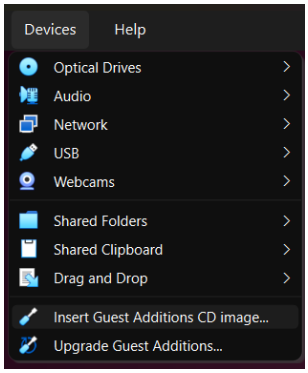
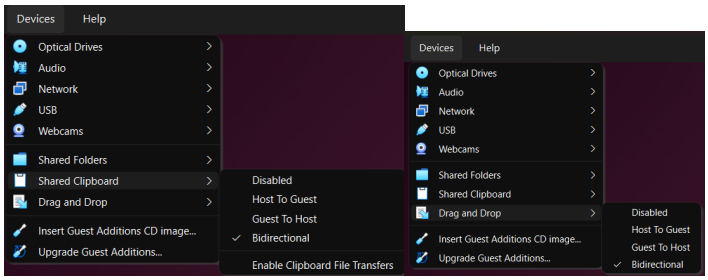
Writeup for cross Development project

My project will be a movie rater/tracker, to see what movies you have already watched and how you rated them. It will contain the functionality to save and load lists from text files (correctly structured ones) as well as some little costumization feature changing the color of the stars

I first started of with installing virtualbox and installing a ubuntu VM on it. Once fully installed I made copy pasting into the VM possible using the following steps

1. add guest addition CD image, on virtualbox go to devices -> insert guest additions CD image
    - creates a virtual CD that contains quest additions installer files
    - 

2. ```
   sudo apt-get install build-essential
   ```
    - build-essential is needed to be able to run step nr. 4
3. ```
   cd /media/jordi/VBox_GAs_7.2.2/
   ```
    - moving to the directory where the quest additions were mounted
4. ```
   sudo ./VBoxLinuxAdditions.run
   ```
    - executes the quest addition script, it contains drivers for better VM integration such as copy paste
5. ```
   sudo reboot
   ```
    - reboot is necessary to load the newly installed drivers
6. enable copy paste and drag & drop bidirectional, on virtualbox click devices -> hover over shared clipboard and then select bidirectional, do the same for drag and drop

- 

installing the MXE cross compiling toolchain on an ubuntu VM

for installing the MXE cross compiler a tutorial was provided: https://mxe.cc/#tutorial

MXE (M Cross Environment) is a toolchain building utility designed to create toolchains for cross compiling from Ubuntu to Windows

1. `sudo apt-get install \ autoconf \ automake \ autopoint \ bash \ bison \ bzip2 \ flex \ g++ \ g++-multilib \ gettext \ git \ gperf \ intltool \ libc6-dev-i386 \ libclang-dev \ libgdk-pixbuf2.0-dev \ libltdl-dev \ libgl-dev \ libpcre2-dev \ libssl-dev \ libtool-bin \ libxml-parser-perl \ lzip \ make \ openssl \ p7zip-full \ patch \ perl \ python3 \ python3-mako \ python3-packaging \ python3-pkg-resources \ python3-setuptools \ python-is-python3 \ ruby \ sed \ sqlite3 \ unzip \ wget \ xz-utils`
    - installing all necessary dependencies to install MXE

2. ```
   git clone https://github.com/mxe/mxe.git
   ```
    - downloads the MXE source code from GitHub and creates a local mxe/ directory with all build scripts and configuration files needed for cross-compilation

3. ```
   cd mxe
   ```
    - entering the just downloaded folder

4. ```
   make -j8 qt6
   ```
    - using the make build system, it builds Qt6 and all its dependencies for Windows cross-compilation using 8 cores. This creates a complete toolchain that can compile Qt applications for Windows from Linux. Here our Ubuntu machine is the Build and Host machine and our windows is the target machine

5. ```
   export PATH=/home/jordi/mxe/usr/bin:$PATH
   ```

    - this is required because the bin folder contains commands that we will execute from the commandline

Now we can build using the MXE toolchain we just created, first we need to create our makefile:

```
/media/sf_Cross_Dev/Project/MovieTracker# /home/jordi/mxe/usr/i686-w64-
mingw32.static/qt6/bin/qmake
```

- this will run the qmake tool from MXE, this command needs to be executed from the folder where the project is stored, it will use Qt's .pro file to create a makefile, this .pro file contains the project's configuration, including it's source files, header files and libraries

when opening the makefile.release we can get some information from it:

- name of the compiler that will be used

  ```
  CC = i686-w64-mingw32.static-gcc
  ```

- CFLAGS, -pipe: uses pipes instead of temp files, -o2: code optimization, -wall: enables common compiler warnings
- sources, contains all the source cpp files that need to be compiled

  ```
  main.cpp \
  movie.cpp \
  movietracker.cpp \
  moviewidget.cpp \
  star.cpp
  /media/sf_Cross_Dev/Project/MovieTracker/movietracker_plugin_import.cpp \
  release/moc_movietracker.cpp \
  release/moc_moviewidget.cpp \
  release/moc_star.cpp
  ```

- object, what object files will be generated during compilation

  ```
  release/main.o \
  release/movie.o \
  release/movietracker.o \
  release/moviewidget.o \
  release/star.o \
  release/movietracker_plugin_import.o \
  release/moc_movietracker.o \
  release/moc_moviewidget.o \
  release/moc_star.o
  ```

- target, defines the name of the executable

```
TARGET = MovieTracker.exe
```

and then at last we just have one more command to execute before we get our executable that we can distribute

```
make
```

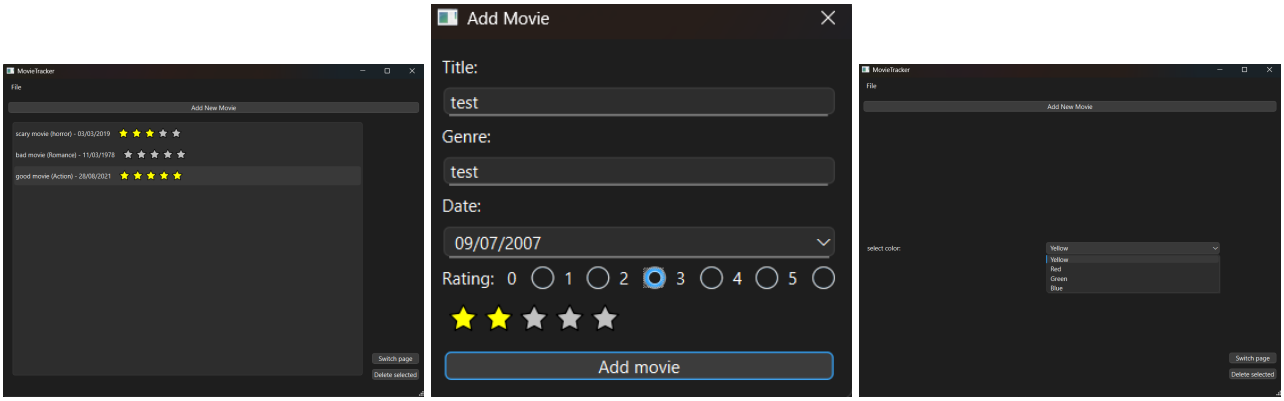
the make command executes the instruction from the makefile, it converts the cpp files from sources to object files using the specified compiler and cflags, it then combines these object files into one executable

changes to the concept of the app as a result of the used technology

no major changes were made to the concept of the app, only the possibility to change the color of the stars was added afterwards, this was not originally planned, but this was done because a window with variable content was required, aside from this only certain parts of the design (css) were simplified in some parts

The end result

my app turned out as a complete movie rating app, here some screenshots of the final result:



proof of cross compilation:

```
/include/QtWidgets -I/home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/include/QtGui -I/home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/include/QtCore -Irelease -I. -I/home/jordi/mxe/usr
/i686-w64-mingw32.static/qt6/mkspecs/win32-g++  -o release/moc_moviewidget.o release/moc_moviewidget.cpp
/home/jordi/mxe/usr/x86_64-pc-linux-gnu/qt6/libexec/moc -DUNICODE -D_UNICODE -DWIN32 -DMINGW_HAS_SECURE_API=1 -DQT_NO_DEBUG -DQT_WIDGETS_LIB -DQT_GUI_LIB -DQT_CORE_LIB -DQT_NEEDS_QMAI
N --include /media/sf_Cross_Dev/Project/MovieTracker/release/moc_predefs.h -I/home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/mkspecs/win32-g++ -I/media/sf_Cross_Dev/Project/MovieTrack
er -I/home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/include -I/home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/include/QtWidgets -I/home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/incl
ude/QtGui -I/home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/include/QtCore -I/home/jordi/mxe/usr/lib/gcc/i686-w64-mingw32.static/11.5.0/include/c++ -I/home/jordi/mxe/usr/lib/gcc/i686-
w64-mingw32.static/11.5.0/include/c++/i686-w64-mingw32.static -I/home/jordi/mxe/usr/lib/gcc/i686-w64-mingw32.static/11.5.0/include/c++/backward -I/home/jordi/mxe/usr/lib/gcc/i686-w64-
mingw32.static/11.5.0/include -I/home/jordi/mxe/usr/lib/gcc/i686-w64-mingw32.static/11.5.0/include-fixed -I/home/jordi/mxe/usr/i686-w64-mingw32.static/include star.h -o release/moc_st
ar.cpp
i686-w64-mingw32.static-g++ -c -pipe -fno-keep-inline-dllexport -O2 -std=gnu++1z -Wall -Wextra -Wextra -fexceptions -mthreads -DUNICODE -D_UNICODE -DWIN32 -DMINGW_HAS_SECURE_API=1 -DQ
T_NO_DEBUG -DQT_WIDGETS_LIB -DQT_GUI_LIB -DQT_CORE_LIB -DQT_NEEDS_QMAIN -I. -I/home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/include -I/home/jordi/mxe/usr/i686-w64-mingw32.static/qt6
/include/QtWidgets -I/home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/include/QtGui -I/home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/include/QtCore -Irelease -I. -I/home/jordi/mxe/usr
/i686-w64-mingw32.static/qt6/mkspecs/win32-g++  -o release/moc_star.o release/moc_star.cpp
i686-w64-mingw32.static-g++ -Wl,-s -Wl,-subsystem,windows -mthreads -o release/MovieTracker.exe release/main.o release/movie.o release/movietracker.o release/moviewidget.o release/sta
r.o release/movietracker_plugin_import.o release/moc_movietracker.o release/moc_moviewidget.o release/moc_star.o /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/plugins/styles/libqmod
ernwindowsstyle.a /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/lib/objects-Release/QWindowsIntegrationPlugin_resources_1/.qt/rcc/qrc_openglblacklists_init.cpp.obj /home/jordi/mxe/u
sr/i686-w64-mingw32.static/qt6/lib/objects-Release/QWindowsIntegrationPlugin_resources_2/.qt/rcc/qrc_cursors_init.cpp.obj /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/plugins/platf
orms/libqwindows.a -limm32 -loleaut32 -lsetupapi -lwinspool -lwtsapi32 -lshcore -lcomdlg32 -ld3d9 -lruntimeobject /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/lib/libQt6OpenGL.a /h
ome/jordi/mxe/usr/i686-w64-mingw32.static/qt6/plugins/iconengines/libqsvgicon.a /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/plugins/imageformats/libqgif.a /home/jordi/mxe/usr/i686
-w64-mingw32.static/qt6/plugins/imageformats/libqicns.a /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/plugins/imageformats/libqico.a /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/
plugins/imageformats/libqjp2.a -ljasper /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/plugins/imageformats/libqjpeg.a /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/plugins/imagefo
rmats/libqmng.a -lmng -llcms2 /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/plugins/imageformats/libqsvg.a /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/lib/libQt6Svg.a /home/jord
i/mxe/usr/i686-w64-mingw32.static/qt6/plugins/imageformats/libqtga.a /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/plugins/imageformats/libqtiff.a -ltiff -llzma -ljpeg /home/jordi/m
xe/usr/i686-w64-mingw32.static/qt6/plugins/imageformats/libqwbmp.a /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/plugins/imageformats/libqwebp.a -lwebp -lwebpdemux -lwebpmux -lsharp
yuv /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/lib/objects-Release/Widgets_resources_1/.qt/rcc/qrc_qstyle_init.cpp.obj /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/lib/objects
-Release/Widgets_resources_2/.qt/rcc/qrc_qstyle1_init.cpp.obj /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/lib/objects-Release/Widgets_resources_3/.qt/rcc/qrc_qstyle_fusion_init.cp
p.obj /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/lib/objects-Release/Widgets_resources_4/.qt/rcc/qrc_qmessagebox_init.cpp.obj /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/lib/
libQt6Widgets.a -ldwmapi /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/lib/objects-Release/Gui_resources_1/.qt/rcc/qrc_qpdf_init.cpp.obj /home/jordi/mxe/usr/i686-w64-mingw32.static/
qt6/lib/objects-Release/Gui_resources_2/.qt/rcc/qrc_gui_shaders_init.cpp.obj /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/lib/libQt6Gui.a -ld3d11 -ldxgi -ldxguid -ld3d12 -lgdi32 -l
uxtheme -lpng -lharfbuzz -lfreetype -lbz2 -lpng16 -lharfbuzz_too -lfreetype_too -lglib-2.0 -lintl -liconv -latomic -lm -lpcre2-8 -lshlwapi -lbrotlidec -lbrotlicommon -ld2d1 -ldwrite /
home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/lib/libQt6Core.a -lz -lsynchronization -lmpr -luserenv -ladvapi32 -lauthz -lkernel32 -lnetapi32 -lole32 -lshell32 -luser32 -luuid -lvers
ion -lwinmm -lws2_32 -lpcre2-16 -lzstd -lmingw32 /home/jordi/mxe/usr/i686-w64-mingw32.static/qt6/lib/libQt6EntryPoint.a -lshell32
make[1]: warning:  Clock skew detected.  Your build may be incomplete.
make[1]: Leaving directory '/media/sf_Cross_Dev/Project/MovieTracker'
make: warning:  Clock skew detected.  Your build may be incomplete.
```

Conclusion

This project demonstrates the successful setup of a cross-compilation environment using MXE on Ubuntu to build a Windows-compatible Qt application. The process included configuring the toolchain, setting up an Ubuntu VM, and creating an application that meets the specified requirements. Despite minor design simplifications, the project achieved its goal of creating a functional and customizable movie tracker, showcasing the effectiveness of cross-compilation tools in streamlining development for multiple platforms