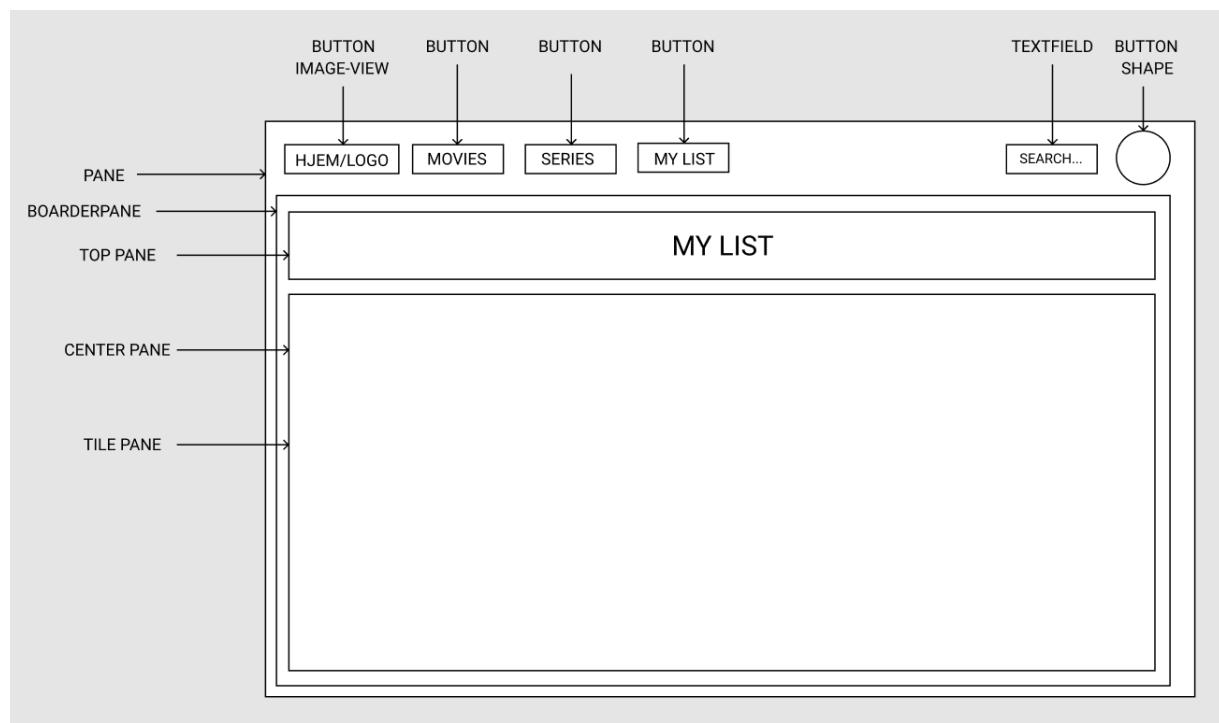
Movies- og series-view er opbygget på samme måde som main-view, med den forskel at top pane nu er udvidet, og at der her er gjort plads til en combobox (drop down menu). Dennes formål er at gøre det muligt for brugeren at vælge genre, uden at forlade scenen.



Figur 5 - Illustration af hierarkiet i mainMovies-view/mainSeries-view.

5.3.5 My List

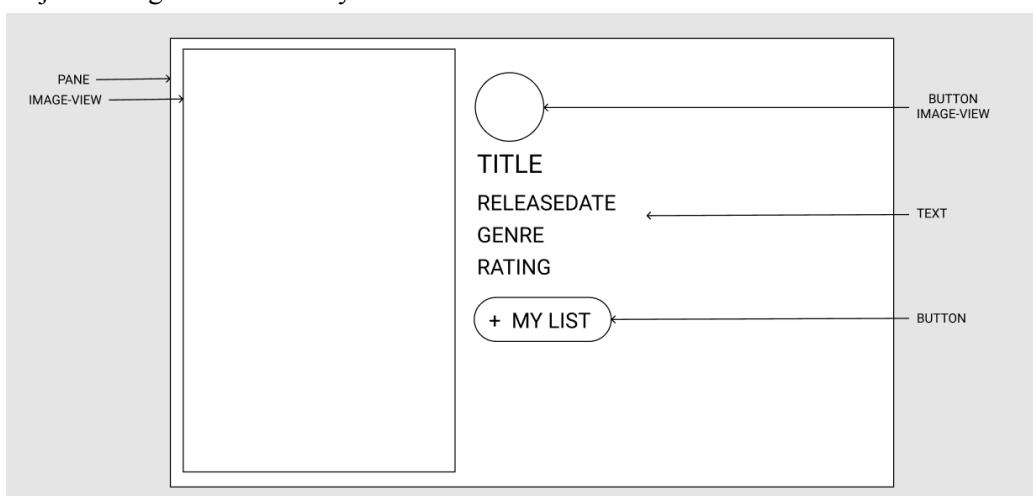
My List er bygget op omkring en simpel pane for at opnå en menulinje på samme vis som i ovenstående eksempler. Her er det dog vurderet at der ikke er behov for en scroll-funktion. Herefter en borderpane for at kunne dele scenen op. I top pane findes 'My List' i text form. Derunder i center pane er der igen benyttet en tile pane for kunne tilføje medier på den mest optimale måde.



Figur 6 - Illustration af hierarkiet i myList-view.

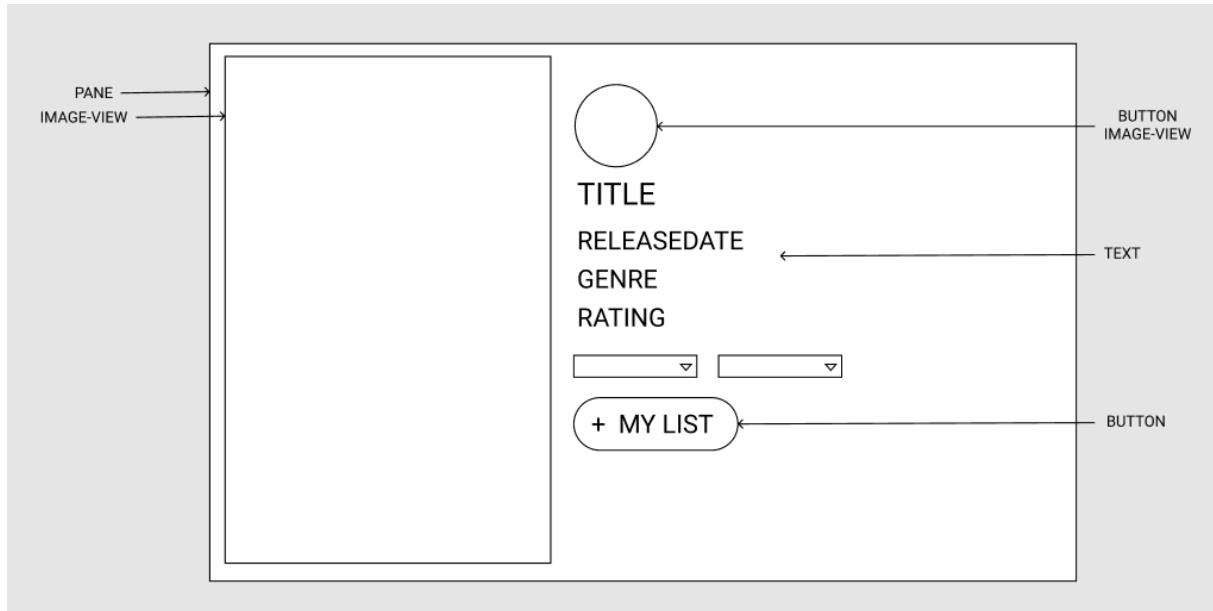
5.3.6 Show Movies / Series

Show Movies eller Series er bestående af endnu en simpel pane. I hele venstre side af denne pane er der placeret et image-view med henblik på film- eller serieforside plakat. I højde side er der øverste placeret en button med image-view der giver en playbutton. Neden for dette, information om udgivelsesdato (kørselstid, i tilfælde af serie), genre og rating, efterfulgt af en button med mulighed for at tilføje det valgte medie til 'My List'



Figur 7 - Illustration af hierarkiet i showMovies

Hvis det valgte medie er i kategorien serie, ser illustrationen en smule anderledes ud, da der her er placeret 2 combo boxes over add-to-myList-knappen. Disse to har til formål at vise seriens sæsoner og episoder, så det er muligt at vælge en specifik episode.



Figur 8 - Illustration af hierarkiet i showSeries.

6. Afprøvning

6.1 Afprøvning af programmet under udvikling

Afprøvningen af programmets kode er essentiel for at sikre sig at programmet udfører de ønskede funktioner på ordentlig vis. Derfor er der løbende blevet testet gennem udviklingsprocessen, ved brug af debugging og System.out.println for at teste om metoderne fungere som de skal. Derudover er der lavet små main-metoder i klasserne der har haft til formål at teste om en specifik metode fungere som forventet. Den største udfordring har været at få indlæst data fra de to givne txt-filer. Der blev først forsøgt med en bufferedReader, men da der opstod mange problemer med denne metode, blev der i stedet brugt en scanner til at indlæse data fra filerne. Derudover opstod der problemer ved at indlæse jpg-filerne af film- og serieforside plakater. Grunden til fejlen fandtes i at stien til filerne ikke var korrekt, og derfor kunne billederne ikke indlæses til programmet. Denne fejl blev løst ved at undersøge hvordan Java håndtere stier, og hvordan de findes. Fejl som denne er blevet løst løbende. Dette har også resulteret i at størstedelen af den afprøvning der er foregået har været uformel. Grundet tidspress er mængden af formel afprøvning begrænset. Dog er der udført en del af afprøvningen ved brug af Unit Testing med JUnit⁵. Her er der udviklet en Unit Test til afprøvning af klassen *ImageLoaders* *findPath*-metode (Se Bilag 10.5). Denne test er blevet udført for at sikre at programmet kan håndtere hvis den ikke kan finde stien for et givet input. For at opsumere var vores overordnede afprøvningsstrategi at teste løbende ved brug af små main-metoder og debugging.

⁵ JUnit: et Library i Java der specifikt er udviklet til test af funktionalitet i et program.

6.2 Tidskrævende fejl opdaget gennem afprøvning

Da .txt- og jpg-filerne skulle indlæses, oplevede gruppen først og fremmest filer i .txt-filen med semi-kolon. Dette var en fejl der tog lang tid at rettet op på, eftersom gruppen havde antaget at gruppens egen kode var forfejlet, således blev der anvendt debugging-funktionen i IntelliJ, med breakpoints til at undersøge problemet nærmere og opdagede et pludseligt stop, hvor gruppen konkluderede, at fejlen befandt sig i .txt-filen for serier. Denne blev rettet op på, hvorefter endnu en fejl opstod. Denne var dog gruppens egen skyld, eftersom filen var konverteret til en anden indkodning, hvori å ikke eksisterede. Dette blev altså skiftet ud med tegnet “, og programmet kunne ikke finde den tilhørende sti til titlen af *Series*-objektet.

| | | | |
|----|--|----|--|
| 21 | House Of Cards; 2013-2018; Drama; 8,9; 1-13, 2-13, 3-13, 4-13, 5-13, 6-8; | 21 | House Of Cards; 2013-2018; Drama; 8,9; 1-13, 2-13, 3-13, 4-13, 5-13, 6-8; |
| 22 | - Fargo; 2014- ; Crime, Drama, Thriller; 9,0; 1-10; 2-10; 3-10; | 22 | + Fargo; 2014- ; Crime, Drama, Thriller; 9,0; 1-10; 2-10; 3-10; |
| 23 | Angel; 1999-2004; Action, Drama, Fantasy; 8,0; 1-22, 2-22, 3-22, 4-22, 5-22; | 23 | Angel; 1999-2004; Action, Drama, Fantasy; 8,0; 1-22, 2-22, 3-22, 4-22, 5-22; |
| 24 | V; 1984-1985; Action, Adventure, Sci-fi; 7,3; 1-13; | 24 | V; 1984-1985; Action, Adventure, Sci-fi; 7,3; 1-13; |
| 25 | Jessica Jones; 2015- ; Action, Crime, Drama; 8,1; 1-13, 2-13; | 25 | Jessica Jones; 2015- ; Action, Crime, Drama; 8,1; 1-13, 2-13; |
| 26 | 00 -38,7 +38,7 00 Alfred Hitchcock Presents; 1955-1962; Crime, Drama, Mystery; 8,6; 1-39, 2-39, 3- | 26 | 00 -38,7 +38,7 00 Alfred Hitchcock Presents; 1955-1962; Crime, Drama, Mystery; 8,6; 1-39, 2-39, 3- |
| 38 | I Love Lucy; 1951-1957; Comedy, Family; 8,3; 1-35, 2-31, 3-31, 4-30, 5-26, 6-27; | 38 | I Love Lucy; 1951-1957; Comedy, Family; 8,3; 1-35, 2-31, 3-31, 4-30, 5-26, 6-27; |
| 39 | 24; 2001-2010; Action, Crime, Drama; 8,4; 1-24, 2-24, 3-24, 4-24, 5-24, 6-24,7-24, 8-24; | 39 | 24; 2001-2010; Action, Crime, Drama; 8,4; 1-24, 2-24, 3-24, 4-24, 5-24, 6-24,7-24, 8-24; |
| 40 | The Americans; 2013-2018; Crime, Drama, Mystery; 8,4; 1-13, 2-13, 3-13, 4-13, 5-13, 6-10; | 40 | The Americans; 2013-2018; Crime, Drama, Mystery; 8,4; 1-13, 2-13, 3-13, 4-13, 5-13, 6-10; |
| 41 | - Girls; 2012-2017; Comedy, Drama; 7,3; 1-10, 2-10, 3-12, 4-10, 5-10; 6-10; | 41 | + Girls; 2012-2017; Comedy, Drama; 7,3; 1-10, 2-10, 3-12, 4-10, 5-10; 6-10; |
| 42 | Mad Men; 2007-2015; Drama; 8,6; 1-13, 2-13, 3-13, 4-13, 5-13, 6-13, 7-14; | 42 | Mad Men; 2007-2015; Drama; 8,6; 1-13, 2-13, 3-13, 4-13, 5-13, 6-13, 7-14; |

Venstre side = film.txt før rettelse. Højre side = film.txt efter rettelse

7. Refleksion over arbejdsprocessen

Indledningsvist mødtes gruppen for at diskutere ambitionerne for projektforløbet, og lave en målsætning. Her blev der sat en målsætning i form af ønskede features ved at udvikle en MoSCoW-tabel (Bilag 10.1). Det blev aftalt at mødes fast på ITU, for at kunne tale sammen om udviklingen af produktet undervejs.

Arbejdsprocessen gennem projektet har hovedsageligt fungeret efter uddelte arbejdsopgaver. En erfaring gruppen tager med sig til næste projekt, ville være en mere dybdegående diskussion af funktionen af alle arbejdsopgaver, så alle gruppemedlemmer får lige stor forståelse for alle dele af programmet. Dette vil hjælpe meget f.eks. i situationer under tidspres, og med den generelle udvikling af programmet, da alle kan bidrage ligeligt.

Derudover har gruppen fået forståelse for vigtigheden i at strukturere udviklingen af programmet, og lægge en plan for udviklingen af produktet. Dermed er det muligt at kortlægge de tidskrævende dele, og lægge disse først i projektet, hvis muligt. En mulighed for dette ville bl.a. være at opbygge en tage udgangspunkt i skitse af en klassestruktur, og de formodede forbindelser gennem et UML diagram⁶. Projektforløbet har været både udfordrende og lærerigt. Gruppen har opnået en dybere forståelse for brugen af læringsmaterialet på kurset, Grundlæggende Programmering. Derudover har alle medlemmer i gruppen taget de ovenstående erfaringer med sig, til benyttelse i et senere projekt.

8. Konklusion

Dette projekt udgør en udgave af et softwaresystem til en streamingtjeneste, med brugerflade grænse i fokus. Projektets analysefase skabte en række krav, og dannede på denne måde rammer om det projekt. Det er lykkedes gruppen at skabe en let tilgængelig brugergrænseflade, med fungerende

⁶ <https://www.uml-diagrams.org>

funktioner i de absolut væsentligste roller. Programmet indeholder dog stadig et par mangler, hvorfaf de største er søgefunktion, og muligheden for at slette elementer fra 'My List'. Eftersom dele af programmet er udviklet under tidspres, er der mange muligheder for udvidelse og forbedring, såsom ovennævnte mangler eller implementering af børneprofiler og bedre sorteringsmuligheder.

Det vurderes at Max Stream lever op til en stor del af de bruger- og systemmæssige krav, men indeholder dog stadig væsentlige mangler.

9. Litteratur

Barnes, D. J. & Kölking, M. (2016) Objects First with Java: A Practical Introduction Using BlueJ (Global Edition), 6. udgave, Pearson.

Link til MoSCoW-metode - <https://www.productplan.com/glossary/moscow-prioritization/>

Brug af billeder og kort resume af 3 valgte film/serie til illustration i programmet

The Shawshank Redemption:

Billede: <https://www.imdb.com/title/tt0111161/mediaviewer/rm3195665921/>

Resumé:<https://www.imdb.com/title/tt0111161/plotsummary>

Jurassic Park:

Billede: <https://www.imdb.com/title/tt0107290/mediaviewer/rm2972830720/>

Resumé:<https://www.imdb.com/title/tt0107290/plotsummary>

Breaking Bad

Billede: <https://www.imdb.com/title/tt0903747/mediaviewer/rm2960540672/>

Resumé:<https://www.imdb.com/title/tt0903747/plotsummary>

10. Bilag

10.1 MOSCOW metode

| MoSCoW | Indhold |
|-------------|--|
| Must have | <ul style="list-style-type: none"> - Genrekategorier - Profiler <ul style="list-style-type: none"> - Min liste - Tilføj/slet film/serier til liste - Skifte i mellem profiler - Søgefunktion - Der skal være billeder knyttet til film og serier |
| Should have | <ul style="list-style-type: none"> - Fortsætte hvor brugeren stoppede - Børneprofiler med begrænset adgang |

| | |
|---------------|---|
| | <ul style="list-style-type: none"> - Sæson + episode navne (serier) - En bruger vil gerne søge efter film, der har større rating end 8,5 - Indstillinger-side - En bruger vil gerne søge efter film fra 80'erne |
| Could have | <ul style="list-style-type: none"> - Søg via rating - Login med brugernavn og adgangskode - Randomize-funktion (foreslår en tilfældig film/serie) |
| Will not have | <ul style="list-style-type: none"> - Tilføj andre medier (e-bøger, lydbøger, spil) - Tilføj film eller serier hvis bruger er admin - Tilføj/fjerne kategorier såsom "Christmas", "halloween" |

10.2 Verb/Noun metode

Substantiver & Verber Analyse:

I systemet er der **medier**, der kan **afspilles**. Medier kan være **film** eller **serie-episoder**. Både **film** og **serier** er **inddelt** i **kategorier** som fx crime, war, drama, family, romance og sci-fi. **Serier** består desuden af **sæsoner** og **episoder**.

Løsningen skal **understøtte** **almindelige opgaver**, som I **kender** det fra streamingtjenester, fx

- Brugeren vil gerne **se** en **oversigt** over alle **krigsfilm**
- Brugeren vil gerne **se** en **oversigt** over alle **dramaserier**
- Brugeren vil gerne **gemme** en **film** i "min liste"
- Brugeren vil gerne **se** sin **liste**
- Brugeren vil gerne **slette** en **film** fra sin **liste**
- Brugeren vil gerne **skifte** til en anden **bruger**

Brugeren vil **søge** efter en bestemt **film** eller **serie**

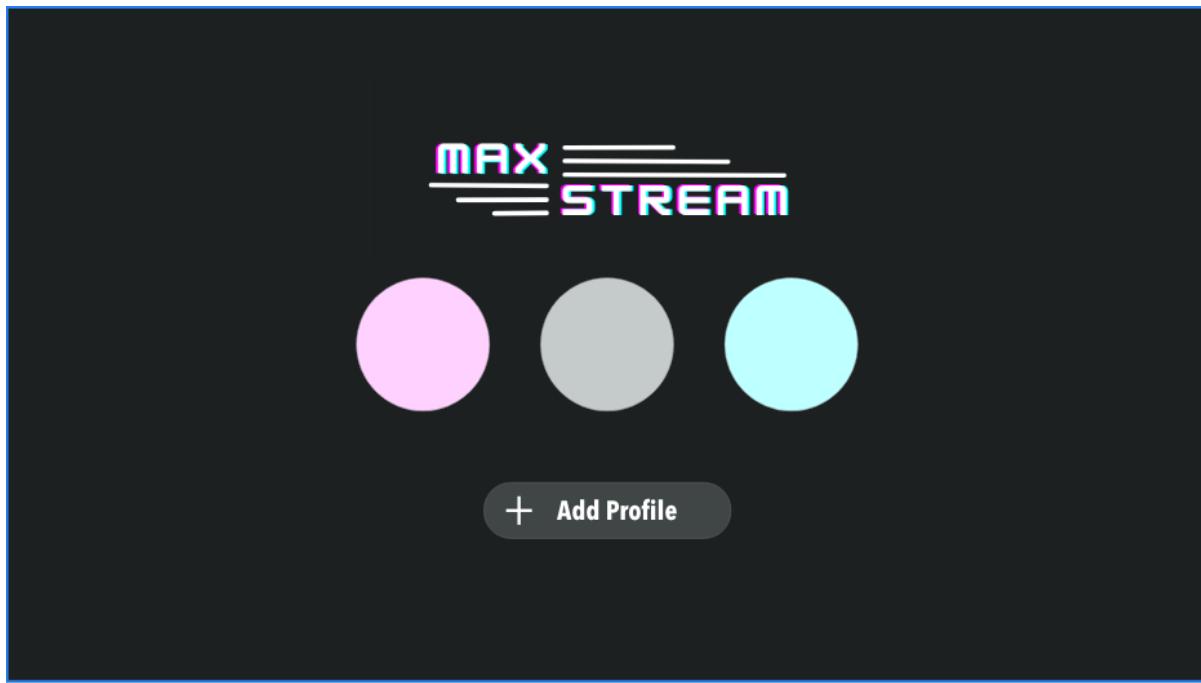
Der skal være **billeder knyttet** til **film** og **serier** – **udleveres** i **.zip-fil**.

Der **kan tænkes** en masse ekstra **features** ind, som fx (ikke prioriteret rækkefølge)

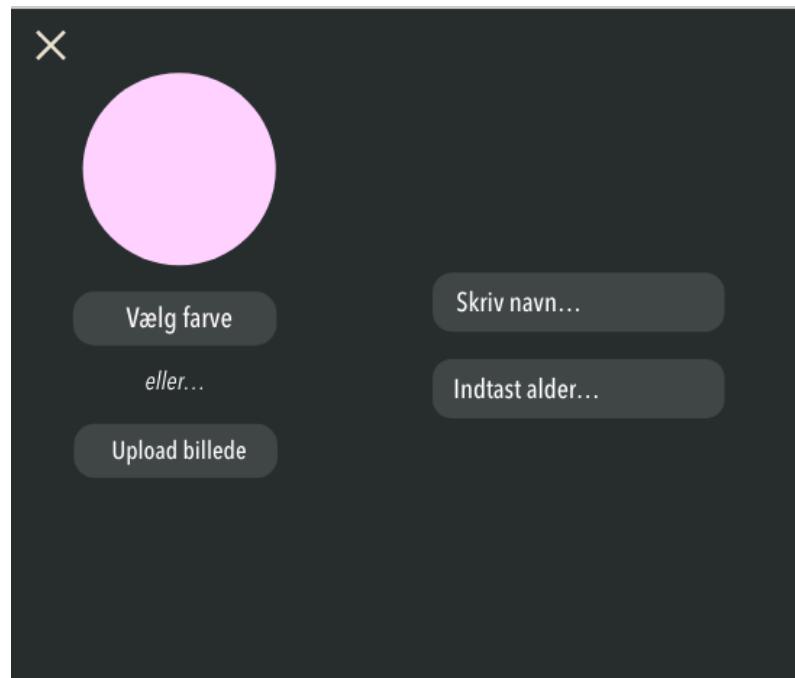
- En **bruger** vil gerne **søge** efter film, der har **rating** større end 8,5
- En **bruger** vil gerne **søge** efter film fra 80'erne
- Der skal kunne **tilføjes** og **fjernes** **kategorier** på en nem måde, fx "Christmas" eller "Halloween"
- Der skal kunne **tilføjes** film, serier, sæsoner og episoder hvis brugeren er administrator
- Der skal kunne **tilføjes** og **fjernes** brugere
- En **bruger** kan være klassificeret som "barn" og må således kun **se** **film** i **kategorien** "Family"

| NOUN - Klasser | VERB - Metoder |
|---------------------|----------------|
| Systemet | Afspilles |
| Medier | Inddelt |
| Film | Består |
| Serie-episode | Understøtte |
| Serie | Se |
| Kategori | Gemme |
| Sæsoner | Slette |
| Episoder | Skifte |
| Løsning | Søge |
| Almindelige opgaver | Knyttet |
| Streamingtjenester | Udleveres |
| Bruger | Tænkes |
| Krigsfilm | Tilføjes |
| Drama | |
| Liste | |
| Billeder | |
| Zip-fil | |
| Features | |
| Rækkefølge | |
| Rating | |
| Barn | |

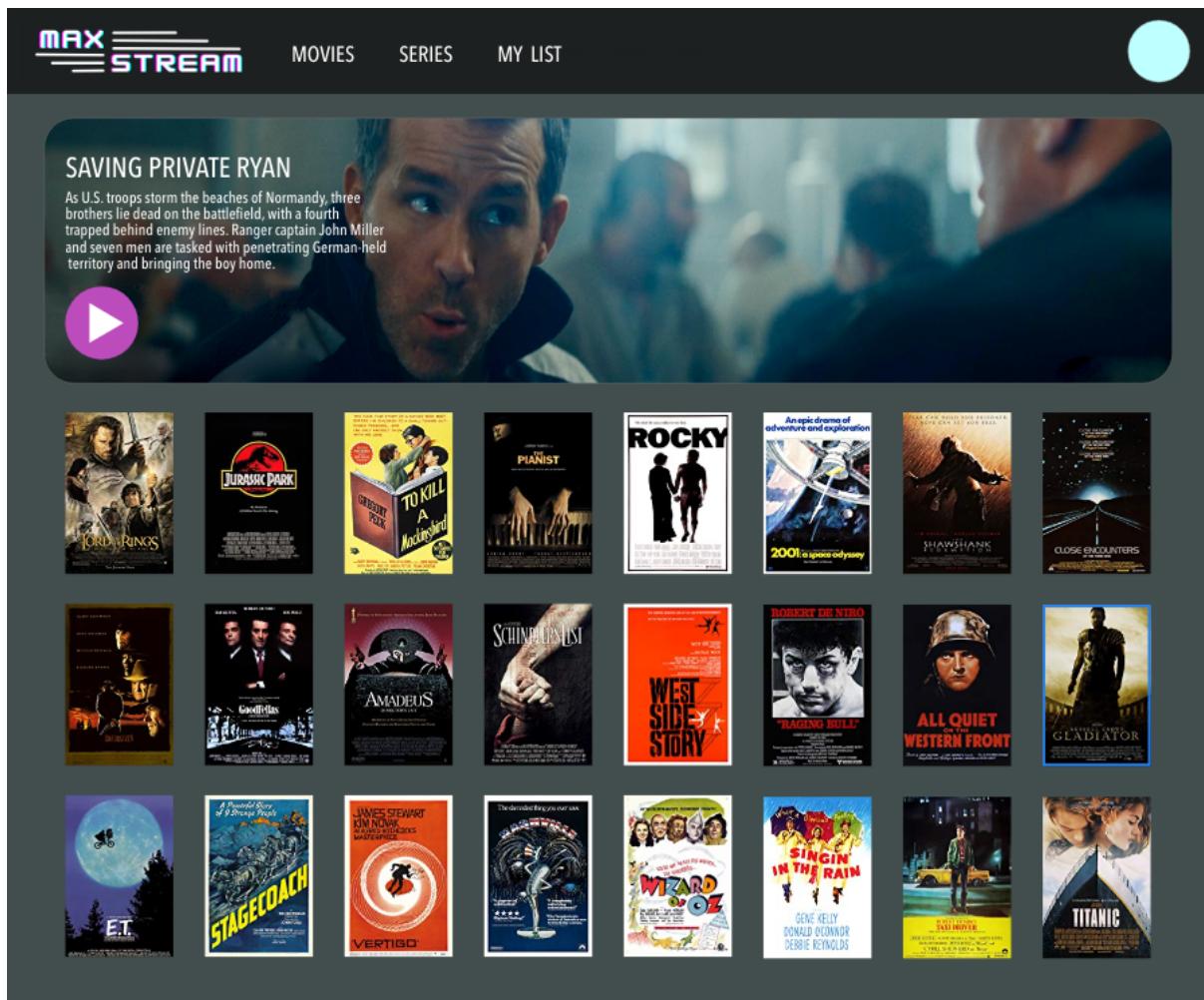
10.3 Første mock-up i AdobeXD



Mock-up af profile-view



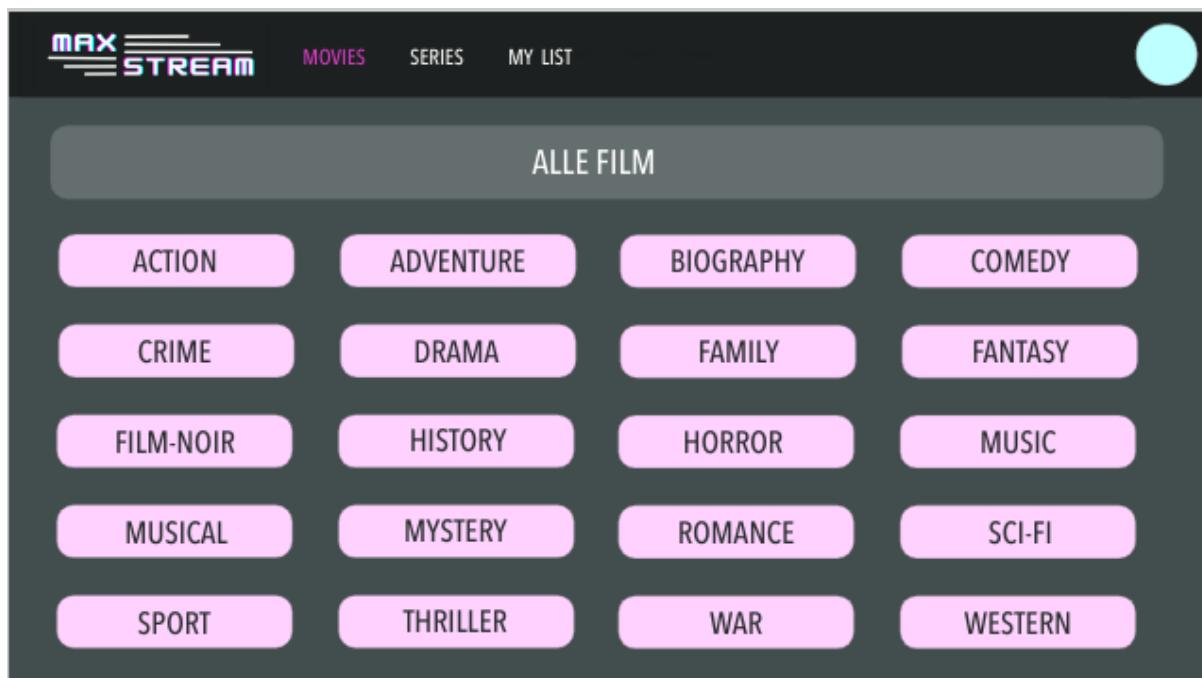
Mock-up af addProfile-view



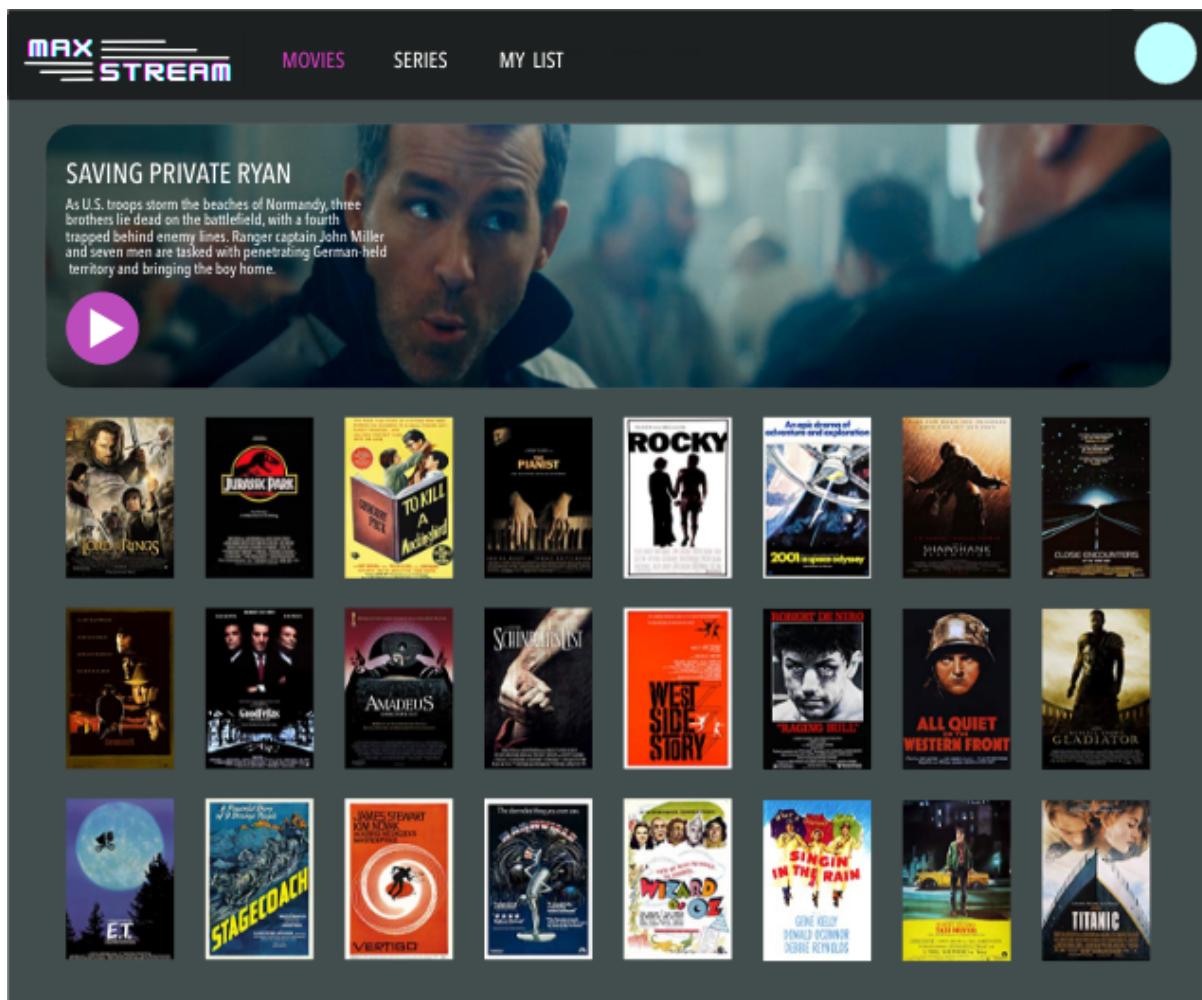
Mock-up af main-view



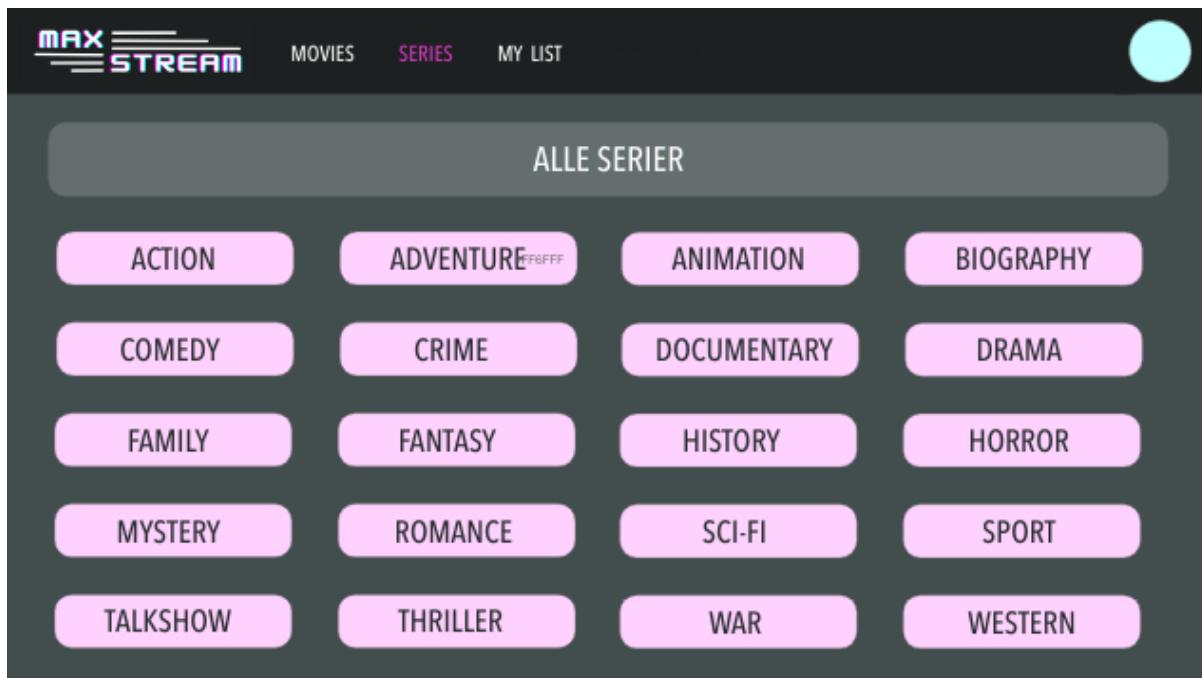
Mock-up af showVideo



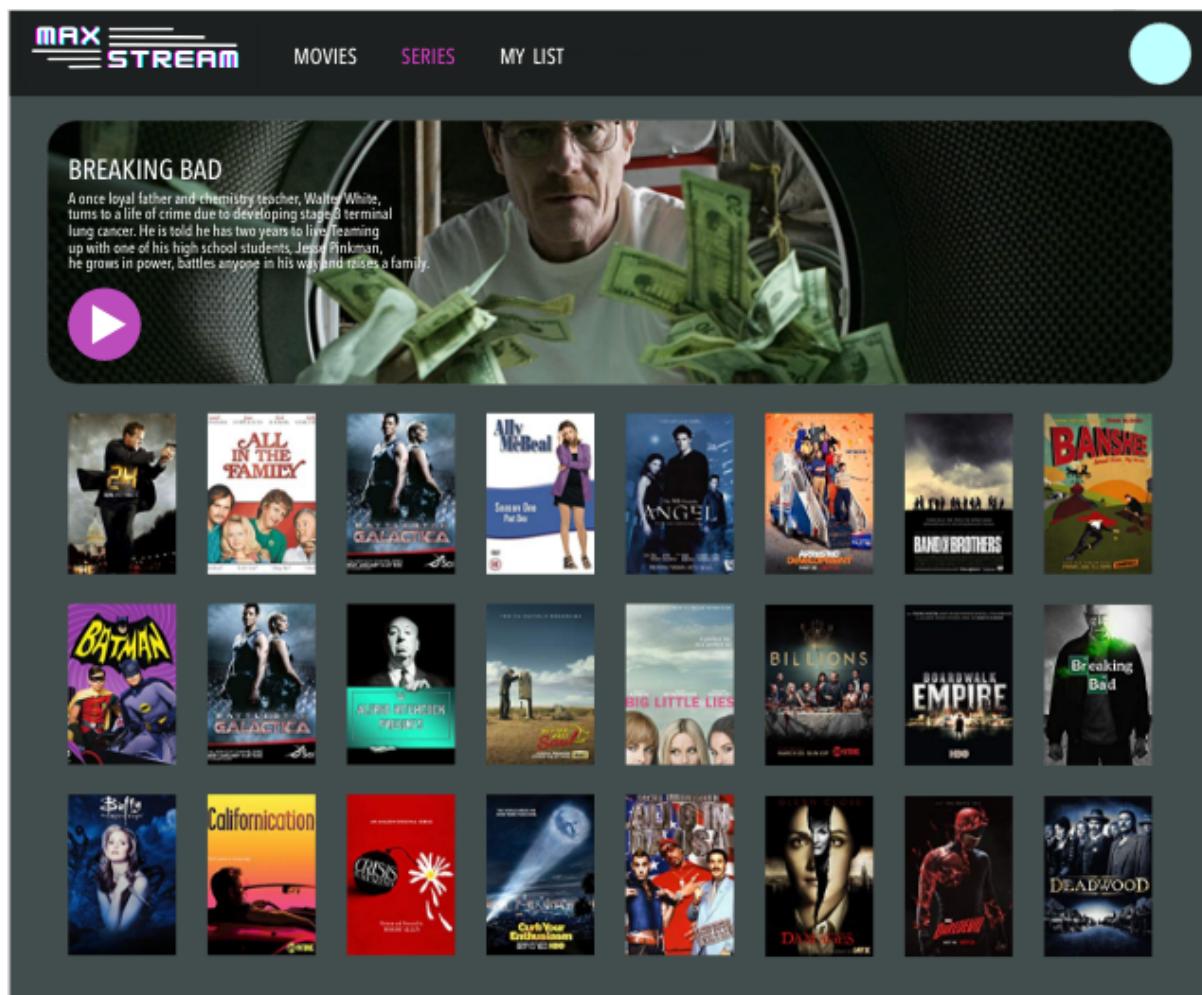
Mock-up af genre display (film)



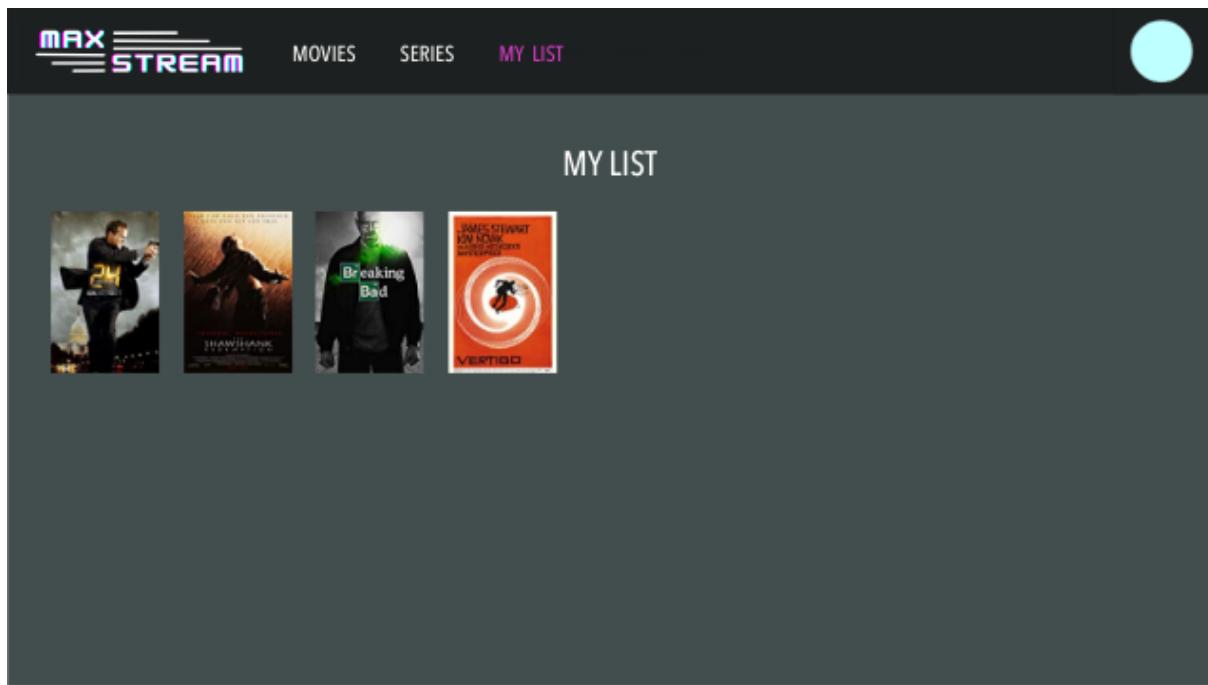
Mock-up af mainMovies-view



Mock-up af genre display (serier)



Mock-up af mainSeries-view



Mock-up af myList-view

10.4 Kildekode til ImageLoader findPath-metode

```
public static String findPath(String fileName) throws NullPointerException
{
    File folder = new File( pathname: "src/main/resources/images/movies");
    File[] listOfFiles = folder.listFiles();

    assert listOfFiles != null;
    for (File file : listOfFiles)
    {
        if (file.isFile() && file.getName().equals(fileName))
        {
            return file.getPath();
        }
    }
    return "No filename found";
}
```

10.5 Unit Test af ImageLoaders findPath-metode

```
public static class myTest
{
    @Test
    public static void getAnyFile(String fileName) throws NullPointerException
    {
        ImageLoader testing = new ImageLoader();
        boolean testSucces = false;

        String filename = null;

        filename = fileName;

        String anyFileName = ImageLoader.findPath(filename);

        System.out.println(anyFileName);

        assert anyFileName != null;
        if(anyFileName.equals("No filename found"))
        {
            testSucces = false;
        } else
        {
            testSucces = true;
        }
        assertTrue(testSucces);
    }
}
```