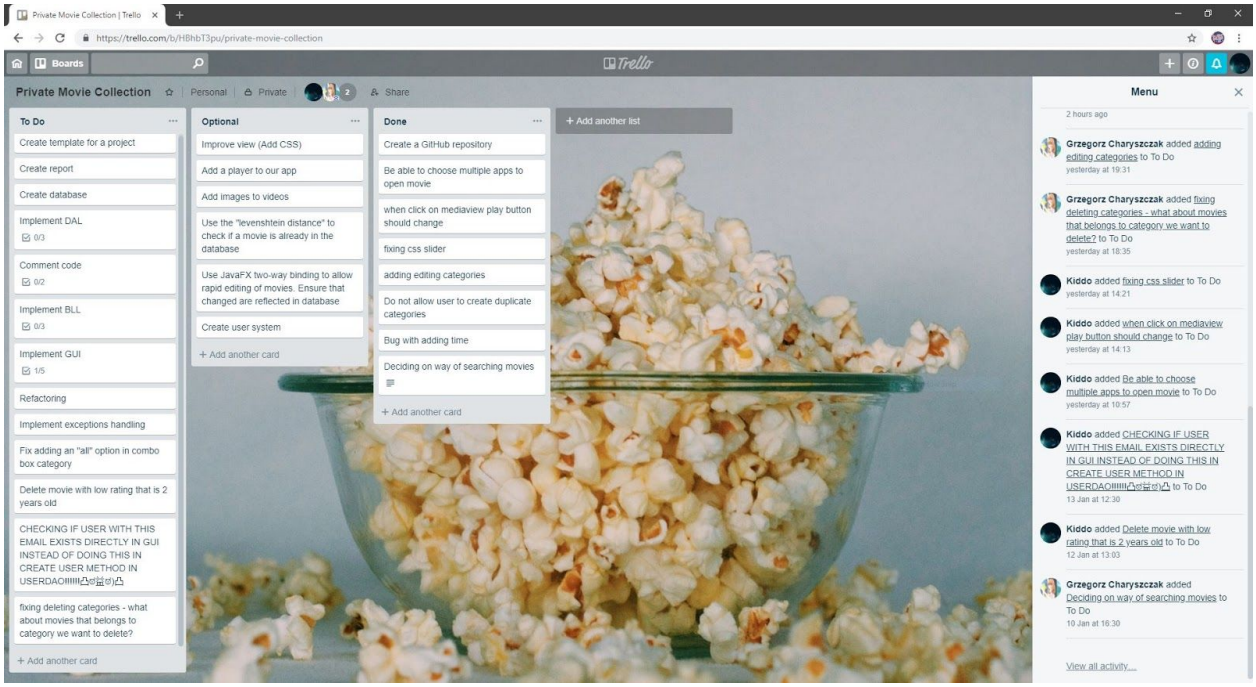
PRIVATE MOVIE COLLECTION

Group: Greg 2: The greggening
Team Members: Greg Charyszczak , David Kalatzis, Matti Brandt

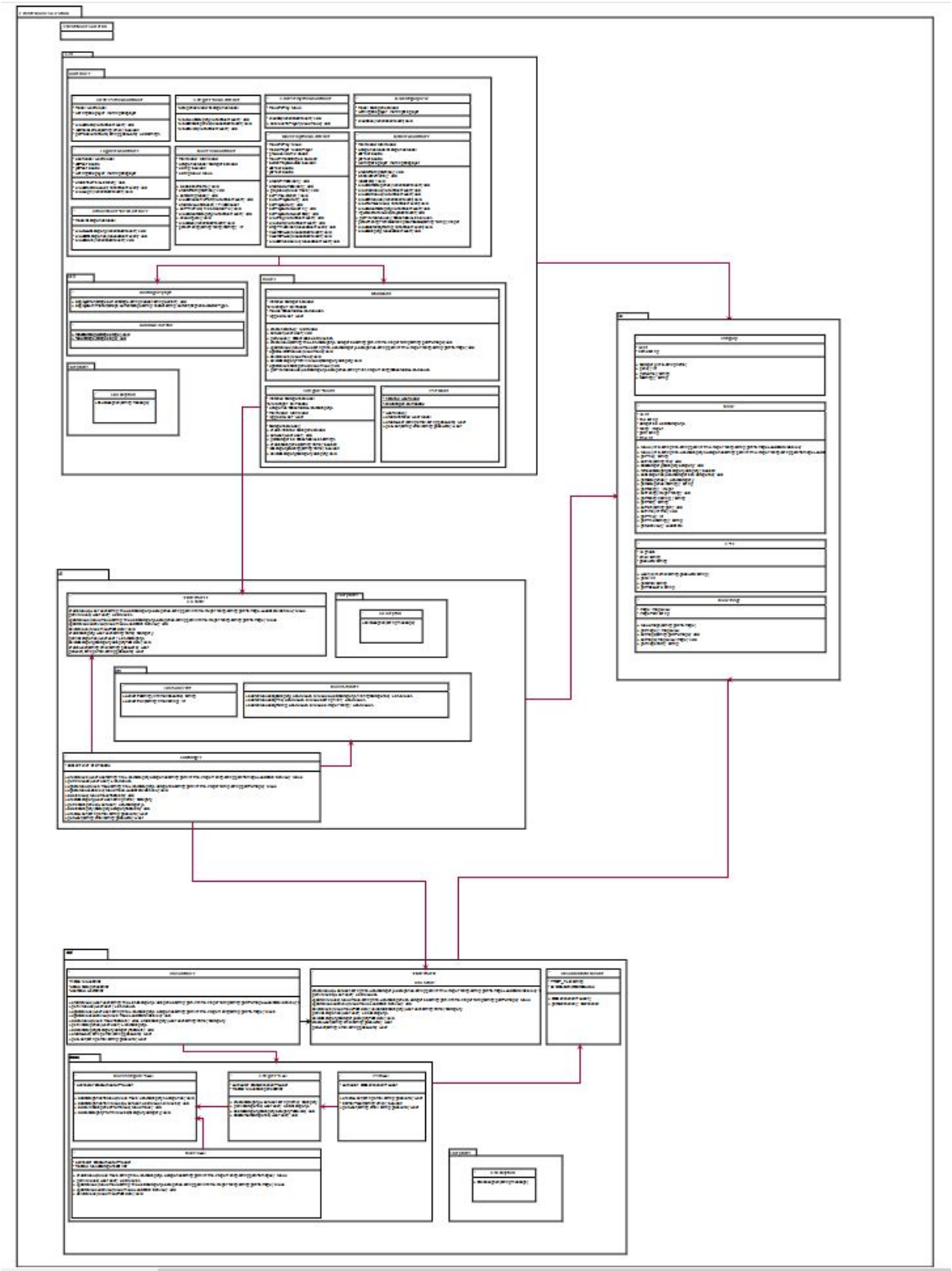
Github repository: https://github.com/schemabuoi/Private-Movie-Collection

REQUIREMENTS LIST

In order to organize a list of the requirements that we will have to complete for the assignment, we used Trello to create a simple and readable list of categorized objectives, and keep track of it throughout the project, as well as add or remove anything we find as we go accordingly.



Class Diagram

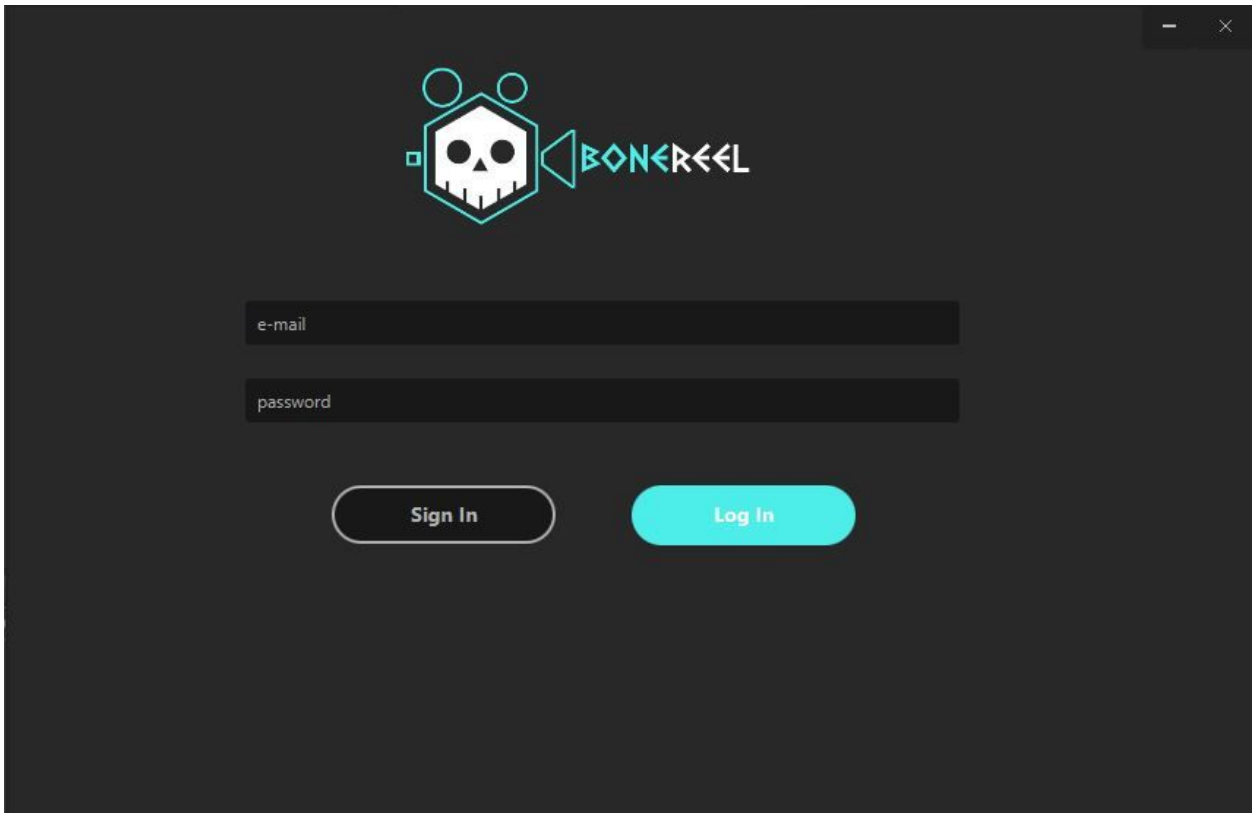


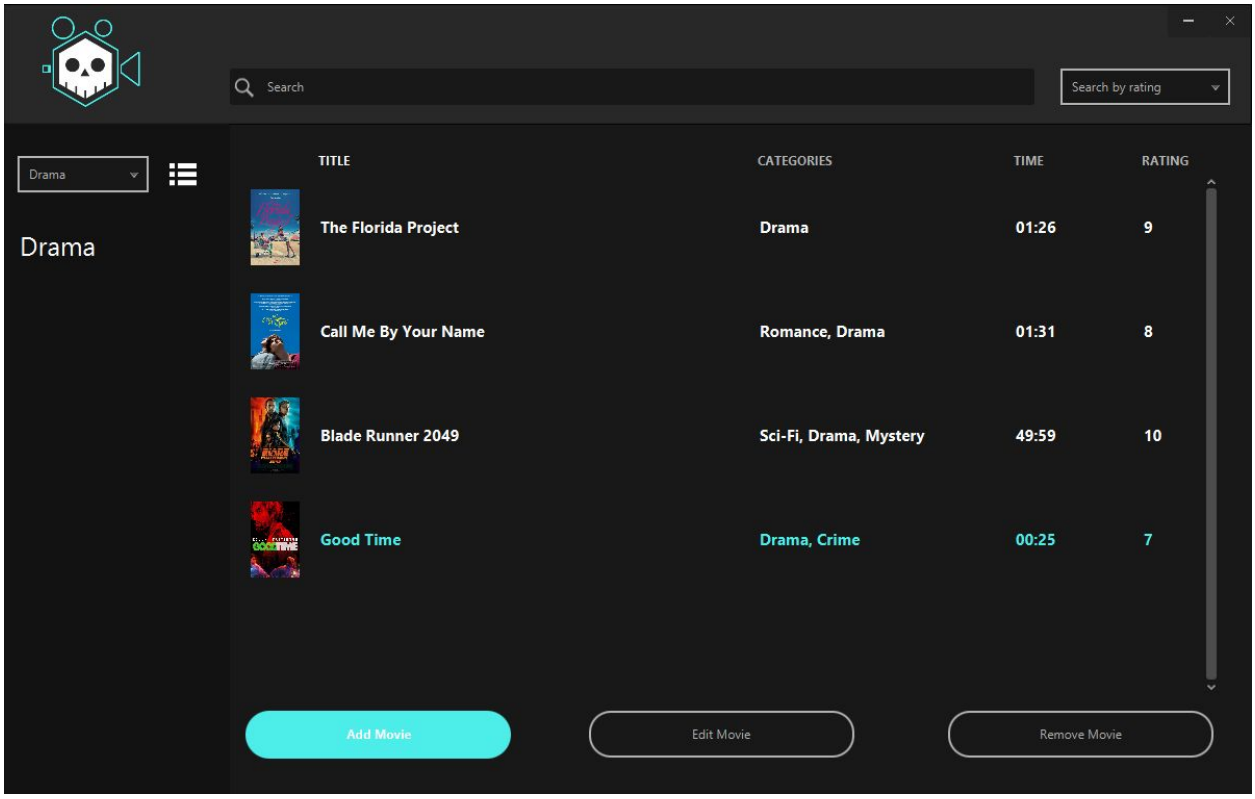
The full XML is uploaded on github.

USER INTERFACE

The user interface was one of the first things we started working on since we had need of some basic view to test any kind of function, later on when most compulsory tasks where done we started working on a better user interface with the use of CSS, and once it was fully done we replaced it and removed the old one.

With the finished system, our user interface includes two views, one for logging the user in, and for creating an account.The main view where most operations are done, such as
searching through movies, adding deleting and editing, as well as starting the media player.One view shared for adding and editing movies, with another one for adding categories, and lastly the view for our own media player.




Search
Search by rating
Drama
Drama
TITLE
CATEGORIES
TIME
RATING
The Florida Project
Drama
01:26
9
Call Me By Your Name
Romance, Drama
01:31
8
Blade Runner 2049
Sci-Fi, Drama, Mystery
49:59
10
Good Time
Drama, Crime
00:25
7
Add Movie
Edit Movie
Remove Movie


Over The Garden Wall
00:23 / 03:04

3-LAYERED ARCHITECTURE

Our approach for the 3-layered architecture started from the very beginning of the project, we created the basis of all presentation, logic and data folders and how they should be structured.

The reasoning for that is that it would be much easier to follow this mindset leading forward, so now the architecture of the system is working as intended, meaning that the layers influence each other in an ascending order and it is only possible to interact differently through the business entities.

HANDLING EXCEPTIONS

In order to handle the exceptions in our program, we added exceptions for every layer and wrapped all checked and some unchecked exceptions into exceptions specific to
the application.

Every exception that occurs is thrown to one of the controllers, where we are displaying the exception error using our WarningDisplayer class. When an exception occurs, all layers catch the exception from the lower level, and wrap it into an exception specific for this level to later throw it up until it reaches the controller.

Using this philosophy, we are decoupling our layers from each other, for example, even if the GUI layer receives an exception from the DAL layer about the database connection, the exception will be wrapped into a BLL one, so that the GUI layer stays unaware of the DAL layer,without any connection between them.In that way we are keeping with the rules of the 3-layered architecture.

Controllers are also completely decoupled from BLL, since every connection occurs through the model, this design is kept with exceptions as well: Exceptions catched in models used in the BLL layer are thrown to the controllers wrapped as model exceptions.

We also used exceptions to prevent the application from crashing at some critical points (Trying to log in with a wrong email or password, trying to play after a file path change, without updating it inside the program)

SYSTEM TEST

We thoroughly analyzed our program with the use of test data, mainly the form of that data would be hardcoded code, since that is an easier and faster approach to check if the results are the ones we hoped for, and it was mostly during database tests that we had to generate test data through the database for confirmation.

When everything was finished we did a couple tests of the whole system, with test data that would resemble that of a user, to make sure of any errors that we haven't noticed yet and fixed the small issues that occured.

ADDITIONAL WORK

- **User system.**
  A simple system that allows the creation of a user account, so that each
  user will have their own collection of movies.
  The creation of the account is done by writing an email and password, with a validation to check whether the password is secure enough.
- **Default categories**
  Everytime a user account is created, we are adding some default categories, to help the user with the hassle of adding basic genres.
- I**mproved user interface**
  An improved user interface done with the use of CSS and a specific theme and set of colors to keep a coherent style through the program.
- **Choosing preferred media player**
  Allowing the user to choose their own prefered media player, with the option being between the basic windows media player, and our own custom made one.
- **Images for movies**
  Images added by the user are shown besides the movie on the view.
- **Our own media player**
  A media player made specifically for this program to fit the theme.
- **Util packages**
  Separated packages for the functional classes, from the gui and bll.

- **Our media player functionality**

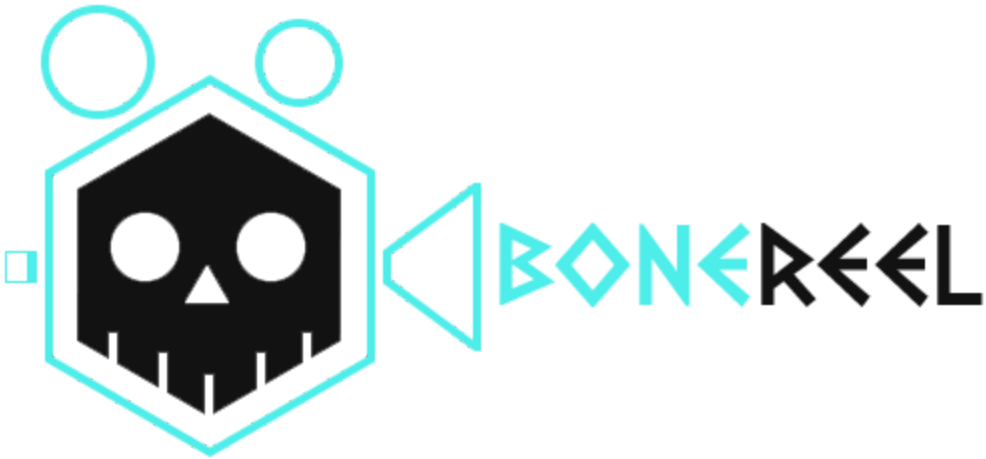
  A time slider showcasing the current time and end time of the movie through a slider and two labels.

  The user is able to drag the slider track to a specific point, and the current time label will change dynamically, otherwise they can also click on that specific point on the slider.

  A fullscreen button resizing the window to fullscreen mode, and entering windowed mode again on second click.
- **Warning displayer**

  A warning will display whenever the user is performing an action that requires the user's confirmation, or if an error occurs to let the user know.
- **Fading windows**

  A subtle fading in effect for the windows in the background whenever a new pop up window appears to give a bigger contrast between background and foreground, this effect also appears at the start and logging in of the app to give a nicer feel.
- **Logo and name for application**
  - A logo and name for the application to make it stand out.


BONEREEL