**Project report – Video Player Simulation**

***Nguyen Hoang Lan***

*University of Greenwich*

*Email: lannhgcd220043@fpt.edu.vn*

# Introduction

In this project, the primary objective is to develop a Video Player Simulation using the Tkinter library in Python. The application will involve creating a Graphical User Interface (GUI) and managing a database to store video information. To accomplish this, Object-Oriented Programming (OOP) principles will be applied to design and structure the system effectively.

The application will consist of four main windows: the Video Player, Check Video, Create Video, and Update Video. Each window will serve specific functions and interact with the user, providing a user-friendly experience. The project will also focus on creating a dynamic and visually appealing interface, ensuring that buttons, input fields, and text areas are responsive and well-designed.

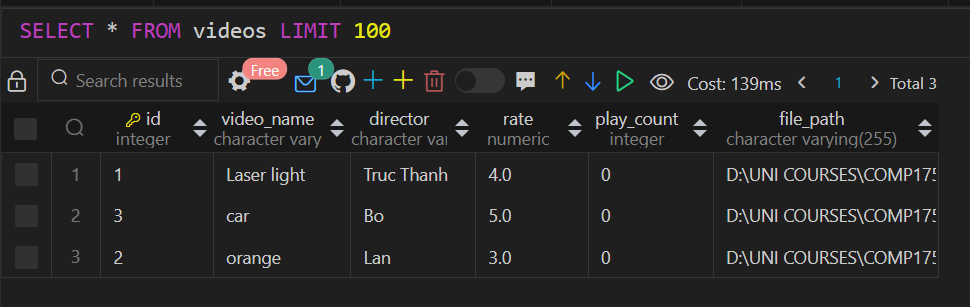
Through this project, I aim to demonstrate a solid understanding of OOP design and implementation, while also showcasing creativity in the application's user interface and functionality.

# Design and development

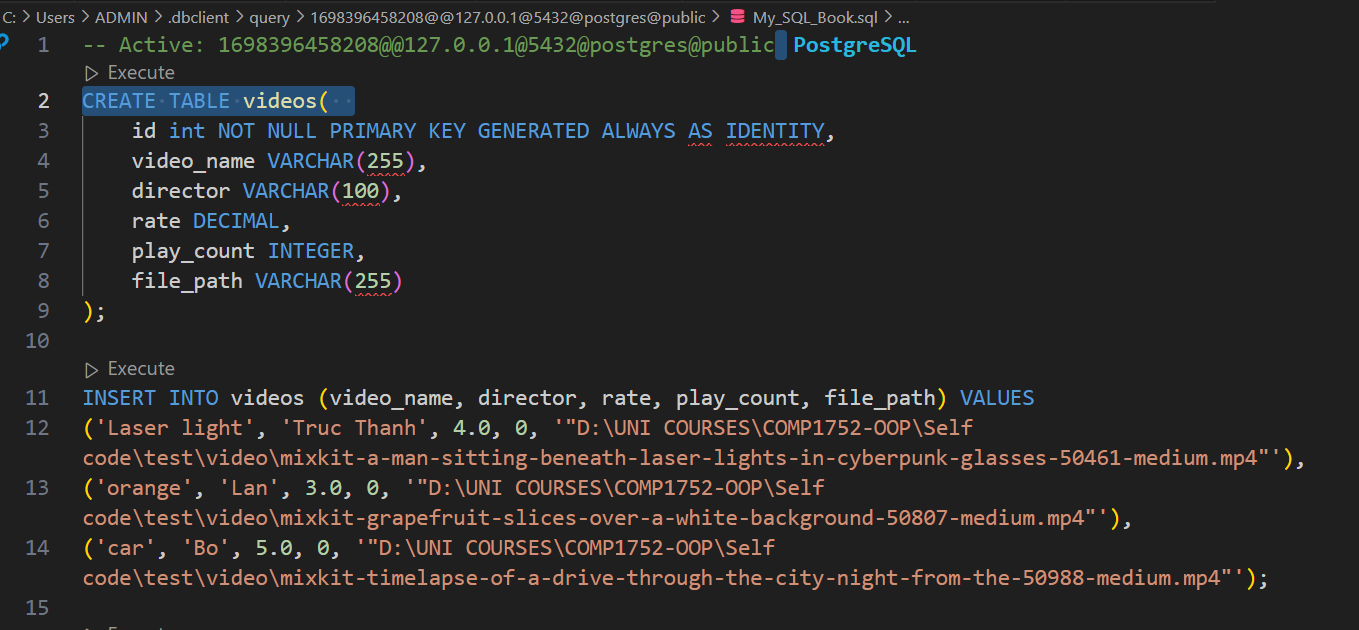
## 2.1. **Data Storage:**

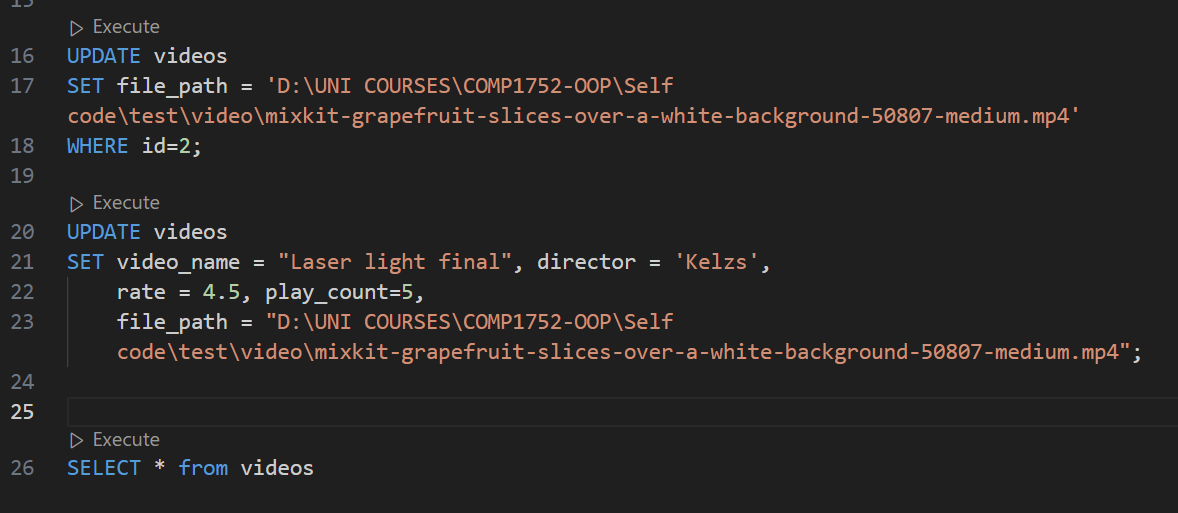
In this project, the data storage is handled using a PostgreSQL database. Here's a brief design for data storage:

* **Database Tables:** The application uses a single table named "videos" to store video information. This table includes columns such as "id," "video\_name," "director," "rate," "play\_count," and "file\_path."



* **PostgresDB Class:** The **PostgresDB** class is responsible for managing database connections and queries. It provides methods to connect to the database, close the connection, select all videos, select a specific video by ID, and update video information.
* **Queries:** SQL queries are used to interact with the database.

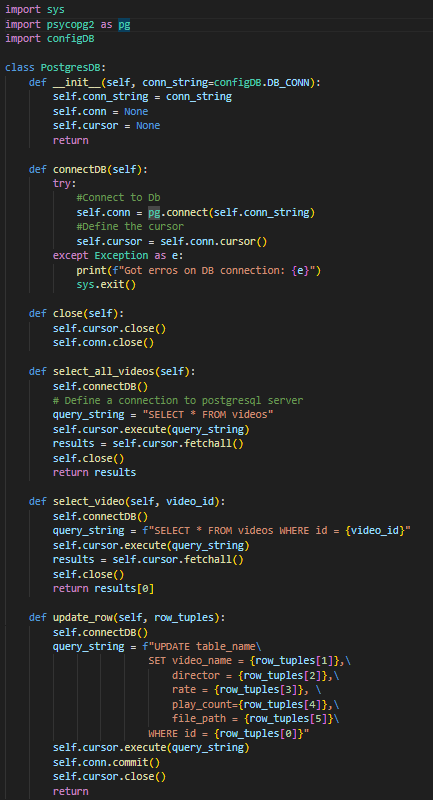




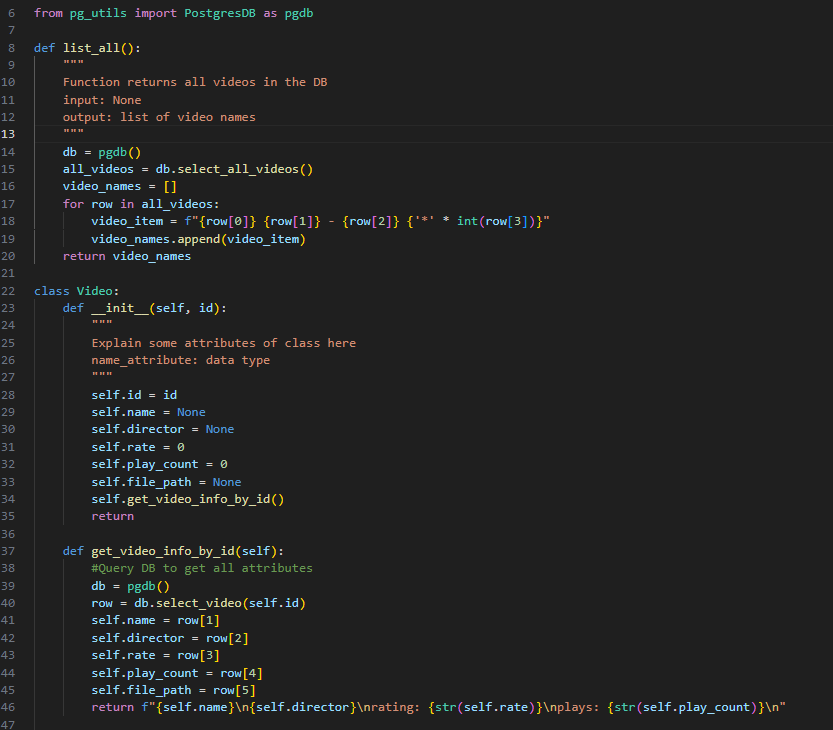
## 2.2. Object-Oriented Design:

The project uses object-oriented design principles to structure the code effectively. Here's a brief overview of the object-oriented design:

* **PostgresDB Class:** This class encapsulates the database connection and operations. It follows the encapsulation principle by keeping the database connection details within the class and providing methods to interact with the database. It also utilizes abstraction by abstracting the database operations into methods like **select\_all\_videos**, **select\_video**, and **update\_row**.



* **Video Class:** The **Video** class represents individual video objects. It follows the encapsulation principle by encapsulating video attributes such as ID, name, director, rate, play count, and file path. The class provides methods for retrieving and updating video information, such as **get\_video\_info\_by\_id**, **get\_name**, **get\_director**, **get\_rating**, **set\_rating**, **get\_play\_count**, and **increment\_play\_count**.
* **Class Interactions:** The **Video** class interacts with the **PostgresDB** class to retrieve and update video information in the database. It maintains a clear separation of concerns between the data storage and object-oriented representation of video data.



## 

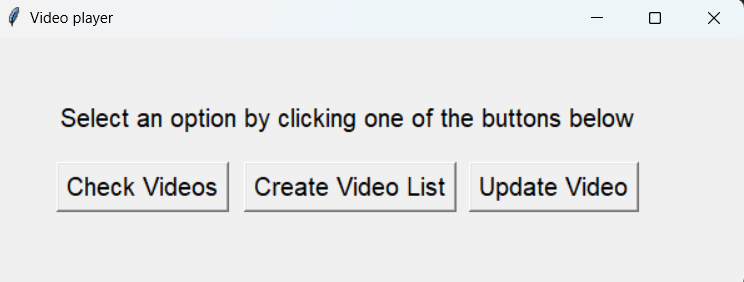
## 2.3) User Interface Design

The application is designed to provide a graphical user interface for managing a video library. It includes three main functionalities: checking videos, creating a video list, and updating video information. The user interface is designed using the Tkinter library.

* Main Window:

The main window displays the title "Video Player" and has a size of 600x200 pixels.

It provides three buttons for the user to select different functionalities: "Check Videos," "Create Video List," and "Update Video."



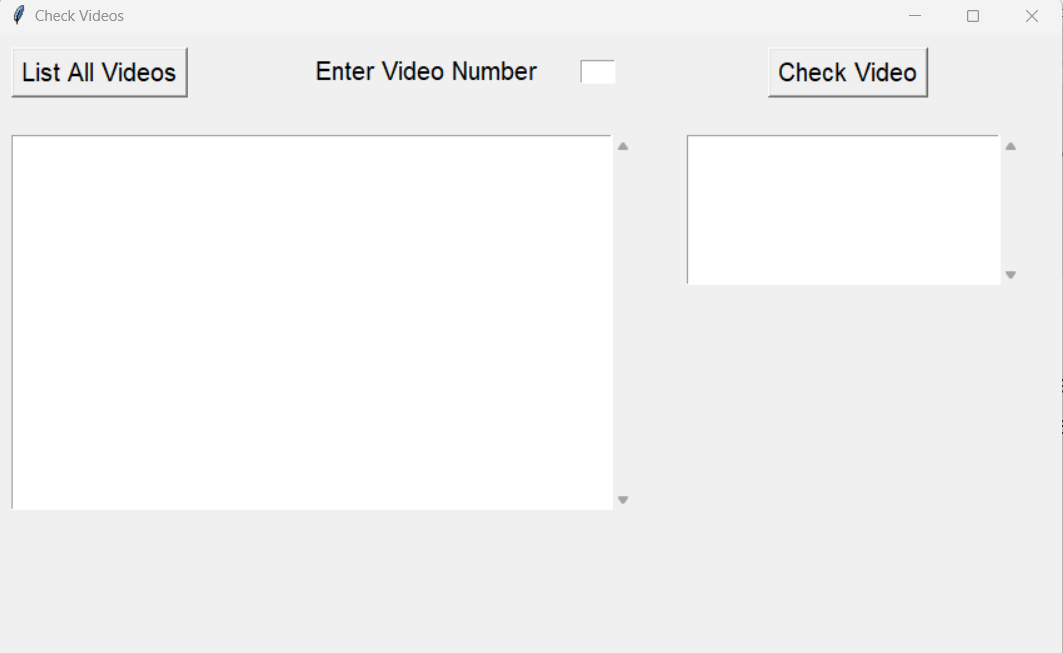
* Check Videos:

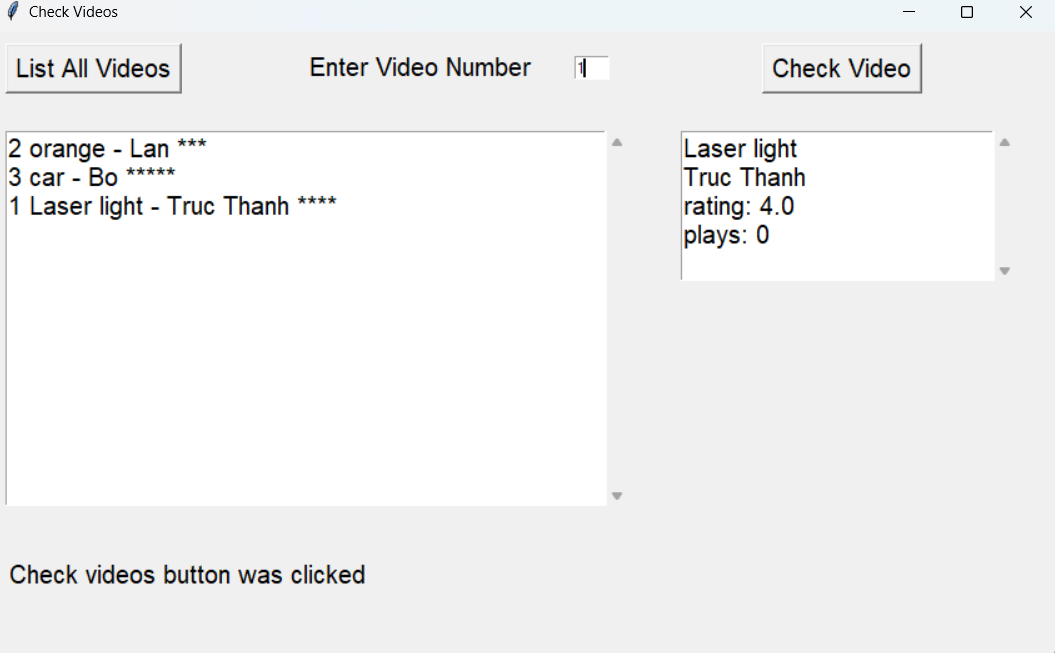
The "Check Videos" button opens a new window to list all available videos.

Users can enter a video number to view video details, and there is an option to check all videos available.

The window includes buttons to list all videos and check a specific video.

Scrollbars are provided for viewing long lists of videos.



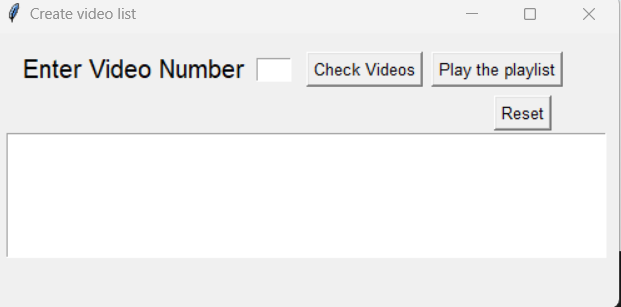


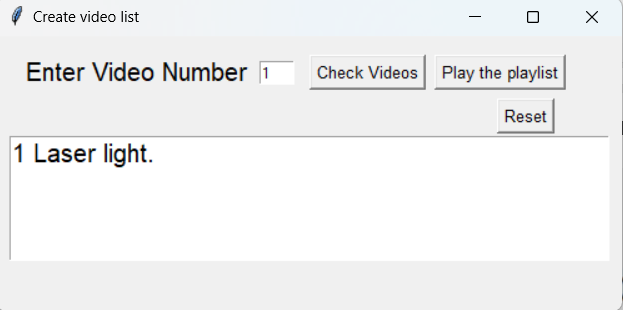
* Create Video List:

The "Create Video List" button opens a new window to create a playlist of videos.

Users can enter a video number to add it to the playlist. The list of selected videos is displayed.

Options to play the playlist and reset the list are provided.





* Update Video:

The "Update Video" button opens a window to update the rating of a video.

Users can enter a video number and a new rating, and the updated video details are displayed.

An option to update the rating is provided.

