Report for Video Player project

# Introduction

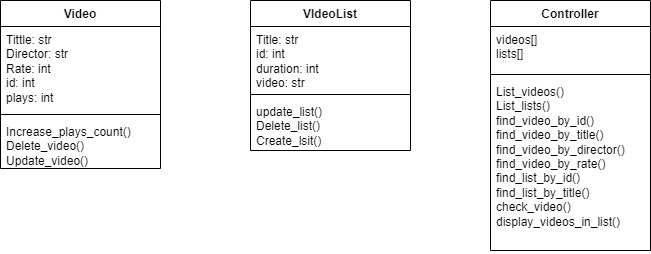
The simulator video player in a Python program that allows user to create and manage a virtual library of videos, and to play them using a graphical user interface (GUI). The project aims to demonstrate the use of object-oriented programming, GUI design, data structures, file handling, and testing techniques in Python. The project also showcases some innovations that enhance the functionality and user experience of the system, such as data is updated after user changes, user can use different mode of searching based on their needs, or having as much list as they want. And many more in this project.

The report is organized as follows: Section 1 provides an overview of the project requirements and the design process. Section 2 describes the implementation of the system, including the code structure, the classes and methods, the GUI components, and the innovations. Section 3 presents the testing and validation of the system, including the unit testing class using PyTest, and the functional testing of the GUI using a table of test cases. Section 4 discusses the evaluation and reflection of the project, including the strengths and weaknesses of the system, the challenges and difficulties encountered, and the future improvements and extensions. Section 5 concludes the report with a summary of the main findings and achievements.

# Design and Development

## Code Structure

The project consists of three main parts: the GUI, the controller, and the models, the GUI is responsible for creating and displaying the graphical user interface of the system, using Tkinter library, the controller is responsible for handling the user input and output, and communicating with the models. The models are responsible for storing and manipulating the data of the system, such as the Videos and VideoList. The code structure is shown in the flowing diagram.

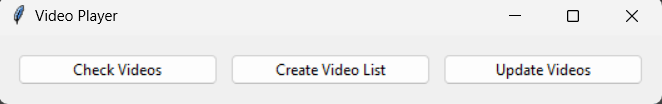


## Classes and Method

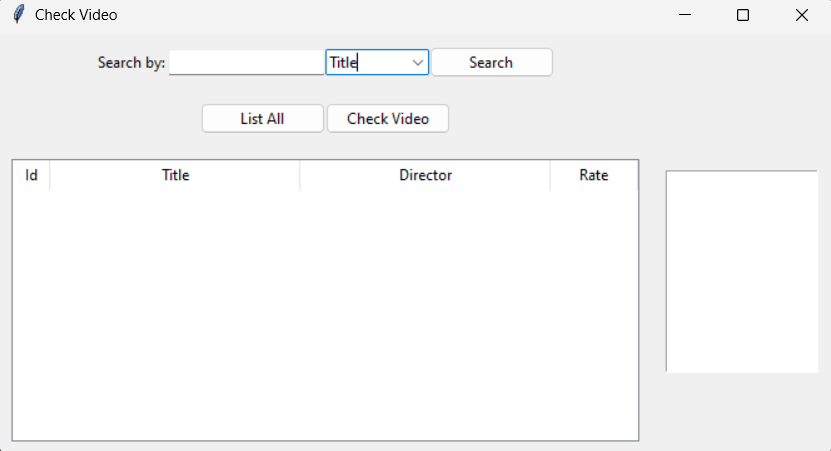
The project has three main classes: Video, VideoList, and Controller. The Video class represents a single video with attributes like title, director, rate, id, and plays. It has methods to increase the plays, delete the video, and update the video information. The VideoList class represents a list of videos with attributes like title, id, videos, and duration. It has methods to create, update, and delete a list. The Controller class contains all the methods that the user can use to interact with the system, such as listing videos or lists, finding videos or lists by different criteria, checking video details, and displaying videos in a list.

## GUI

The main window of the system is shown in the following screenshot.

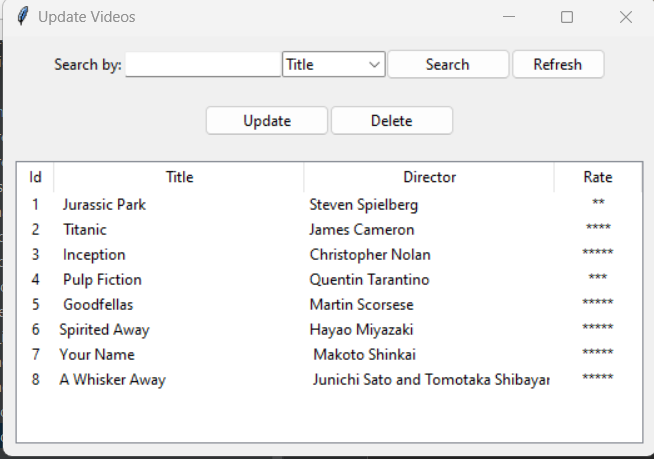


There are three different functions, the Check videos helps user can get more information about the videos they interested in, the Create video list allows user can create playlist, add their favorite videos, and play it. Last is the Update videos, which is a great function helps user can change the title, director, rate of the videos, or even reset the plays count.

  
This is the UI of check video functionality, user can search video with the search bar in four mode title, director, rate, or id. They can list all videos by a list all button and can check more information of that video by clicking on it then click check video button, more data of that video is displayed in the list box, which is the small display next to the main one.

**A screenshot of a computer

Description automatically generated**

The UI of the Create video list function has a search bar that lets the user search by id or by title. The user can create a new list by clicking on the New list button, and then add videos and name the list. The user can also select an existing list and use the List option function to change its data and save it. The user can play a list by clicking on the Play button, and a message will confirm the successful playback. The user can delete a list by clicking on the Delete button.

The Update videos window has a search bar with four modes to choose from. The user can update a video by selecting it and clicking on the Update button. A setting window will appear that allows the user to change the title, director, rate, or plays of the video. The user has to click on the Refresh button to see the updated display. The user can also delete a video by clicking on the Delete button.

## The innovations

I have implemented several innovations in my system that improve its functionality and user experience. These are:

Updating the searching function to make it more efficient and accurate.

Giving the user more rights to customize their lists and videos, such as adding, deleting, editing, or sorting them.

Allowing the user to have as many lists as they want, without any limit or restriction.

Improving the GUI by using ttk in the tkinter library instead of tk, which makes the project look more modern and stylish.

Changing the way the data is stored, from a dictionary to a csv file. This allows the data to be saved and updated automatically after the user makes any changes. To do this, I have used three different libraries: os, pandas, and Threading. The os library helps me to get the path of the csv file that I placed in the project folder. The pandas library helps me to read and write the data from and to the file. The Threading library helps me to keep the data updated while the user is using the application.

I chose these innovations because they fit my vision of the project from the beginning, which is to give the users as much ability to use the application as possible. I think storing the data in a csv file is especially important, because it prevents the users from losing their lists and videos every time they close the application.

# Testing and Faults

In general, most of the important errors have been handling by returning warning to user instead of returning error, except an error cause by using the thread function to update the data overtime, when user close the video player, the project cannot stop immediately but will start return error until user stop the program. Due to understanding ability with this library, I still cannot fix this error

# Conclusion, further development, and reflection

## Conclusion

For conclusion, the application contains all basic functions of a video player, and also have some upgrade which help the app work smoothly and effectively.

## Further development

I believe that if I had more 3 months to work on this project, I can find a solution to fix the error that cause by Thread function, and I can upgrade more function for this application, such as add images for each movie, design the whole GUI again, or even make it actually play videos.

## Reflection

In this reflection, I will discuss what I achieved with this element of learning, which were the most difficult parts, and why they were difficult for me, and which were the most straightforward parts, and why I found them easy.

the element of learning that i worked on was creating a simulate video player that allows users to store and manage video's data and list's data using csv file. The application has various features such as creating, updating, deleting, and searching for records. it also has a graphical user interface(GUI) that makes it easy and convenient for user to interact with application.

One of the most difficult parts of this element of learning was implementing data persistence using csv files. This was the first time i had to deal with this type of file format, so I had to do a lot of research and learning. I used pandas, a Python library that provides tools for data analysis and manipulation, to open and read the csv file into a data frame. Then, I used various methods and functions from pandas to perform operations on the data frame, such as adding, modifying, or deleting. I also had to make sure that the csv file was formatted correctly so that it could work properly with the application. For example, I had to store the personal information of each user in a separate row, with each column representing a different attribute, such as with video are id, title, director, rate, plays or in list is id, title, video\_ids, duration.

Another difficulty that i faced was updating the data in the csv file while the application was running. I noticed that the changes that I made in the application were not updated to the application until i restart the application. Therefore, I decided to use threading, a Python module that allows multiple threads of execution within a single process. By using threading, I was able to create a separate thread that would update the changes for my application every time a record was added, updated, or deleted in the application. This way, i ensured that the data in the csv file was always synchronized with the data in the application. On the other hand, one of the easiest parts of this element of learning was creating the GUI for the application using Tkinter, a Python module that provides a standard toolkit for creating GUIs. Although I found it difficult at first, i soon got familiar with the concept and structure of Tkinter. I learned how to create different widgets such as buttons, labels, entries, etc., and how to arrange them using geometry mangers such pack, grid. I also learned how to bind events and callbacks to widgets so that they could response to user actions. Moreover, I made all the windows look similar in purpose so that I could reuse some of the code later on, which saved me a lot of time. In conclusion, this element of learning was a valuable learning experience for me because it helped me to develop my knowledge and skills in Python programming. I learned how to use various libraries and modules such as pandas, threading and Tkinter to create a functional and user-friendly application. I also learned how to overcome some of the challenges and problems that I encountered along the way by doing research, consulting sources, and seeking feedback. Some of the feedback that I received from my peers and instructors were about improving the design and usability of the GUI, adding more error handling and validation for user inputs, and documenting my code more clearly. These are some of the areas that I can improve my work in the future.

# Appendices

## Appendix A

this is an example of how I commented my code

A screenshot of a computer program

Description automatically generated

## Appendix B

a. Create list video

|  |  |  |  |
| --- | --- | --- | --- |
| Sample Input | Sample Actions | Expected Output | Actual Output |
| None | Click List All button | List out all list | List out all list |
| List 2(title mode) | Click Search button | display “List2” on screen | display “List2” on screen |
| 2(id mode) | Click Search button | display “List2” on screen | display “List2” on screen |
| List 1(clicked in) | Click Play button | video’s play in list1 +1 | video’s play in list1 +1 |
| List 1(clicked in) | Click List option\* | open list option, data of list 1 display on the window | open list option, data of list 1 display on the window |
| List 1(clicked in) | Click Delete List | Delete list 1 | Delete list 1 |
| List 1 | Double click list 1 | Display all video in list 1 | Display all video in list 1 |
| None | Click New List\* | Open New list window | Open new list window |

* New List

|  |  |  |  |
| --- | --- | --- | --- |
| Sample Input | Sample Actions | Expected Output | Actual Output |
| Video 1 | Click Add Video button | Video 1 transfer to Video Added screen | Video 1 transfer to Video Added screen. |
| Video 1 | Click Remove Video button | Video 1 transfer back to All Videos screen | Video 1 transfer back to All Videos screen |
| List’s name | Click Create button | Create a new list | Create a new list |

* List option

|  |  |  |  |
| --- | --- | --- | --- |
| Sample Input | Sample Actions | Expected Output | Actual Output |
| Video 1 | Click Add Video button | Video 1 transfer to Video Added screen | Video 1 transfer to Video Added screen. |
| Video 1 | Click Remove Video button | Video 1 transfer back to All Videos screen | Video 1 transfer back to All Videos screen |
| None | Click Save button | Save changes in the list | Save changes in the list |

|  |  |  |  |
| --- | --- | --- | --- |
| Sample Input | Sample Actions | Expected Output | Actual Output |
| Video 1(title mode) | Click Search button | Video 1 displayed only | Video 1 displayed only |
| Director 1 (director mode) | Click Search button | Video contains director 1 displayed only | Video contains director 1 displayed only |
| 4(rate mode) | Click Search button | Videos have 4 rate displayed only | Videos have 4 rates displayed only |
| 1(id mode) | Click Search button | The Video have id 1 displayed only | The Video have id 1 displayed only |
| Video 1(clicked in) | Click \*Update button | Display update window whit data of video 1 inside | Display update window whit data of video 1 inside |
| Video 1(clicked in) | Click Delete button | Delete video 1 | Delete video1 |

1. Update videos

* Update

|  |  |  |  |
| --- | --- | --- | --- |
| Sample Input | Sample Actions | Expected Output | Actual Output |
| 4 (plays count) | Click Reset button | Plays count become 0 | Plays count become 0 |
| Video 1 title to video 12 | Click Save button | Title of video 1 saved to video 12 | Title of video 1 saved to video 12 |

1. Check Videos

|  |  |  |  |
| --- | --- | --- | --- |
| Sample Input | Sample Actions | Expected Output | Actual Output |
| None | Click list all button | List all videos | List all videos |
| choose video 1 | Click check video button | Get title, director, rate and plays | Get title, director, rate and plays |
| Type video 1(title mode) | Click search button | Video 1 display | Video 1 display |
| Type director 1(director mode) | Click search button | Video of director 1 display | Video of director 1 display |
| Type 3(rate mode) | Click search button | Videos have 3 rate display | Videos have 3 rate display |
| Type 1(id mode) | Click search button | Video has id 1 display | Video has id 1 display |