Team: μ-1 *(mjuː·ˈmaɪnəs·wʌn)*

Project: Kinojecture **Deadline: May 31st**

A Qt-based application with inspiration taken from the well-known family game "Guess Who?" with a focus on cinema-related personalities and pieces of media, using IMDb Non-Commercial Datasets as the data source.

Target audience

Our target market for this project is comprised of people who generally enjoy watching movies - this group, as a whole, would potentially be interested in using this project. The target audience, however, will be movie enthusiasts.

The problem

Those with a keen interest in a certain field may often be tempted to compare their level of knowledge against someone else's. There are a variety of such solutions in many different mediums - be that movie-centred TV shows, YouTube video essays on certain pieces of cinematography or, indeed, competitive games. Our application falls into the latter category.
By playing our game, users will be able to test how good they are at asking "Yes-or-No" questions which would be the most efficient in terms of guessing a specific movie, actor, director or composer. This will, most likely, aid them in memorising cinematography-related trivia, which might be useful as a conversation starter at social gatherings.

The solution

UX Solution (Views):

SB	IF	RB	CB	DL	SL	TVS	TVC
"Submit" button	Input field	Radio button	Checkbox	Droplist with multiple choices + search	Slider	Table view, selection with limit	Table view, click trigger

Menu:

The game opens to the Menu view.
Possible player actions on this view:
> SB - Start a game (button press or press [G]) - the player enters a game creation screen
> SB - Connect to an over-the-network game (button press or press [N]) - the player enters a list view with all available in-network games.
> SB - Enter leaderboard (button press or press [L]) to see all the users you've competed with locally or over the network and see their scores at the time of your last competition.

> SB - Enter settings (button press or press [S]) to control the colour scheme of the game board, change music and sound volume. Additionally, there may be a nickname modification box, but nicknames are never displayed without an additional ID during multiplayer gameplay, as they are not guaranteed to be unique.
> SB - Enter credits (button press or press [C]) to view the credits

Create a game:

Possible player actions on this view:

> SB - switch to preset view
> SB - switch to custom view

Custom:

Possible player actions on this view:

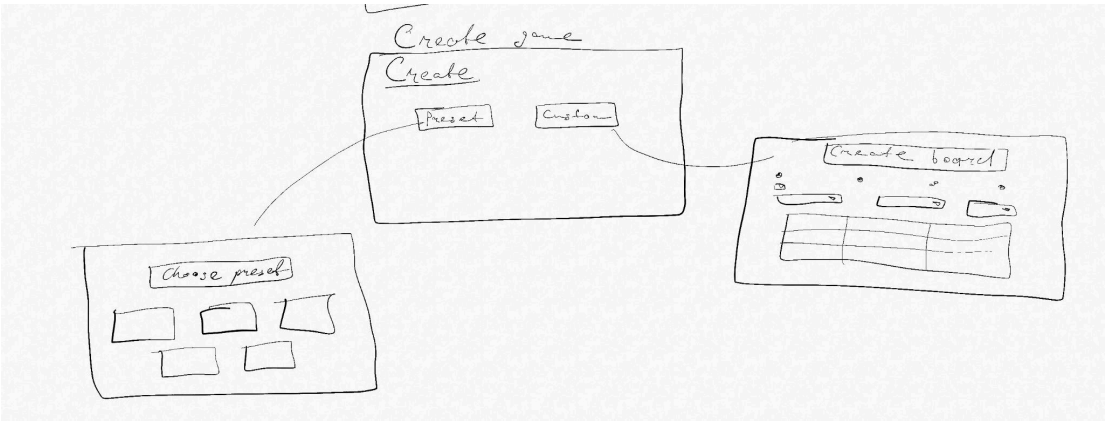
> RB - Choose a gamemode (Movies/Directors/Actors/Composers)
> DL - Choose genres (if [MOVIES] is chosen)
> SL - Years (release date(s) if [MOVIES], life if [ACTORS/DIRECTORS/COMPOSERS]
> DL - Choose movies (if [ACTORS/DIRECTORS/COMPOSERS])
> DL - Choose actors (if [MOVIES])
> DL - Choose directors (if [MOVIES])
> DL - Choose composers (if [MOVIES])
> SB - Show/Update table (show if no table is shown, update if there already is one in the view)
> TVS - Select up to 25 items from the displayed table. Table items can be sorted A-Z and Z-A by clicking on the header of a specific column

> SB - Random selection (randomly select (25 minus NUM_OF_SELECTED_ROWS) rows from the table, so that there's enough selected rows for the game to start)
> SB1 - Start local game
> SB2 - Start LAN game

Preset:

Possible player actions:

> many buttons with images: choose one preset game from many
> SB1 - Start local game
> SB2 - Start LAN game

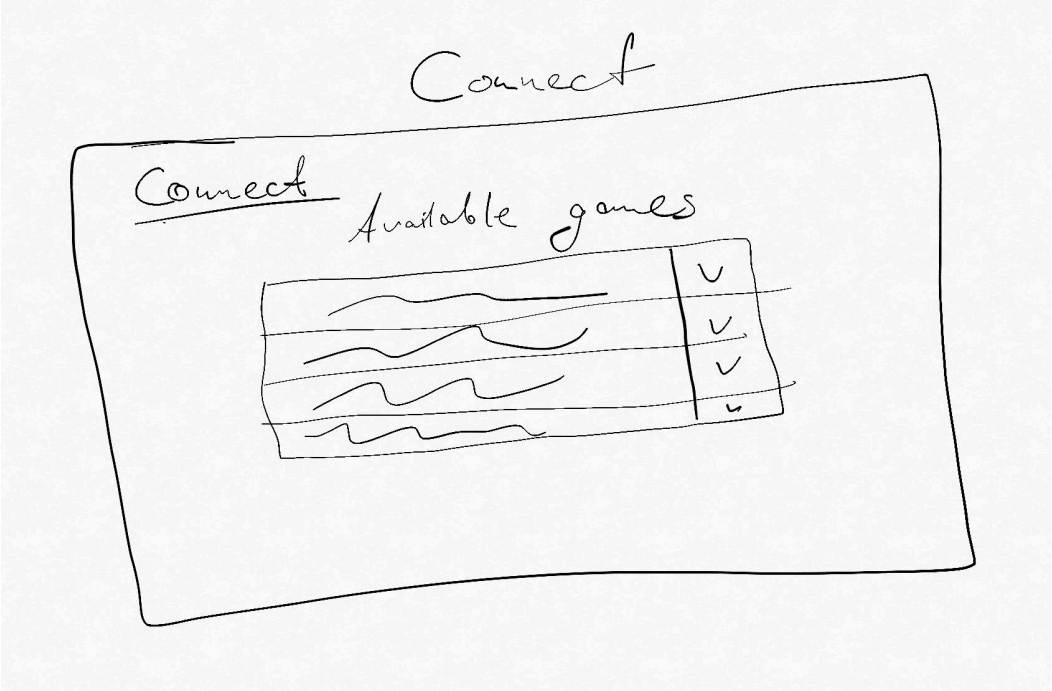


Connect to a LAN game:

This view checks for a network connection upon initialisation. If no network connection can be found, it shows a "No Connection" warning and prompts the player to try again.

Possible player actions:

> SB - reload (updates the view | if there is currently no connection, looks for a connection and updates the view if a network connection is found)
> TVC??? - a table with all of the current games on the network. A single click shows some info about the selected game, a double click initiates a connection to the game. In order to not have conflicts related to two people pressing the join button at the same time, the creator of the game has to approve a user before the game starts. Then that confirmation is sent to the clients of those who wanted to join, and either tells them to try again or connects them to the game.



Game [all modes]:

Possible player actions:

> TVS - a table with 25 cards. Hovering over a card offers the player more information about the item depicted on the card. The player may also select up to n-1 cards.

> SB - lower selected cards. Pressing this button discards the selected cards from the guessing pool. Once a card is lowered, it cannot be put back up. If no card is selected, a button press prompts you to either select cards or proceed without doing so

> SB - skip

> SB - guess. This button prompts the player to select one card (if they haven't selected any or have selected multiple) and press it again. If a singular card has already been selected prior to the press, it is considered the final guess of the game. Past that guess, the game checks whether the guess was correct or not. In case of a correct guess by player 1 (later - P1), they get pts = 5000*(percentage of nondiscarded cards out of the original 25) points added to their leaderboard. 25% of pts is deducted from player 2's (P2) score. In case of an incorrect guess by P1, they get 80% of pts deducted from their score and P2 gets 25% of pts added to their score.

> SB - rules. The rules are first shown in the beginning of a round, but a player can look at them at any point throughout the game by clicking this button

End of the game:

The player(s) will be prompted to add their opponent(s) score(s) to the leaderboard. If the opponent's name is not yet present on the leaderboard, they get added. If such a name is present on the leaderboard, the player has to choose between:

(i) replacing the previous scores scored by a player with that name;
(ii) adding this new player to the leaderboard [a (1) will be added after the name];
(iii) renaming and adding this player to the leaderboard;

Game [differences]:

Game [singleplayer]:

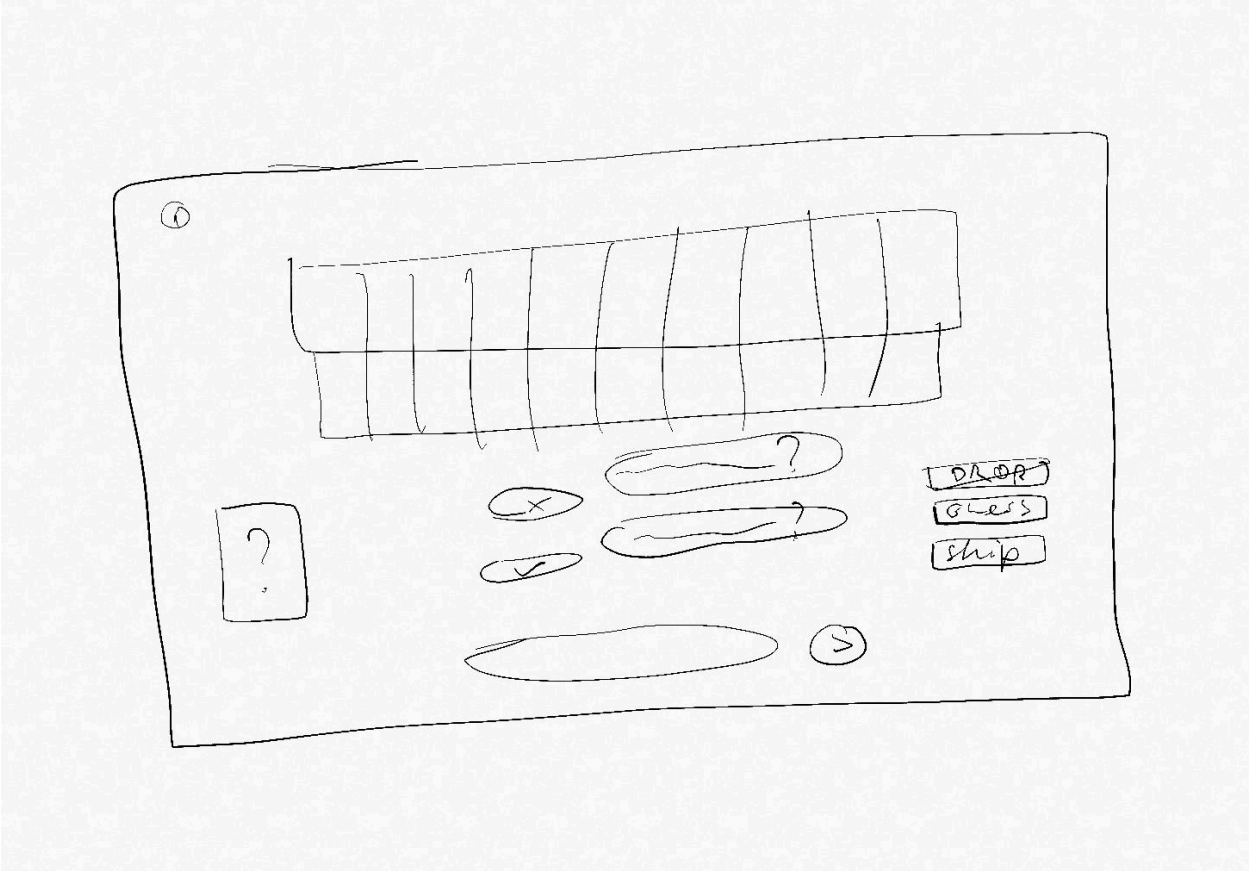
At the start of the game, the second player has to input their name.
Upon a player finishing their turn, the game switches to a “Handover” screen, which prompts P1/P2 to give the machine running the game to their opponent.
At the end of the game, both players’ scores, as well as who won, are displayed on the endgame screen

Game [multiplayer]:

Unlike singleplayer, which assumes that questions will be asked and answered by players in real life, multiplayer is based on the opposite assumption. Due to that, there’s a built-in chat function specifically for asking and answering questions. During each round, a player has to ask one question - that is, they are only allowed to send one message. Their opponent has to answer the question with “Yes” or “No” - to do that, they will get two buttons with such labels. These buttons only appear after the first player asks a question and disappear once the second player successfully answers it.
At the end of the game, each player is shown a screen with the words “YOU LOSE” or “YOU WIN” and their score vs their high score, as well as the opponent’s score (less prominent than the player’s)

Game [multiplayer/initiator]:

Upon creating a game, the player will first be met with an intermediate screen - one which will tell them who, if anyone, wishes to join their game. From there, they may accept any of these players. Following that action, the game begins



Leaderboard:

Possible player actions:

> TVC - a table with all of your past competitors, where you could press either of two buttons in each row:

> *SB - Delete the selected player from the leaderboard*

> *SB - Rename the selected player*

> SB - Save the leaderboard as a file

> SB - Upload a leaderboard file to add data to the current leaderboard (there have to be no name conflicts - people with the same name being present in both the uploaded and current leaderboards, yet having different scores and/or high scores)

> SB - Upload a leaderboard file to override the current leaderboard

Settings:

Possible player actions:

> IF, SB - Username change input field + confirmation button

> SB (many) - Switch colour scheme

> SL - Music volume

> SL - Sound volume

Credits:

Possible player actions:

There is nothing the player can do.

Technical solution:

The datasets required for this project are taken from https://datasets.imdbws.com. All of them are provided in the .tsv format. We will be using title.basics, title.crew, title.principals, title.ratings, name.basics.

As of now, the updates to our database will be done prior to the release of a new version of the application, rather than on a user's device. This is done in order to ensure that our datasets aren't faulty in any way and that players on the same version have the same datasets and that no errors will come up.

Because of this, we will develop a separate helper application that will be responsible for updating our database. It may or may not be merged into the main app later in the future, depending on the approach we take.

Helper:

```
> Takes the five files featured above as input, unzips them (the input format is tsv.gz)
> Creates tables for each of the files using SQL
> Purges any "tvEpisode" record from title.basics. This is done to ensure that we won't wind up with tens of records for each TV show, as well as a worrying amount of YouTube videos (Some may still be hidden behind other tags, such as "movie", but we shall deal with that later, if necessary)
> Merges title.basics and title.ratings, then removes anything with a rating below 7.0 (Some well-known movies might be lost after this action, however most of the ones we will be left with will still be known by the majority of players)
> Further modifies and creates the tables to fit our database structure (WIP)
```

In the main application, we'll also have to rely on a separate, smaller dataset - consisting of just one table with the following columns:

playername (unique, as of now - may be deprecated/replaced if a server multiplayer feature is added)

score

highscore

The leaderboard relies on this table in particular. It is edited each time a game ends or the player uploads a save file for the leaderboard

As we do not yet rely on a server, there is no UUID system.

Qt implementations of elements mentioned in the UX Solution section:

```
SB - QPushButton
IF - QLineEdit
RB - QRadioButton
CB - QCheckBox
DL - most likely QComboBox, perhaps with a personal implementation of the search functionality
SL - QSlider
TVS, TVC - QTable
```

Server-client connection functionality will be implemented using QTcpServer and QTcpSocket.
When the initial player starts a multiplayer game, their machine starts running the server to which any other computer on the network may connect - those other computers will act as clients sending requests to the server.
If the player running the server accepts one of the clients' requests to join their game, that client receives a packet confirming the beginning of the game, as well as another packet with all the necessary data to populate the game board with.
Throughout the game, the client and the server continue sending packets related to the players' actions to each other, until the end of the game, when the final packet is sent to the client in order to inform them of their win or loss. It is after that moment that the client gets disconnected from the server. Once the game is over and the "client" player has disconnected, the server is shut down.

Technical addendum: Multiplayer functionality will be added after everything else is done - to ensure that the main requirements for the project are met no matter if we manage to implement multiplayer before the end of May or not!!!

Platforms
Windows, Mac
(Mobile platforms may be included in the future)

Further considerations
In the future, a separate server could be implemented. This would make it so that none of the players have to run local servers in order to play with other people wirelessly.

Privacy concerns
Most of the privacy concerns stem from the multiplayer implementation - there needs to be some sort of protection against sending malicious packets from one machine to another. Of course, since the game doesn't do that much with the packets it sends and receives, there's not much to worry about, but we should still pay attention to this problem.

Development timeline
Week 1:
Helper application development, all the basic work required to create the databases, a basic shell of the GUI - windows with buttons and other widgets.
Week 2:
Anything left over from week 1 + every environment other than the game and game over screens. Pay the most attention to the game creation menu. The "presets" window may house placeholders at this point in time, which will be populated later
Week 3:
Anything left over from week 2 + the game environment [truthfully, it would've been better to begin working on this during week 2, however it may be too much]

Week 4:

Polishing the general gameplay loop in the local gamemode + deciding whether or not to go through with the LAN gamemode. If the decision is affirmative, start working on that

Week 5:

Finish the LAN implementation to the best of your ability. Final bug testing.