Project Report

Online Video Streaming Management System

Members (Group 9):

Aydin Arik, Kai-Hsuan Chan, Cynthia Wang, Ruoda Wang, Jeff Zheng

Statement:

As we enter the Web 3.0 Era, the fast and reliable connection has offered us new ways of interacting with our world via smart devices. Especially in the world of entertainment, people are more willing to watch movies and tv shows online nowadays compared to the conventional ways of viewing back in the old days. From youtube to netflix, then hulu and disney +, there are more and more video streaming platforms that are available for us to view different well known titles of movies and tv shows. Our group believes that making an online streaming service management system more important than ever, as there are still countless amount of production companies that would like to dip their feet into this "game". However, since those platforms are not shared to each other, creating a brand new from scratch might take up too much time and effort for them. We would like to tackle this problem with the knowledge we have and provide those new and upcoming companies with a product they can use.

Background Research:

Initially, when we are trying to understand what current online streaming platforms have, we have found out that most of the platforms have ratings of the movies, as long as the user's ratings. These areas are very typical for a system and essential for a management system to have. However, one thing has caught our eyes, recommendations. This inspired us to track down any information or studies that are related to this topic. Luckily, we have found an article that has done an empirical test on the degree of agreement in the appraisal of commercially produced major motion pictures by Pascal Wallisch[1], that also contains data of the user's rating on questions that are related to their characteristics and background info. In this article, it has mentioned that user ratings have a significant relationship with one's characteristics. Our group has believed that this research is really helpful for our topic and we can use this data as information that helps develop an algorithm that can thus predict what users may want to view depending on their information. Then, since some of us also believed that we should look into which type of Interface would be suitable for the system, we then found there are two GUI we can use: javafx, and javax.swing. These two are both already available java interfaces that have been built into the java package. So we considered looking into both of them initially, even though we then chose to use javafx in the end. After initial discussion and the research, we finalized what our management system needs as below:

1. Video Management - add/delete/modify, and view
2. User Management - type of plans, login information and user's rating information
3. Manager Management - their access right and login information

[1] Jake Alden Whritner & Wallisch Pascal, 2017, "Strikingly Low Agreement in the Appraisal of Motion Pictures".

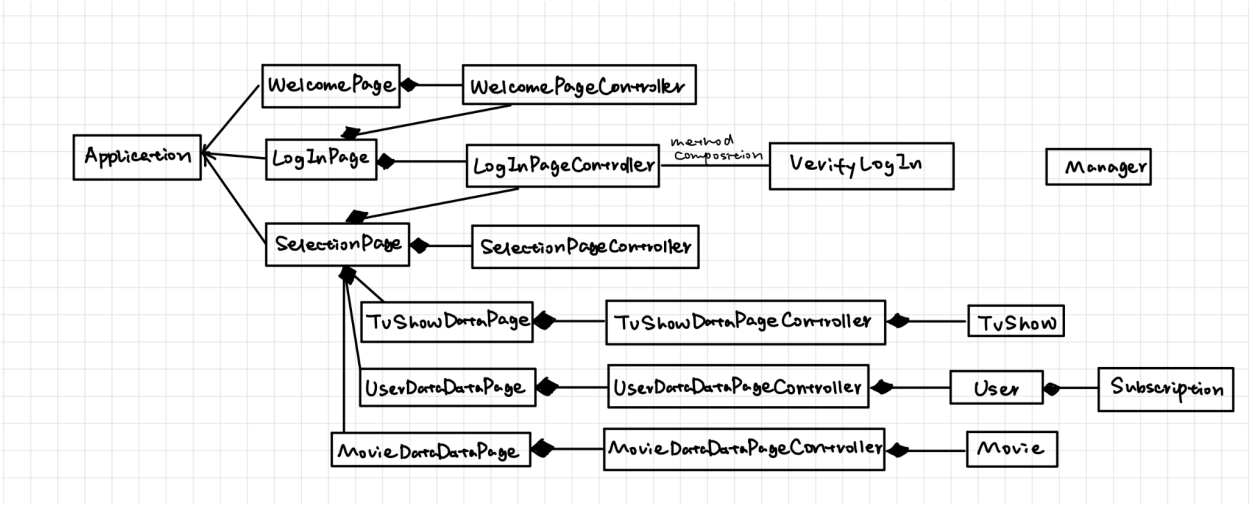
Classes & UML(Design Solutions):

P.S.

1. TvShow, Movie, User data is stored in tvshow_data.txt, movie.txt, user.txt respectively.
2. implementation of controller is visualized through javafx scene builder, shown in fxml files.

Brief descriptions for all methods:

Methods in Movie/TV Shows classes: getters & setters for every single variables;
Methods in User class: getters and setter for all variables, along with object variable Subscription;
Methods in Subscription class: getters and setters for all variables;getAnnualPrice returns monthly price times 12;
Methods in Manager class: only has constructors;
VerifyLogin: verify, check if the username and the password matches the ones in the txt file called manager_data.txt;
Methods and variables in GUI: all generated by building javafx;
back,to(MovieData/TvShowData/UserData)Page are all methods that reroute the javafx to a specific page.
**Exception: InvalidPlanException, if activated, print this is not a valid plan combination;



Main classes

Movie

- year : int
- name : String
- rating : double
- genre : String
+ Movie()
+ Movie(name:String, genre:String, year:int, rating:double, writeData:boolean)
+ Movie(name:String, genre:String, year:int, rating:double)
+ getName() : String
+ getYear() : int
+ getRating() : double
+ getGenre() : String
+ setName(name:String) : void
+ setYear(year:int) : void
+ setRating(rating:double) : void
+ setGenre(genre:String) : void

TvShow

- year : int
- name : String
- rating : double
- genre : String
- episodes int
+ TvShow()
+ TvShow(name:String, genre:String, year:int, rating:double, episodes:int, writeData:boolean)
+ TvShow(name:String, genre:String, year:int, rating:double, episodes:int)
+ getName() : String
+ getYear() : int
+ getRating() : double
+ getGenre() : String
+ getEpisodes() : int
+ setName(name:String) : void
+ setYear(year:int) : void
+ setRating(rating:double) : void
+ setGenre(genre:String) : void
+ setEpisodes(episodes:int) : void

Manager

- id : String
- password : String
+ Manager()
+ Manager(id:String, password:String)

User

- name: String
- password: String
- email : String
- plan: String
- sub: Subscription
+ User()
+ User(name:String, password:String, plan: String, writeData: boolean)
+ User(name:String, password:String, plan: String)
+ getName(): String
+ getPlan(): String
+ getEmail() : String
+ getPassword(): String
+ storeData()
+ setName(name: String)
+ setPlan(plan: String)
+ setEmail(email: String)
+ setPassword(password: String)

InvalidPlanException

+ InvalidPlanException()

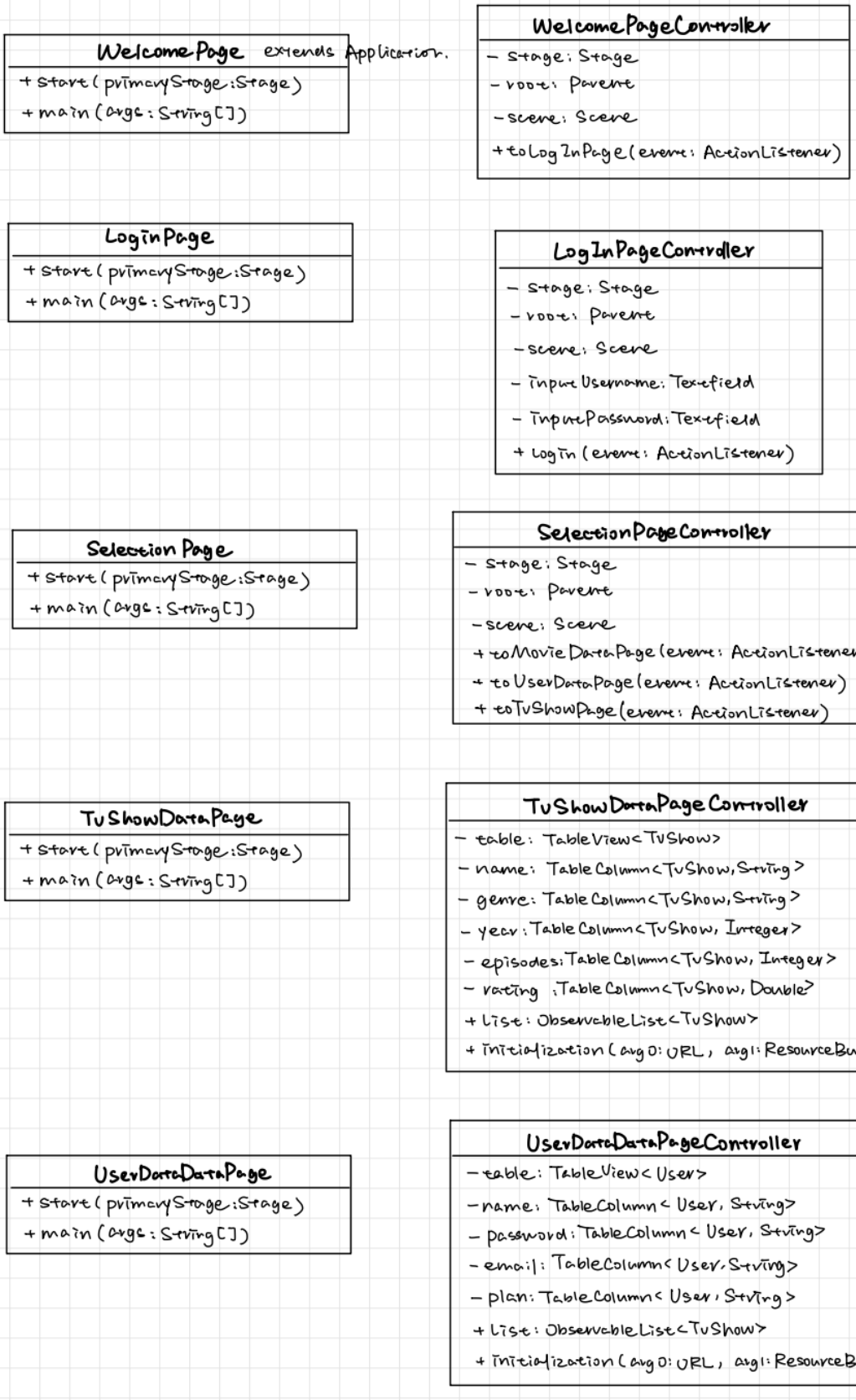
VerifyLogIn

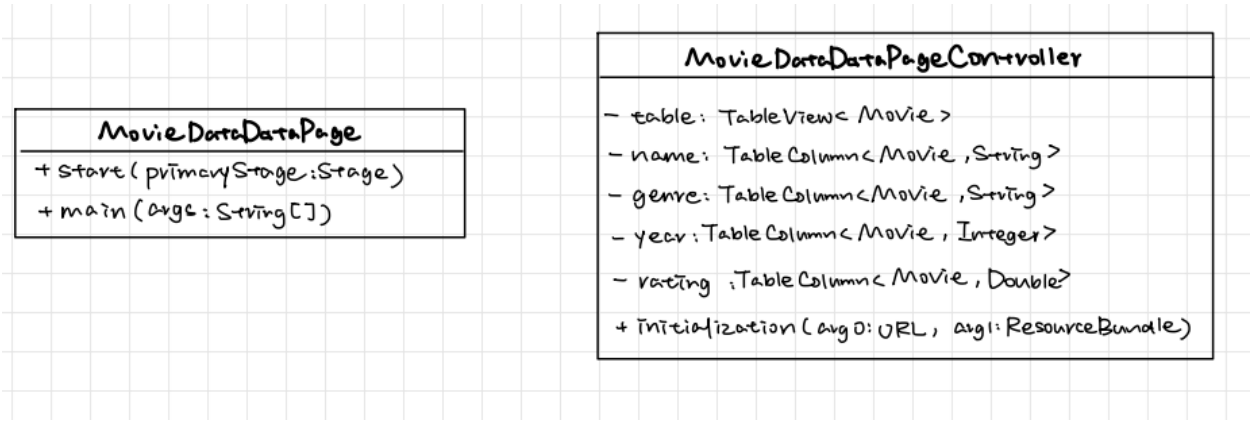
+ filePath: String
+ inputUsername: String
+ inputPassword: String
+ verify(inputUsername: String, inputPassword: String, filePath: String): boolean

Subscription

- monthlyPrice: double
- plan: String
- availableDowoloads: int
- supportedDecices: int
- numberOfSubs : int
+ Subscription()
+ Subscription(plan: String)
+ toString(): String
+ getMonthlyPrice(): double
+ getAnnualPrice(): double
+ getNumberOfSubs(): int
+ getPlan(): String
+ getAvailableDowoloads(): int
+ getSupportedDevices(): int
+ cancelSubscription()
- setMonthlyPrice(price: double)
- setPlan(plan: String)
- setAvailableDownloads(downloads: int)
- setSupportedDevices(devices: int)
+ storeUserData()

GUI:


WelcomePage extends Application.
+ start(primaryStage:Stage)
+ main(args:String[])
WelcomePageController
- stage: Stage
- root: Parent
- scene: Scene
+ toLogInPage(event: ActionListener)
LogInPage
+ start(primaryStage:Stage)
+ main(args:String[])
LogInPageController
- stage: Stage
- root: Parent
- scene: Scene
- inputUsername: Textfield
- inputPassword: Textfield
+ logIn(event: ActionListener)
SelectionPage
+ start(primaryStage:Stage)
+ main(args:String[])
SelectionPageController
- stage: Stage
- root: Parent
- scene: Scene
+ toMovieDataPage(event: ActionListener)
+ toUserDataPage(event: ActionListener)
+ toTvShowPage(event: ActionListener)
TvShowDataPage
+ start(primaryStage:Stage)
+ main(args:String[])
TvShowDataPageController
- table: TableView<TvShow>
- name: TableColumn<TvShow, String>
- genre: TableColumn<TvShow, String>
- year: TableColumn<TvShow, Integer>
- episodes: TableColumn<TvShow, Integer>
- rating : TableColumn<TvShow, Double>
+ list: ObservableList<TvShow>
+ initialization(arg0: URL, arg1: ResourceBundle)
UserDataDataPage
+ start(primaryStage:Stage)
+ main(args:String[])
UserDataDataPageController
- table: TableView<User>
- name: TableColumn<User, String>
- password: TableColumn<User, String>
- email: TableColumn<User, String>
- plan: TableColumn<User, String>
+ list: ObservableList<TvShow>
+ initialization(arg0: URL, arg1: ResourceBundle)



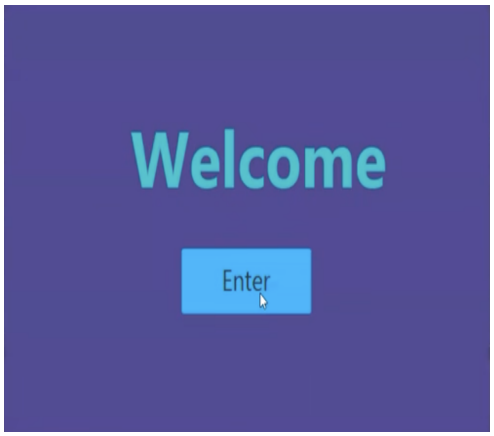
User Guide:

*****A few notes beforehand:**

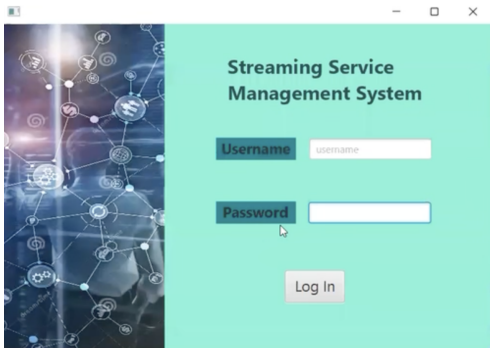
- We have included a detailed README file that is included in the program folder, please read it through and do everything that is needed to set-up before you run the program. Otherwise, it might result in multiple errors and the application might not run.
- Most importantly, in the current stage (as it is still a prototype and a work in progress), an IDLE, along with the javafx package is much needed to use the program.
- All needed txt files might not work on your computer, please relocate the file or change the corresponding code to make sure the program runs properly.

Here is the run through of the program:

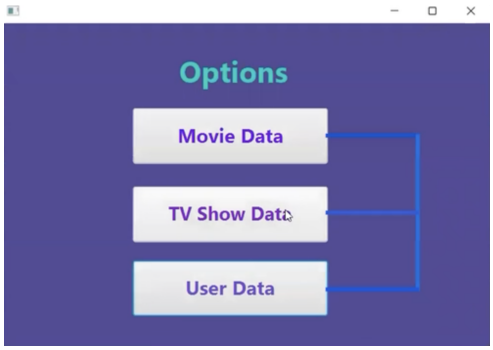
1. The Welcome Page: we first prompt the user with our welcome page, press enter to continue.



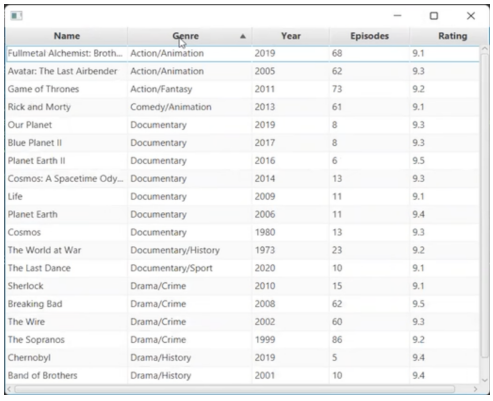
2. The Login Page: Then, we have a login page for the user to enter their ID and Password. Our program validates the username and the password entered in the manager_data.txt file that stores the usernames and passwords. Press Login after you have entered Username and Password, if it is valid, you will be connected to the following page, or **it will show up login-failed in the terminal.**



3. The Main Page: There are three options available, movie data, tv show data, and user data. Press the desirable option and the program will connect you to the corresponding page.



4. Movie/Tv Show/User Data Page: Now, you have the power to sort them by any of the categories. There is no back button, so if you would like to access another data type, you would have to close the application and launch it again.



Name	Genre	Year	Episodes	Rating
Fullmetal Alchemist: Broth...	Action/Animation	2019	68	9.1
Avatar: The Last Airbender	Action/Animation	2005	62	9.3
Game of Thrones	Action/Fantasy	2011	73	9.2
Rick and Morty	Comedy/Animation	2013	61	9.1
Our Planet	Documentary	2019	8	9.3
Blue Planet II	Documentary	2017	8	9.3
Planet Earth II	Documentary	2016	6	9.5
Cosmos: A Spacetime Ody...	Documentary	2014	13	9.3
Life	Documentary	2009	11	9.1
Planet Earth	Documentary	2006	11	9.4
Cosmos	Documentary	1980	13	9.3
The World at War	Documentary/History	1973	23	9.2
The Last Dance	Documentary/Sport	2020	10	9.1
Sherlock	Drama/Crime	2010	15	9.1
Breaking Bad	Drama/Crime	2008	62	9.5
The Wire	Drama/Crime	2002	60	9.3
The Sopranos	Drama/Crime	1999	86	9.2
Chernobyl	Drama/History	2019	5	9.4
Band of Brothers	Drama/History	2001	10	9.4

Reflection:

Overall, we would say that this is a fun and rewarding experience. Implementing all of the knowledge that we learned in this class to make a program that looks like this is truly remarkable. Our group is also very well-coordinated and everyone has put in their best effort to work on this project.

One thing we are glad about is that our group chose to use Github as a way to save all of our works. This kept every process up to date, and much easier to work on for someone to look back into it. All we have to do is to remember to push everything one has worked on and commit it to upload with comments that explain all the changes. This also helped eliminate problems such as different names or storing bytecodes in the wrong format/locations.

The most challenging part of this project is definitely the GUI implementations. As mentioned in the background research, we tried to learn both javax.swing and javafx. At first, we attempted to tackle the GUI design using Javax.swing(as it is built in java and no installation of reference library needed), but we faced troubles implementing it and make it look like what we would want it to look like, so instead, we've switched to JavaFX to make the GUI. However, since javafx is no longer included in the current stage of Java, we have to learn how to install it into the system. This process took us at least an entire weekend each to get it work correctly, and then we can start working on the different pages that are now presented to the user.

Currently, there are still a few progress we would like to make so the program is running as we would have initially imagined as it is still a prototype. Firstly, is to learn about how to package all the windows into an application that you can use without using the idle to start the program. Secondly, it is for us to implement the article data that we found during research so we can then create this algorithm that can suggest new movies and tv shows for the user. Thirdly, we would like to add buttons that allow for creating and modifying new users/tv shows/movies in the future. Lastly would be to learn how to automatically get all the information from sites like IMDB so we can have the most up-to-date ratings for each and everything movie that is for this management system.

Contributions

Aydin Arik: Graphic User Interface, Presentation, and Coding
Kai-Hsuan Chan: Brainstorming Idea, Coding, Graphic User Interface, Presentation, Demo Video, and Structure Clean-Up
Cynthia Wang: Coding, UML, and Coordination
Ruoda Wang: Initial Structure Coding, Project Report, and Presentation
Jeff Zheng: Initial Set-up (github), Preliminary research, Demo Video, Project Report, Coordination, Coding, Final Clean-Up