

TV Programme Scheduling System

Team roles-

- Leader: Erik M00866654
- Secretary: Justin M00912380
- Developers: Sebastian M00852517, Lei M00968419
- Tester: Arfanul M00915474

Introduction -

Develop a program scheduling application utilising arrays or vectors for data storage. Consider the option of applying a CSV file for program storage, or alternatively, stick to using arrays. Determine whether to establish fixed programming slots for specific shows or opt for a randomised scheduling approach while maintaining fixed time slots. Additionally, the capability to insert commercial breaks between programs.

Objectives-

- Develop an extensive TV Programme Scheduling System capable of managing multiple channels and diverse programming.
- Implement diverse scheduling algorithms to optimise program placement based on viewer preferences, genre, and advertising revenue
- Provide a user-friendly interface for scheduling managers to efficiently manage program schedules and make data-driven decisions.

Design features & functionality -

Features that can be made:

- Record function - Allows users to save their favorite programs.
- Save to favorites -
- Show program time/dates - Display when programmes are scheduled
- Pause/rewind - Enhance user control over live Tv.
- Parental controls - Enables restricting content based on age suitability

Functions that may be made (harder/longer side)

- Category list - store specific genres together that the user can call
- Create multiple channels with their own program schedule that the user can switch to
- User can have their own account (might not be needed)
- Advanced Search: Implement robust search functionality allowing users to search for programs based on various criteria such as genre, actors, directors, or keywords.
- Multi-room viewing -
- Optimise the allocation of time slots based on factors like viewer demographics, popularity of programs, genre diversity, and advertising revenue potential.

Potential classes - Program class: Justin

- Get Program ID
- Get and show program Name/Title
- Show how long a program lasts
- What channel the program is in (If we create multiple channels)
- Is the program saved by the user/made favorite. (Can be done using a boolean)

Schedule Class: Erik

- If creating multiple schedules (not identical as well) then the system could grab the schedule ID
- Display the schedule of a channel, duration of a program, commercial breaks in the schedule and display the program name that will be on
- Create schedule
- Display schedule
- Make it daily / weekly

End of class functions:

- Add timer to display programme and then move to next programme

Channel class (Optional): Arfunal

- Display channel Name
- Show the shows/programs in channel
- Grab and call the channel ID

Saved/Favorites class: Sebastian

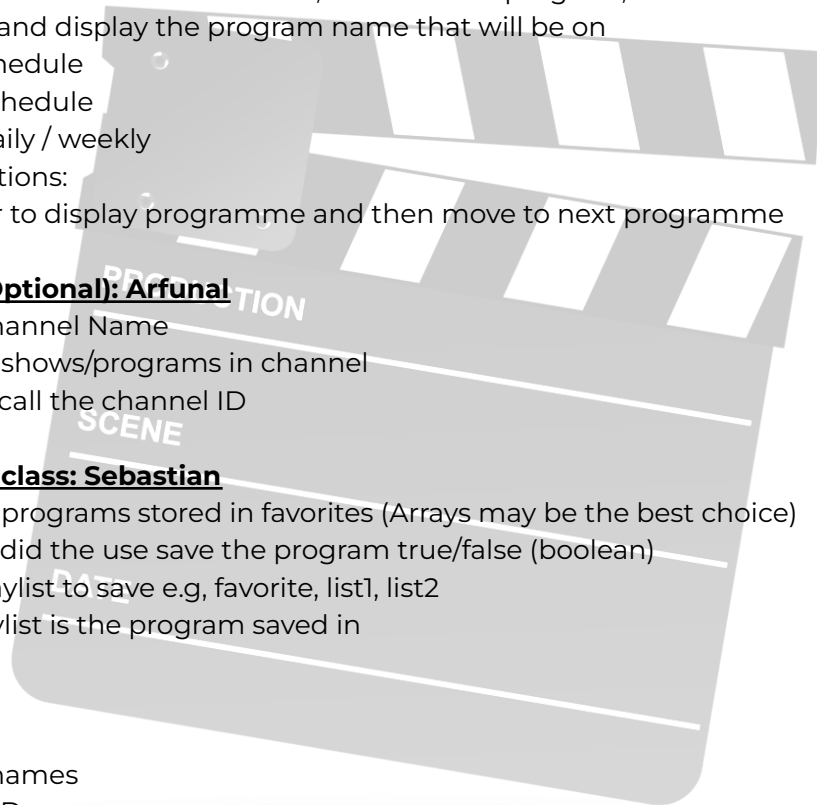
- Show the programs stored in favorites (Arrays may be the best choice)

Start functions: -did the use save the program true/false (boolean)

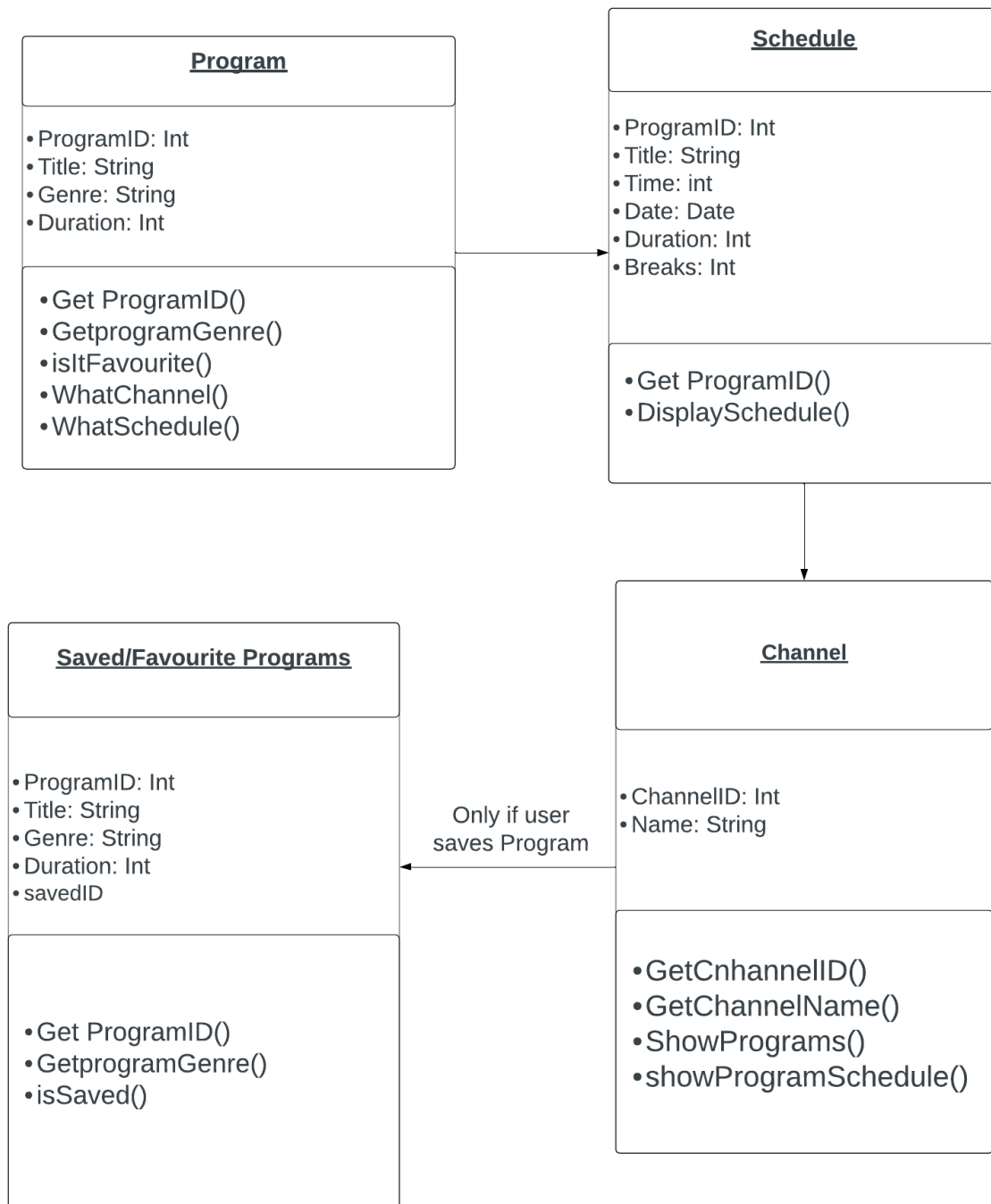
- Create playlist to save e.g, favorite, list1, list2
- What playlist is the program saved in

CSV:

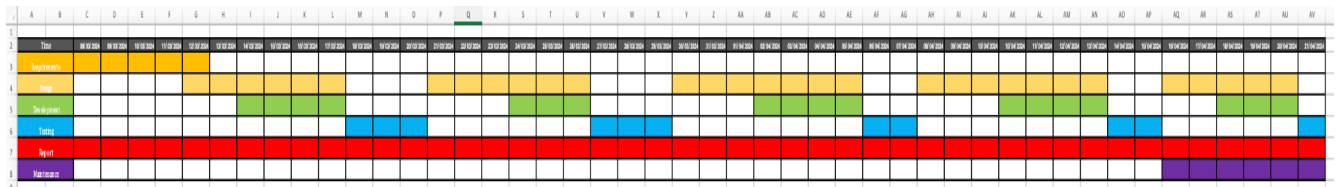
- Genres
- Channel names
- Program ID
- Duration



UML Designs -



Time Management -



Duration - 08/03/2024 - 21/04/2024 (Grey bar at the top)

Requirements, Design, Development, Testing, Report and maintenance- (y axis)

Time management was crucial since we wanted to get things done out of the way and make tasks easier as we get closer to submission date. The way we managed our group project was at the start we gathered as a group to deliberate what our system was going to require and perform. Then it was a repetitive process of developing our system and making adjustments if needed. In addition, as we got closer to the submission date, until we were all satisfied with our project, all we had to do was maintain the code. As well as a last test.

Start of the project:

We set up the main files, after running the main file I started to see if I could break the programmed. The main file broke after I inputted a string instead of a number when inputting the menu. We also tested the Makefile to see if the files would connect, first few times it wasn't setting up properly. Then fixed it and managed to link the classes.

We discussed the design in further detail, we created a couple of example codes and I tested them. I tried breaking them by different means of inputting and testing, usable code was shared among each other- however nothing was finalised to push to github. We just kept working on the code that was partially usable.

We also ran a couple tests to see whether or not our code is working. We did come across some errors as we mentioned in the testing/validation section.

Middle of the project:

Made improvements to the project, where it was able to read the csv file correctly, but still had issues reading the time. Despite this, we managed to add a menu when initial startup. Although it was very simple and would work if we entered a letter instead of a number. We knew we had to make readjustments to our design that tackled this situation which Lei has done.

End of the project:

With all things considered, we managed to get what we wanted. All we needed to do was maintain the code such as making sure the formatting was correct and seeing if the programs are running in the correct order. As well as finishing off the report.

Testing and validation -

Our first initial test:

```
[Running] cd "c:\Users\erikc\Documents\Comp sci\C++\Assignments\T
++\Assignments\TvScheduleProgram\"schedule
Channel  Program                      Start Time  End Time
-----  -
Genre    Program ID      00:0000:00
Comedy    1                00:0000:00
Documentry2  3                00:0000:00
Music     4                00:0000:00
Sport     5                00:0000:00
Drama     6                00:0000:00
Comedy    7                00:0000:00
Documentry7 8                00:0000:00
Family friendly8 00:0000:00
Action    9                00:0000:00
Music     10               00:0000:00

[Done] exited with code=0 in 1.928 seconds
```

Semi-completed - We had trouble reading the csv file and calculating the time.

Displaying Channel names and start times:

```
Channel: Channel
, Program: Name, Start Time: 06:00, End Time: 06:00
Channel: Channel x
, Program: WorldDishes, Start Time: 06:05, End Time: 07:06
Channel: Disney
, Program: Cooking with Justin, Start Time: 07:11, End Time: 29:33
Channel: Disney
, Program: laddington, Start Time: 29:38, End Time: 30:39
Channel: High
, Program: Jurrasic Park, Start Time: 30:44, End Time: 31:45
Channel: SportsX
, Program: formula 1 , Start Time: 31:50, End Time: 72:30
Channel: Vevo
, Program: party wit boogie, Start Time: 72:35, End Time: 92:55
Channel: Vevo
, Program: Dance wit me, Start Time: 93:00, End Time: 113:20
Channel: high
, Program: Arf's vlog , Start Time: 113:25, End Time: 154:05
Channel: high
, Program: Canibal charly, Start Time: 154:10, End Time: 155:11
Channel: laugh central
, Program: Top fails, Start Time: 155:16, End Time: 175:36
```

We were able to display all the start times of all the shows however, there was an issue with the hours being read as minutes which we needed to handle as quickly as possible.

Corrected hours and minutes:

```
, Program: Dance wit me, Start Time: 06:00, End Time: 06:00
, Program: Name, Start Time: 06:05, End Time: 06:05
, Program: WorldDishes, Start Time: 06:10, End Time: 06:10
, Program: Cooking with Justin, Start Time: 06:15, End Time: 06:15
, Program: laddington, Start Time: 06:20, End Time: 06:20
, Program: Jurrasic Park, Start Time: 06:25, End Time: 06:25
, Program: formula 1 , Start Time: 06:30, End Time: 06:30
, Program: party wit boogie, Start Time: 06:35, End Time: 06:35
, Program: Arf's vlog , Start Time: 06:40, End Time: 06:40
, Program: Canibal charly, Start Time: 06:45, End Time: 06:45
, Program: Top fails, Start Time: 06:50, End Time: 06:50
PS C:\eriksschedule\TvScheduleProgram-scheduleBranch>
```

Resolved the issue with the hours being displayed as minutes. bUt it didn't get the correct minutes from the csv file.

```

Channel: laugh central, Program: Slow minute, Start Time: 06:00, End Time: 07:00
Channel: Channel
, Program: Name, Start Time: 07:05, End Time: 07:05
Channel: Channel x
, Program: WorldDishes, Start Time: 07:10, End Time: 08:10
Channel: Disney
, Program: Cooking with Justin, Start Time: 08:15, End Time: 08:37
Channel: Disney
, Program: laddington, Start Time: 08:42, End Time: 09:42
Channel: High
, Program: Jurrasic Park, Start Time: 09:47, End Time: 10:47
Channel: SportsX
, Program: formula 1 , Start Time: 10:52, End Time: 11:32
Channel: Vevo
, Program: party wit boogie, Start Time: 11:37, End Time: 11:57
Channel: Vevo
, Program: Dance wit me, Start Time: 12:02, End Time: 12:22
Channel: high
, Program: Arfs vlog , Start Time: 12:27, End Time: 13:07
Channel: high
, Program: Canibal charly, Start Time: 13:12, End Time: 14:12
Channel: laugh central
, Program: Top Falls, Start Time: 14:17, End Time: 14:37

```

time is working now, hours and minutes being added perfectly, commercial break is being added twice(fix)

Start-up menu:

```

**** Menu ****
1. See channel list
2. Switch to channel
3. See favourites
0. Exit the program
Enter your choice > 1
Calling function to see channel list
**** Menu ****
1. See channel list
2. Switch to channel
3. See favourites
0. Exit the program
Enter your choice > 2
Calling function to switch to channel
**** Menu ****
1. See channel list
2. Switch to channel
3. See favourites
0. Exit the program
Enter your choice > 3
Calling function to see favourites
**** Menu ****
1. See channel list
2. Switch to channel
3. See favourites
0. Exit the program
Enter your choice > 4
Invalid choice, please try again
**** Menu ****
1. See channel list
2. Switch to channel
3. See favourites
0. Exit the program
Enter your choice > 0
Exiting program...
PS C:\Norden\project\3anal\wdid_hb\formal2022\电视机顶盒系统\1\1\cheshu\program>

```

This is what our startup will look like when you run the program. It has 4 options; 'See channel list', 'switch channel', 'see favourites' and 'exit'.

Finalised list of programs and run times

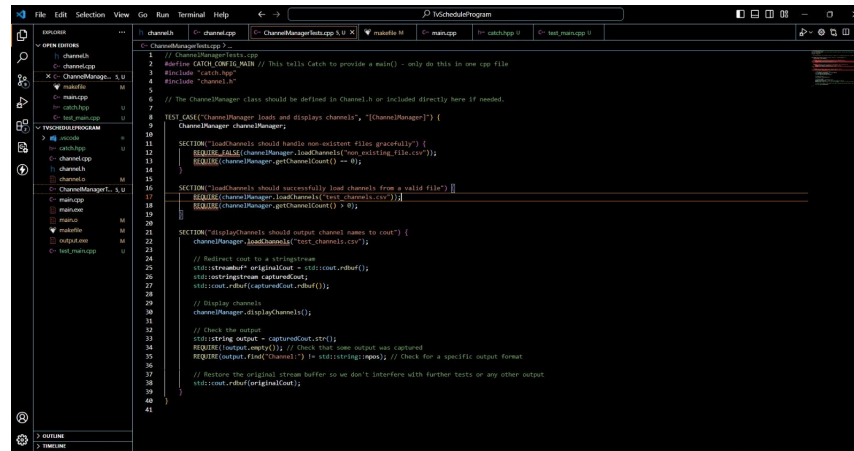
```

Schedule 2
Channel: Channel x
, Program: Deep Sea Documentaries, Start Time: 06:00
Channel: high
, Program: Classic Cartoons, Start Time: 07:05
Channel: high
, Program: Detective Series, Start Time: 07:40
Channel: Channel x
, Program: WorldDishes, Start Time: 08:30
Channel: Channel x
, Program: History Uncovered, Start Time: 09:35
Channel: high
, Program: Action Movie Classics, Start Time: 10:35
Channel: high
, Program: Sci-fi Universe, Start Time: 12:40
Channel: Disney
, Program: laddington, Start Time: 13:45
Channel: SportsX
, Program: formula 1 , Start Time: 14:50
Channel: laugh central
, Program: Top falls, Start Time: 15:35
Channel: high
, Program: Arfs vlog , Start Time: 16:00
Channel: high

```

This is 1 of 3 schedules, and all three schedules have a unique run time for each programme that run from 6am to 10pm with 5 minute commercial breaks in between.

Catch 2 file:



```
1 // ChannelManagerTest.cpp
2 #include "ChannelManager.h"
3 #include "catch.hpp"
4 #include "Channel.h"
5 // The ChannelManager class should be defined in Channel.h or included directly here if needed.
6
7 TEST_CASE("ChannelManager loads and displays channels", "[ChannelManager]") {
8     ChannelManager channelManager;
9
10    SECTION("loadChannels should handle non-existent files gracefully") {
11        REQUIRE_FALSE(channelManager.loadChannels("non_existent_file.csv"));
12        REQUIRE(channelManager.getChannelCount() == 0);
13    }
14
15    SECTION("loadChannels should successfully load channels from a valid file") {
16        REQUIRE(channelManager.loadChannels("test_channels.csv"));
17        REQUIRE(channelManager.getChannelCount() > 0);
18    }
19
20    SECTION("displayChannels should output channel names to cout") {
21        channelManager.loadChannels("test_channels.csv");
22
23        // Redirect cout to a stringstream
24        std::stringstream originalCout = std::cout.rdbuf();
25        std::stringstream captureCout;
26        std::cout.rdbuf(captureCout.rdbuf());
27
28        // Display channels
29        channelManager.displayChannels();
30
31        // Check for output
32        std::string output = captureCout.str();
33        REQUIRE(!output.empty()); // Check that some output was captured
34        REQUIRE(output.find("Channel") != std::string::npos); // Check for a specific output format
35
36        // Restore the original stream buffer so we don't interfere with further tests or any other output
37        std::cout.rdbuf(originalCout.rdbuf());
38    }
39}
```

Attempt to read other files with the catch2 file

Future adaptations -

1. Artificial Intelligence Enhancements:

- Predictive Scheduling: Use machine learning algorithms to analyze historical viewership data and predict future trends. This can help in automatically adjusting the schedules to maximize viewer engagement and advertising revenue.
- Automated Content Recommendation: Implement AI to suggest programming that complements the current schedule based on viewer preferences and past performance, optimizing both audience satisfaction and channel performance.

Conclusion -

The TV Programme Scheduling System is designed to make it easier for TV networks to organize their programs effectively. By using the system, The system is built to be easy to use, helping those in charge of scheduling to do their jobs better and faster. Looking ahead, we planned to add new features that will make the system even more useful, such as a cloud intergration and using AI to better predict what viewers want to watch. In the end, the system allowed the user to see the list of programmes available, display the run time of each program and when it's scheduled.