# YOUTUBE API DATABASE PROJECT

**A PROJECT REPORT**

**SUBMITTED BY :-**

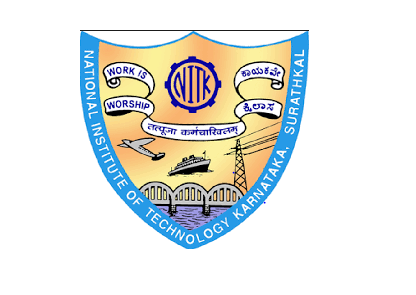
**ADARSH THAKUR (194CA002)**

**PARAS NARAYAN GAUTAM (194CA030)**

**TO:-**

# MS. USHA KIRAN

# **In partial fulfillment for the award of the degree of**

**MASTER OF COMPUTER APPLICATIONS**

**NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA,**

**SURATHKAL**

JUNE 2020

**Mathematical and Computational Science**

**National Institute Of Technology Karnataka, Surathkal**

# DECLARATION

# We hereby to declare that the DBMS project entitled “YOUTUBE API DATABASE PROJECT” which is being submitted to the National Institute of Technology Karnataka, Surathkal, is record of an original work done by us under the guidance of Ms. Usha Kiran in partial fulfillment of the requirements for the award of the degree of Master of Computer Applications in the department of Mathematical and Computational Sciences, is a bonafide report of the work prepared by us. This material is collected from various sources with utmost care and is based on facts and truth.

**ADARSH THAKUR (194CA002)**

**PARAS NARAYAN GAUTAM (194CA030)**

MCA II SEM

# NITK, SURATHKAL

# 

# 

# 

# 

# 

# 

# CERTIFICATE

# Certified that this project report “Youtube API Database Project” is the bonafide work of ADARSH THAKUR and PARAS NARAYAN GAUTAM who carried out the project under my supervision. This is to further certify to the best of my knowledge, that this project has not been carried out earlier in this institute and the university.

# Signature

# MS. USHA KIRAN

# Certified that the above mentioned project has been duly carried out as per the norms of the college and statutes of the university.

# Signature

# (PROF. B.R. SHANKAR)

# HEAD OF THE DEPARTMENT

# 

# 

# 

# 

# 

# ACKNOWLEDGEMENT

# I wish to express my profound and sincere gratitude to MS. USHA KIRAN Department of Mathematical and Computational Science NITK, Surathkal, who guided me into the intricacies of this project.

# I am highly grateful to our team who evinced keen interest and invaluable support in the progress and successful completion of my project work.

# I am indebted to our team for their constant encouragement, co-operation and help. Words of gratitude are not enough to describe the accommodation and fortitude which they have shown throughout my endeavor.

# 

# MS. USHA KIRAN

# 

# 

# 

# 

# 

# 

# 

# 

# CONTENTS

# 

TITLE PAGE NO.

# Abstract 6

1. Introduction 7
2. Literature Survey 10-11
3. Explanation 12-13
4. Scope of Project 14
5. Purpose of Project 14
6. System Requirements 15-16
7. Code 17-28
8. Screenshots 29-37
9. Future Aspects 38
10. Conclusion 38
11. References 39

# 

# 

# 

# ABSTRACT

Basically, users can search any kind of detail between a given time period on our search engine. Here in this project we are dealing with the youtube’s data by using the Official Youtube API Key. And Store the data in our database and fetch it from our database in well sorted order.

YouTube allows users to upload, view, rate, favorites, share, add to playlists, flag, report, comment on videos, and subscribe to other users. It offers a wide variety of user-generated and corporate media videos.

An API approach is an architectural approach that revolves around providing a program interface to a set of services to different applications serving different types of consumers.

Actually here, API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other.

Our project, which uses the youtube data api v3 . Its function is to search the channels based on the keyword entered and give the result in the form of a list. The details we got name ,uploaded date, subscriber count, video count, views count, channel id Etc. Eventually we are not creating data, we are just fetching already existing data from the particular youtube channel.

# 

# 

# INTRODUCTION

The main aim of a DBMS is to supply a way to store up and retrieve database information that is both convenient and efficient. By data, we mean known facts that can be recorded and that have embedded meaning.

We use SQLite which is a Relational Database Management System (RDBMS) contained in a C Library. In contrast to many other database management systems, SQLite is not a client server database engine. Rather, it is embedded into the end program.

This Project is aimed at to see the Youtube account data like its associated subscriber count, like count, comment count, view count, etc.

Frontend and backend both are implemented using Python. Along with the Tkinter python GUI (Graphical User Interface) toolkit in the frontend to make it more accessible. Python with tkinter is the fastest and easiest way to create the GUI applications.

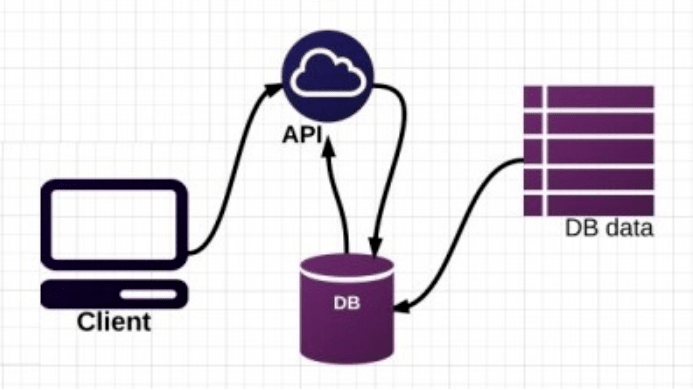


Image 1 :-Depicting how an API interacting with Database.

Here, Client is FRONTEND.

**ER- Diagram**

****

**LITERATURE SURVEY**

**Introduction to Database Management System**

DBMS stands for Database Management System. We can break it like this DBMS= Database +Management System. Database is a collection of data and Management System is a set of programs to store and retrieve those data. Based on this we can define DBMS like this: DBMS is a collection of interrelated data and a set of programs to store and access those data in an easy and effective manner.

Database systems are basically developed for large amounts of data. When dealing with huge amounts of data, there are two things that require optimization: Storage of data and retrieval of data. According to the principles of database systems, the data is stored in such a way that it acquires a lot less space as the redundant data(duplicate data) has been removed before storage.

Along with storing the data in an optimized and systematic manner, It is also

important that we retrieve the data quickly when needed. Database system ensures that data is retrieved as quickly as possible.

**Application of DBMS**

The development of computer graphics has been driven both by the needs of

the user community and by the advances in hardware and software. The

applications of database are many and varied; it can be divided into four major

areas:

1. Hierarchical and network system

2. Flexibility with relational database

3.Object oriented application.

**Display Information**

In this particular project, we are using Python to design our web page as a front end in order to display the information which is stored in the backend database called SQLite3.

**Design**

Unlike client–server database management systems, the SQLite engine has no standalone [processes](https://en.wikipedia.org/wiki/Process_(computing)) with which the application program communicates. Instead, the SQLite [library](https://en.wikipedia.org/wiki/Library_(computing)) is [linked](https://en.wikipedia.org/wiki/Linker_(computing)) in and thus becomes an integral part of the application program. Linking may be [static](https://en.wikipedia.org/wiki/Static_library) or [dynamic](https://en.wikipedia.org/wiki/Dynamic_linker). The application program uses SQLite's functionality through simple [function calls](https://en.wikipedia.org/wiki/Subroutine), which reduce [latency](https://en.wikipedia.org/wiki/Latency_(engineering)) in database access: function calls within a single process are more efficient than [inter-process communication](https://en.wikipedia.org/wiki/Inter-process_communication).

SQLite stores the entire database (definitions, tables, indices, and the data itself) as a single cross-platform [file](https://en.wikipedia.org/wiki/Computer_file) on a host machine. It implements this simple design by [locking](https://en.wikipedia.org/wiki/Lock_(computer_science)) the entire database file during writing. SQLite read operations can be multitasked, though writes can only be performed sequentially.

**User Interfaces**

Our interactions with computers has become dominated by a visual paradigm that

includes windows, icons, menus, pointing devices, such as a mouse.

Although we are familiar with the syntax of SQLite3, advances in MySQL have

made possible other forms of advantages.

**EXPLANATION**

## Python sqlite3 module APIs

Following are sqlite3 module routines, which can suffice the requirement to work with SQLite database from Python program.

|  |  |
| --- | --- |
| **Sr.No.** | **API & Description** |
| 1 | **sqlite3.connect(database [,timeout ,other optional arguments])**  This API opens a connection to the SQLite database file. We can use ":memory:" to open a database connection to a database that resides in RAM instead of on disk. If the database is opened successfully, it returns a connection object. |
| 2 | **connection.cursor([cursorClass])**  This routine creates a cursor which will be used throughout your database programming with Python. This method accepts a single optional parameter cursorClass. If supplied, this must be a custom cursor class that extends sqlite3.Cursor. |
| 3 | **cursor.execute(sql [, optional parameters])**  This routine executes an SQL statement. The SQL statement may be parameterized (i. e. placeholders instead of SQL literals) |
| 4 | **connection.commit()**  This method commits the current transaction. If you don't call this method, anything you did since the last call to commit() is not visible from other database connections. |
| 5 | **cursor.fetchall()**  This routine fetches all (remaining) rows of a query result, returning a list. An empty list is returned when no rows are available. |
| 6 | **connection.close()**  This method closes the database connection. |

## 

## CONNECT TO DATABASE

Following Python code shows how to connect to an existing database.

import sqlite3

conn = sqlite3.connect(“youtube.db”)

print "Opened database successfully";

## CREATE A TABLE

import sqlite3

conn = sqlite3.connect('test.db')

print "Opened database successfully";

cur.execute("CREATE TABLE IF NOT EXISTS new\_youtube (TITLE TEXT , YOU\_ID TEXT PRIMARY KEY, VIEW\_COUNT INTEGER,COMMENT\_COUNT INTEGER,SUBSCRIBER\_COUNT INTEGER,HIDDEN\_SUBSCRIBER\_COUNT INTEGER,VIDEO\_COUNT INTEGER)")

**SCOPE OF PROJECT**

The scope of the project is clear to give a simple and attractive experience to simplify the work as well as to reduce the efforts while doing it on another platform or we can say by doing it with old methods.

**PURPOSE OF PROJECT**

The purpose of this project is to outline Youtube’s Channel data and requirements, to recommend data management solutions and to provide information. Or is to develop a data management system to consolidate, organize, document, store and distribute information.

A centralized database created to consolidate data ,allowing integrated, long term

analyses, and dynamic search ability with user friendly query tools to be performed to support adaptive management. Many data collection, analysis and presentation software programs that are currently being used must be able to interface with any new data system. Continuity with consistent data collection methodology is enforced by a common database system, allowing for standardized format between projects.

**SYSTEM REQUIREMENTS**

For the application to run on computer, the device is expected to meet the following system requirements. The system requirements were categorized into hardware and software requirements as shown in Table 1 and Table 2.

Hardware:-

|  |  |  |
| --- | --- | --- |
| **Hardware** | **Minimum Requirement** | **Reason** |
| Processor speed | 1.8GHZ | Accommodate most PCs |
| Memory of user PC | 512MB RAM | Relatively fast |
| Disk Space of user PC | 3GB | Adequate Storage capacity |
| Memory of server Pc | 4GB | Fast |
| Bandwidth(network connection) | 1Mbps | Relatively Good |

Table 1: Minimum Hardware Requirements

A fast processor is required because there is need to handle large amounts of data queries.Good network connection is vital because the application is majorly internet based and there is need for faster communication and retrieval of information.

**Software:-**

|  |  |  |
| --- | --- | --- |
| **Software** | **Minimum requirement** | **Reason** |
| Operating System for PC | Windows 7,8.1,10,linux,Mac | Accessed all over the World |
| Database Management System | SQLite3 | Easy to use and scalable |
| Browser | Opera, Chrome, IE, Mozilla Firefox | Standard browser |

Table 2: Minimum Software Requirements

Here we use SQLite3 in the development of the databases and is relatively cheap, easy to use and scalable. Browsers: any browser including opera, Google chrome, Microsoft Internet explorer and Mozilla Firefox.

**CODE**

**# FRONTEND:-**

from tkinter import \*

from tkinter import ttk

from typing import List, Any

import apiclient.discovery

import tkinter.messagebox

import urllib.request

import youtube\_api\_backend

from PIL import ImageTk, Image

from datetime import datetime

import json

import sys

import os

import tkinter as tk

api\_key = "AIzaSyADsCqG71P\_Xoesodbz-I-75wVh96QJk2w"

youtube = apiclient.discovery.build('youtube', 'v3', developerKey=api\_key)

class hello:

def \_\_init\_\_(self, root):

def restart\_program():

os.system('python "D:\\sublime programs\\my1st.py"')

sys.exit()

def funky():

self.root = root

self.root.title("Search Screen (SCREEN 1)\n")

self.img = Image.open('blue2.png')

self.new\_image = self.img.resize((1532, 835))

self.new\_image.save('blue2.png')

self.new\_image = ImageTk.PhotoImage(Image.open('blue2.png'))

self.my\_label = Label(image=self.new\_image)

self.my\_label.place(relx=0, rely=0)

You = StringVar()

You\_tube\_url = StringVar()

name = StringVar()

start\_Date = StringVar()

end\_Date = StringVar()

TITLE = StringVar()

SUBSCRIBER\_COUNT = StringVar()

VIEW\_COUNT = StringVar()

VIDEO\_COUNT = StringVar()

COMMENT\_COUNT = StringVar()

HIDDEN\_SUBSCRIBER\_COUNT = StringVar()

YOU\_ID = StringVar()

PUBLISHED\_AT = StringVar()

self.l1 = ttk.Label(self.root, font=('Times New Roman', 9, 'bold'), text="search your choice :")

self.l1.place(relx=0.2093, rely=0.512)

self.l2 = ttk.Label(self.root, font=('Times New Roman', 9, 'bold'), text="End Date :")

self.l2.place(relx=0.5, rely=0.7)

self.l3 = ttk.Label(self.root, font=('Times New Roman', 9, 'bold'), text="Starting Date :")

self.l3.place(relx=0.3, rely=0.7)

self.txtYou\_tube\_url = Entry(font=('COMIC SANS MS', 20, 'bold'), textvariable=You\_tube\_url, width=39)

self.txtYou\_tube\_url.place(relx=0.3, rely=0.5)

self.txtstart\_Date = Entry(font=('COMIC SANS MS', 20, 'bold'), textvariable=start\_Date, width=10)

self.txtstart\_Date.place(relx=0.3, rely=0.75)

self.txtend\_Date = Entry(font=('COMIC SANS MS', 20, 'bold'), textvariable=end\_Date, width=10)

self.txtend\_Date.place(relx=0.5, rely=0.75)

def fun():

if len(start\_Date.get()) and len(end\_Date.get()) != 0:

start\_time\_date = datetime.strptime(str(start\_Date.get()), "%d/%m/%Y")

start\_time = datetime(year=start\_time\_date.year, month=start\_time\_date.month,day=start\_time\_date.day).strftime('%Y-%m-%dT%H:%M:%SZ')

end\_time\_date = datetime.strptime(str(end\_Date.get()), "%d/%m/%Y")

end\_time=datetime(year=end\_time\_date.year, month=end\_time\_date.month, day=end\_time\_date.day).strftime('%Y-%m-%dT%H:%M:%SZ')

req = youtube.search().list(part='snippet', q=You\_tube\_url.get(), maxResults=50,type='video',publishedAfter=start\_time,publishedBefore=end\_time).execute()

else:

req=youtube.search().list(part='snippet',q=You\_tube\_url.get(),maxResults=50,type='video').execute()

my\_list = []

for item in req['items']:

my\_list.append(item['snippet']['title'])

self.combo = ttk.Combobox(font=('Times New Roman', 15), width=60, textvariable=You\_tube\_url, )

self.combo['values'] = my\_list

self.combo.place(relx=0.3, rely=0.552)

name = self.combo.get()

# req = youtube.search().list(part='snippet', q=You\_tube\_url.get(), maxResults=50,

# type='video').execute()

def pilo():

# self.root = root

# self.root.title("SCREEN 2")

# self.root.geometry("1350x750+0+0")

# self.root.config(bg="cadet blue")

#====================================Functions===============================================================

def iExit():

iExit = tkinter.messagebox.askyesno("You Tube Information System", "confirm if you want to exit")

if iExit > 0:

root.destroy()

return

def clearData():

self.txtYou\_tube\_url.delete(0, END)

def addData():

if len(name) != 0:

for item in req['items']:

if name == item['snippet']['title']:

You\_tube\_url = item['snippet']['channelId']

break

data=urllib.request.urlopen("https://www.googleapis.com/youtube/v3/channels?part=statistics&id=" + You\_tube\_url.get() + "&key=" + api\_key).read()

SUBSCRIBER\_COUNT="SUBSCRIBER\_COUNT:"+str(json.loads(data)['items'][0]["statistics"]["subscriberCount"])

VIEW\_COUNT="VIEW\_COUNT:"+str(json.loads(data)['items'][0]["statistics"]["viewCount"])

VIDEO\_COUNT="VIDEO\_COUNT:"+str(json.loads(data)['items'][0]["statistics"]["videoCount"])

COMMENT\_COUNT="COMMENT\_COUNT:"+str(json.loads(data)['items'][0]["statistics"]["commentCount"])

HIDDEN\_SUBSCRIBER\_COUNT="HIDDEN\_SUBSCRIBER\_COUNT:"+str(json.loads(data)['items'][0]["statistics"]["hiddenSubscriberCount"])

YOU\_ID = " Channel ID : " + str(You\_tube\_url)

TITLE = " VIDEO TITLE :" + str(item['snippet']['title'])

PUBLISHED\_AT = " PUBLISHED\_AT :" + str(item['snippet']['publishedAt'])

youtube\_api\_backend.youtube\_data()

youtube\_api\_backend.addytdRec(TITLE, PUBLISHED\_AT, YOU\_ID, VIEW\_COUNT, COMMENT\_COUNT,SUBSCRIBER\_COUNT, HIDDEN\_SUBSCRIBER\_COUNT, VIDEO\_COUNT)

youtubelist.delete(0, END)

youtubelist.insert(END, (TITLE, PUBLISHED\_AT, YOU\_ID, VIEW\_COUNT, COMMENT\_COUNT,SUBSCRIBER\_COUNT,HIDDEN\_SUBSCRIBER\_COUNT, VIDEO\_COUNT))

def DisplayData():

youtubelist.delete(0, END)

for row in youtube\_api\_backend.viewData():

youtubelist.insert(END, row, str(""))

def youtubeRec(event):

global sd

searchYtd = youtubelist.curselection()[0]

sd = youtubelist.get(searchYtd)

self.txtYou\_tube\_url.delete(0, END)

self.txtYou\_tube\_url.insert(END, sd[0])

def DeleteData():

if len(You\_tube\_url.get()) != 0:

youtube\_api\_backend.deleteRec(sd[1])

clearData()

DisplayData()

def SearchDatabase():

youtubelist.delete(0, END)

for row in youtube\_api\_backend.searchData(sd[0]):

youtubelist.insert(END, row, str(""))

def Sorting():

def youtubeRec2(event):

global sd2

searchYtd2 = youtubelist2.curselection()[0]

sd2 = youtubelist2.get(searchYtd2)

scrollbar2 = Scrollbar(root)

scrollbar2.place(relx=0.002, rely=0.5)

youtubelist2=Listbox(width=177,height=45,font=('arial',12,'bold'),yscrollcommand=scrollbar.set)

youtubelist2.bind('<<ListboxSelect>>', youtubeRec2)

youtubelist2.place(relx=0.02, rely=0.18)

scrollbar2.config(command=youtubelist2.yview)

youtubelist2.delete(0, END)

for row in youtube\_api\_backend.SortData():

youtubelist2.insert(END, row, str(""))

#====================================Frames=====================================

MainFrame = Frame(self.root, bg="TEAL")

MainFrame.grid()

TitFrame = Frame(MainFrame, bd=2, padx=10, pady=8, bg="black", relief=RIDGE)

TitFrame.pack(side=TOP)

self.lblTit = Label(TitFrame, font=('Times New Roman', 47, 'bold'), text="Implementation page",bg="white")

self.lblTit.grid(sticky=W)

ButtonFrame = Frame(MainFrame, bd=2, width=10, height=10, padx=18, pady=10, bg='TEAL', relief=RIDGE)

ButtonFrame.pack(side=BOTTOM)

DataFrame = Frame(MainFrame, bd=1, width=10, height=10, padx=20, pady=20, relief=RIDGE, bg='grey')

DataFrame.pack(side=BOTTOM)

DataFrameLEFT = LabelFrame(DataFrame, bd=1, width=1000, height=600, padx=20, relief=RIDGE, bg='red',font=('comic sans MS', 20, 'bold'), text='student details:-\n')

DataFrameLEFT.pack(side=LEFT)

DataFrameRIGHT = LabelFrame(DataFrame, bd=1, width=670, height=680, padx=31, pady=3, relief=RIDGE,bg='dark blue', font=('comic sans MS', 30, 'bold'),

text='Fetched Data:-\n')

DataFrameRIGHT.pack(side=RIGHT)

# ===============================================LABELS AND\_ENTRY\_WIDGET=======================================================

self.lblYou\_tube\_url = Label(DataFrameLEFT, font=('COMIC SANS MS', 20, 'bold'),

text="Enter youtube link:",

padx=2, pady=2, bg="red")

self.lblYou\_tube\_url.grid(row=0, column=0, sticky=W)

self.txtYou\_tube\_url = Entry(DataFrameLEFT, font=('COMIC SANS MS', 20, 'bold'),textvariable=You\_tube\_url,width=39)

self.txtYou\_tube\_url.grid(row=0, column=1)

self.screen2=ttk.Button(DataFrameLEFT,width=20,text="gotosearchscreen",command=restart\_program)

self.screen2.grid(row=4, column=1)

#==================================================================listBox & scrollBar widget=====================================================

scrollbar = Scrollbar(DataFrameRIGHT)

scrollbar.grid(row=0, column=1, sticky='ns')

youtubelist = Listbox(DataFrameRIGHT, width=41, height=16, font=('arial', 12, 'bold'),

yscrollcommand=scrollbar.set)

youtubelist.bind('<<ListboxSelect>>', youtubeRec)

youtubelist.grid(row=0, column=0, padx=8)

scrollbar.config(command=youtubelist.yview)

#===============================================================Button\_Width=====================================================

self.btnAddData = Button(ButtonFrame, text="Add New", font=('times new roman', 20, 'bold'),height=6, width=13,bd=4, command=addData)

self.btnAddData.grid(row=0, column=0)

self.btnAddData = Button(ButtonFrame, text="Display", font=('times new roman', 20, 'bold'),height=6, width=13,bd=4, command=DisplayData)

self.btnAddData.grid(row=0, column=1)

self.btnAddData = Button(ButtonFrame, text="Clear", font=('times new roman', 20, 'bold'), height=6,width=13,bd=4, command=clearData)

self.btnAddData.grid(row=0, column=2)

self.btnAddData = Button(ButtonFrame, text="Delete", font=('times new roman', 20, 'bold'), height=6,width=13,bd=4, command=DeleteData)

self.btnAddData.grid(row=0, column=3)

self.btnAddData = Button(ButtonFrame, text="Search", font=('times new roman', 20, 'bold'), height=6,width=13,bd=4, command=SearchDatabase)

self.btnAddData.grid(row=0, column=4)

self.btnAddData = Button(ButtonFrame, text="Sorting", font=('times new roman', 20, 'bold'),height=6, width=13,bd=4, command=Sorting)

self.btnAddData.grid(row=0, column=5)

self.btnAddData = Button(ButtonFrame, text="Exit", font=('times new roman', 20, 'bold'), height=6,width=13,bd=4, command=iExit)

self.btnAddData.grid(row=0, column=6)

self.l2 = ttk.Label(self.root, font=('Times New Roman', 9, 'bold'),text="<----press select one more time")

self.l2.place(relx=0.8, rely=0.515)

self.screen2 = tk.Button(width=25, height=5, text="go to 2nd screen ----->", command=pilo)

self.screen2.place(relx=0.8, rely=0.8)

self.screen1 = tk.Button(width=15, height=2, text="Select", command=fun)

self.screen1.place(relx=0.71, rely=0.502)

funky()

if \_\_name\_\_ == '\_\_main\_\_':

# app = tk.Tk()

root = tk.Tk()

application = hello(root)

root.mainloop()

**# BACKEND:-**

import sqlite3

def youtube\_data():

con = sqlite3.connect("youtube6.db")

cur = con.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS new1\_youtube (TITLE TEXT , YOU\_ID TEXT PRIMARY KEY,PUBLISHED\_AT VARCHAR,VIEW\_COUNT INTEGER,COMMENT\_COUNT INTEGER, SUBSCRIBER\_COUNT INTEGER,HIDDEN\_SUBSCRIBER\_COUNT INTEGER,VIDEO\_COUNT INTEGER)")

con.commit()

con.close()

def addytdRec(TITLE, PUBLISHED\_AT, YOU\_ID, VIEW\_COUNT, COMMENT\_COUNT, SUBSCRIBER\_COUNT, HIDDEN\_SUBSCRIBER\_COUNT,VIDEO\_COUNT):

con = sqlite3.connect("youtube6.db")

cur = con.cursor()

cur.execute("INSERT INTO new1\_youtube VALUES (?,?,?,?,?,?,?,?)",

(TITLE, PUBLISHED\_AT, YOU\_ID, VIEW\_COUNT, COMMENT\_COUNT, SUBSCRIBER\_COUNT,HIDDEN\_SUBSCRIBER\_COUNT, VIDEO\_COUNT))

con.commit()

con.close()

def viewData():

con = sqlite3.connect("youtube6.db")

cur = con.cursor()

cur.execute("SELECT \* FROM new1\_youtube")

rows = cur.fetchall()

con.close()

return rows

def deleteRec(YOU\_ID):

con = sqlite3.connect("youtube6.db")

cur = con.cursor()

cur.execute("DELETE FROM new1\_youtube WHERE YOU\_ID=?",(YOU\_ID,))

con.commit()

con.close()

def searchData():

con = sqlite3.connect("youtube6.db")

cur = con.cursor()

cur.execute("SELECT \* FROM new1\_youtube WHERE TITLE=?", (TITLE,))

rows = cur.fetchall()

con.close()

return rows

def SortData():

con = sqlite3.connect("youtube6.db")

cur = con.cursor()

cur.execute("SELECT \*FROM new1\_youtube ORDER BY TITLE")

# cur.execute("SELECT \* FROM new\_youtube ORDERBY CONVERT(DateTime, PUBLISHED\_AT,101) DESC")

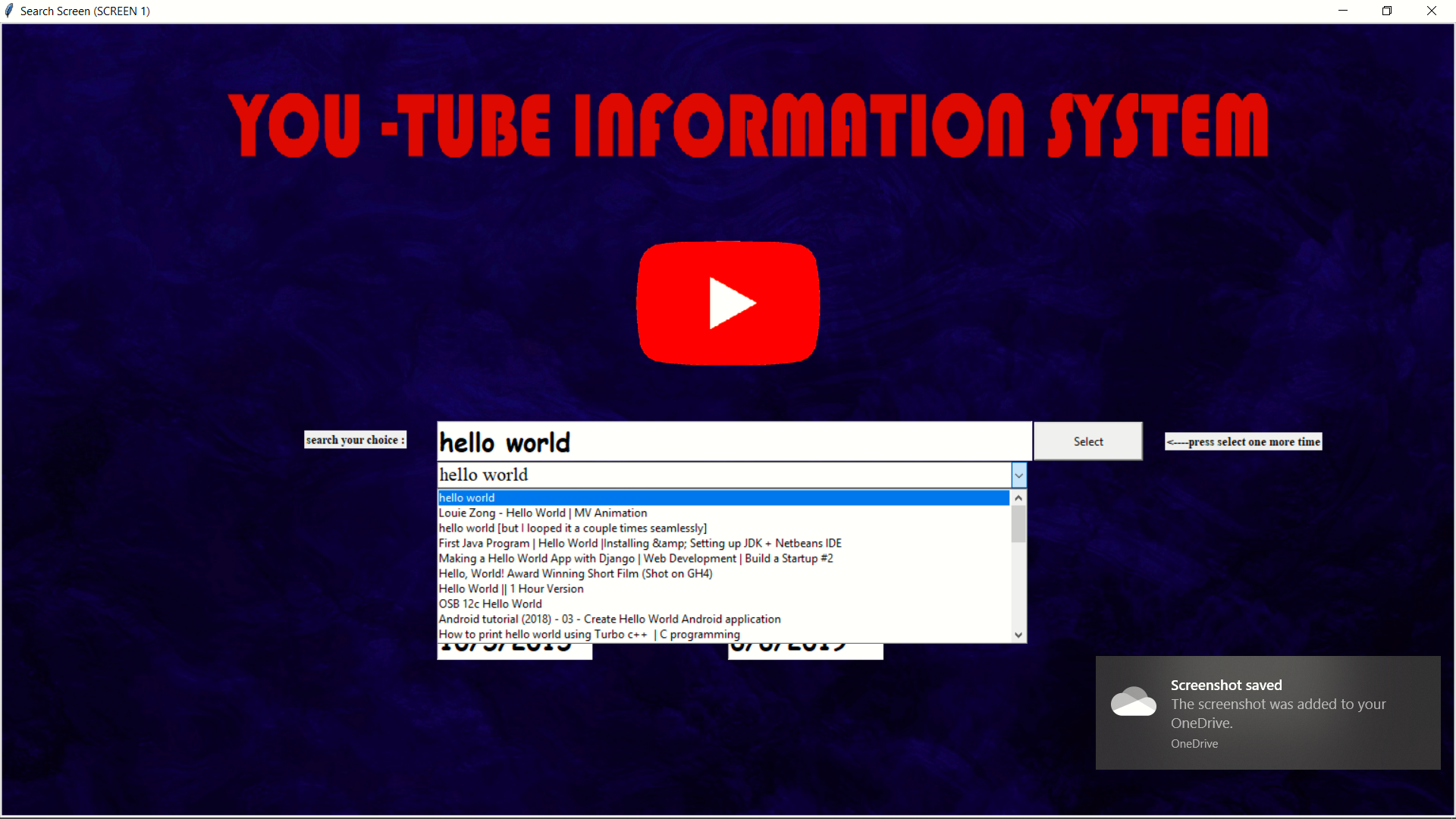
rows = cur.fetchall()

con.close()

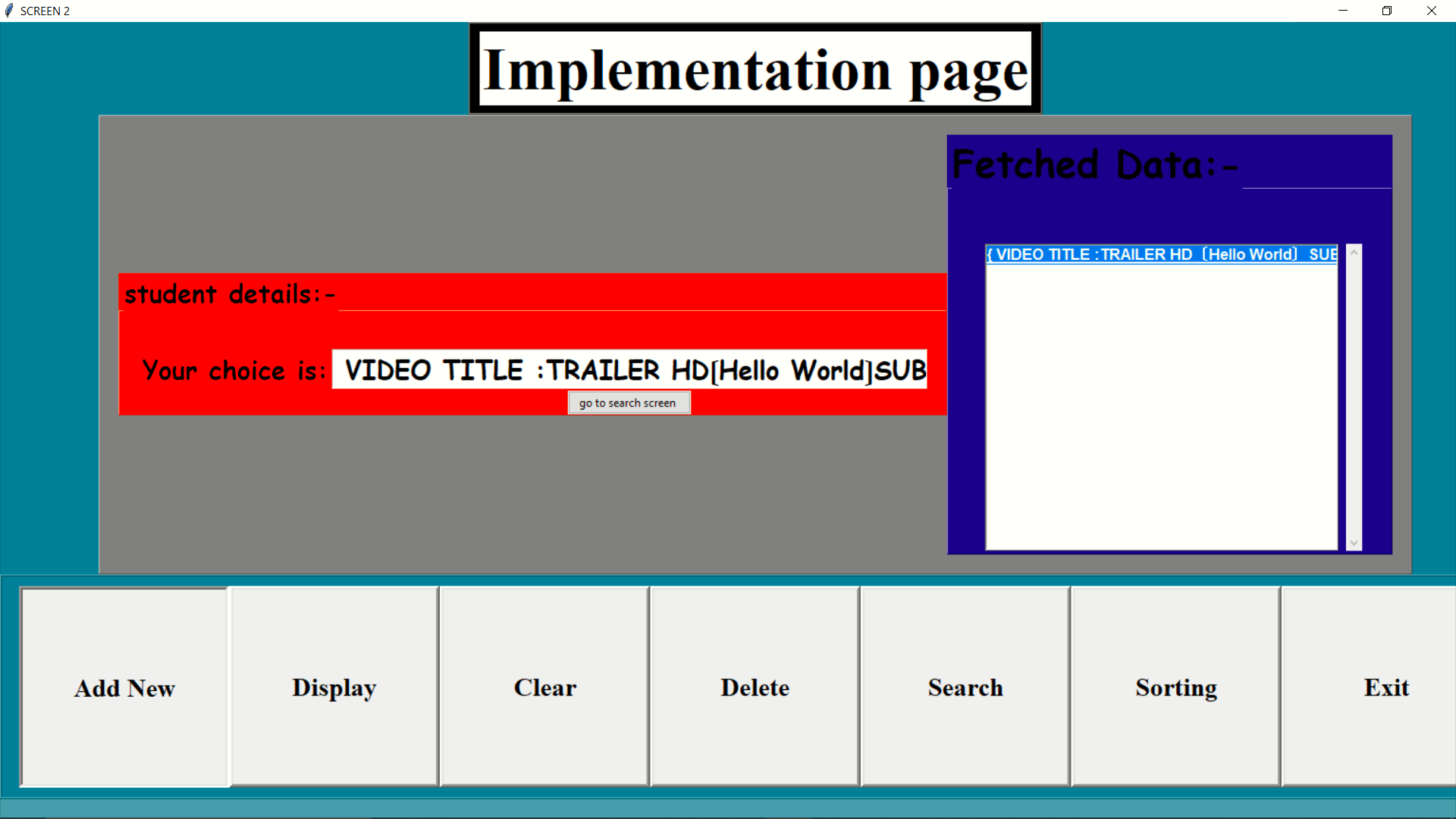
return rows

**SCREENSHOTS**

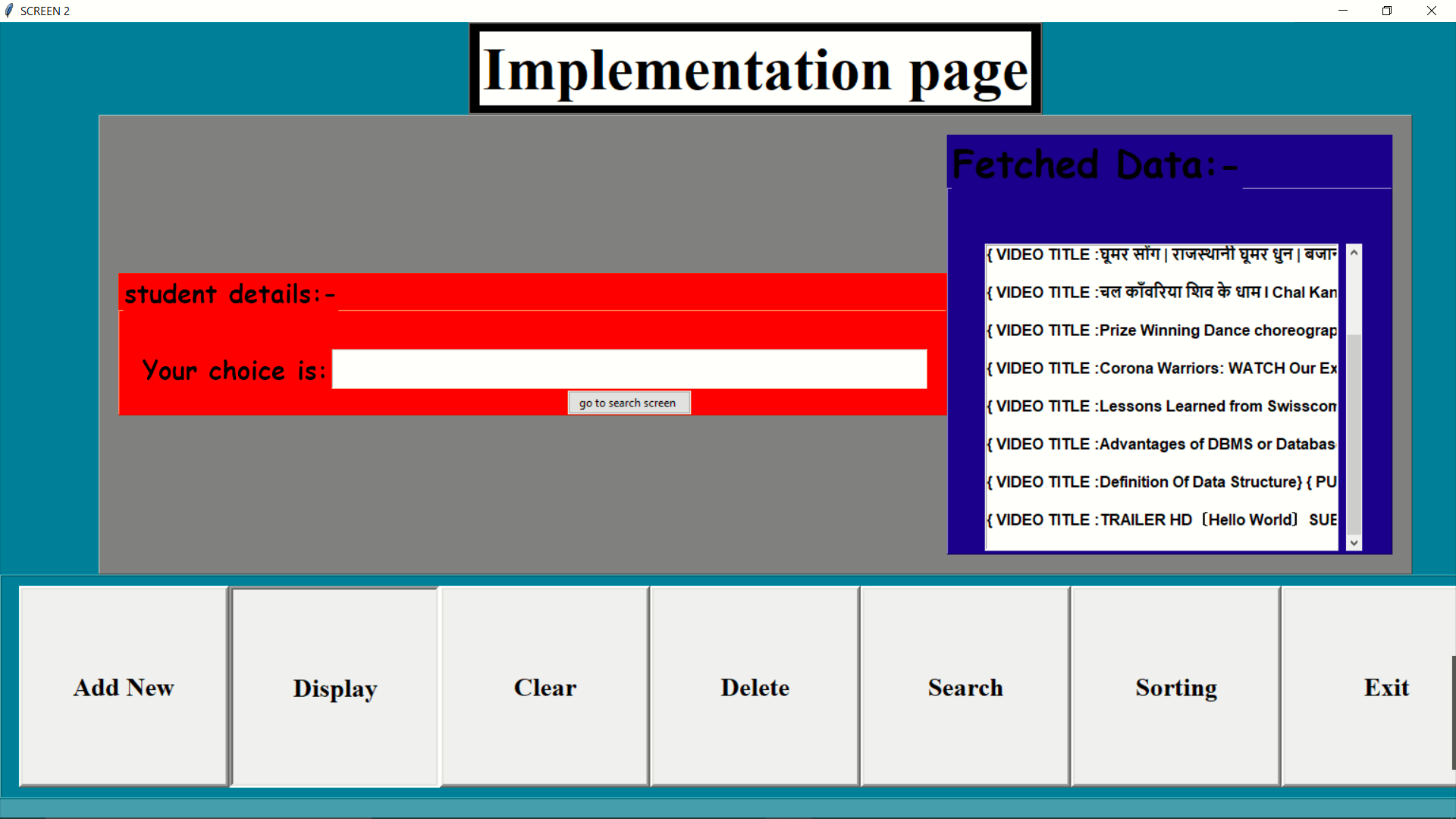
* **SCREEN 1(SEARCH PAGE):-**

****

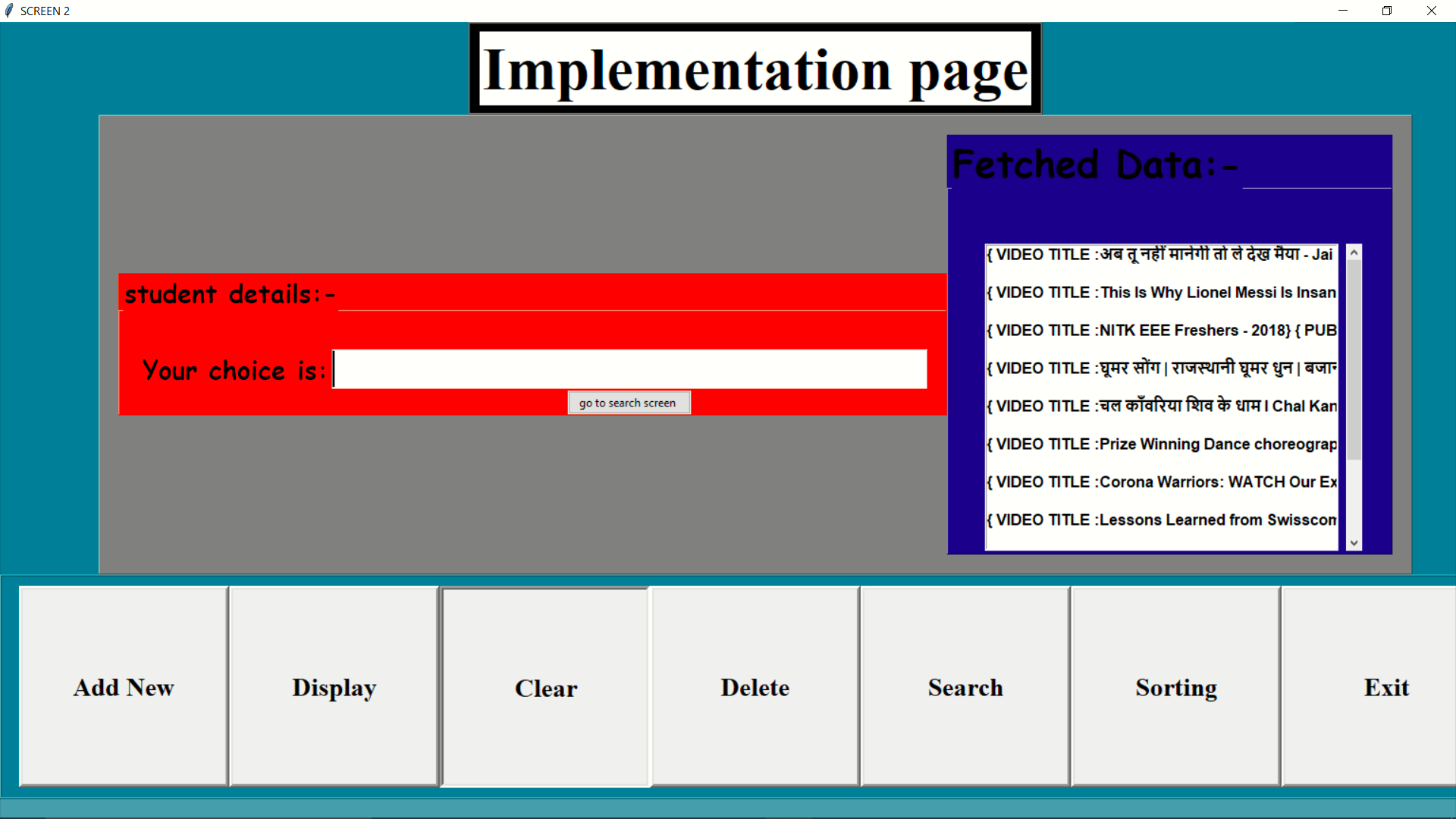
* **SCREEN 2(IMPLEMENTATION PAGE):-**
* **ADD BUTTON :-**



* **DISPLAY BUTTON:-**

****

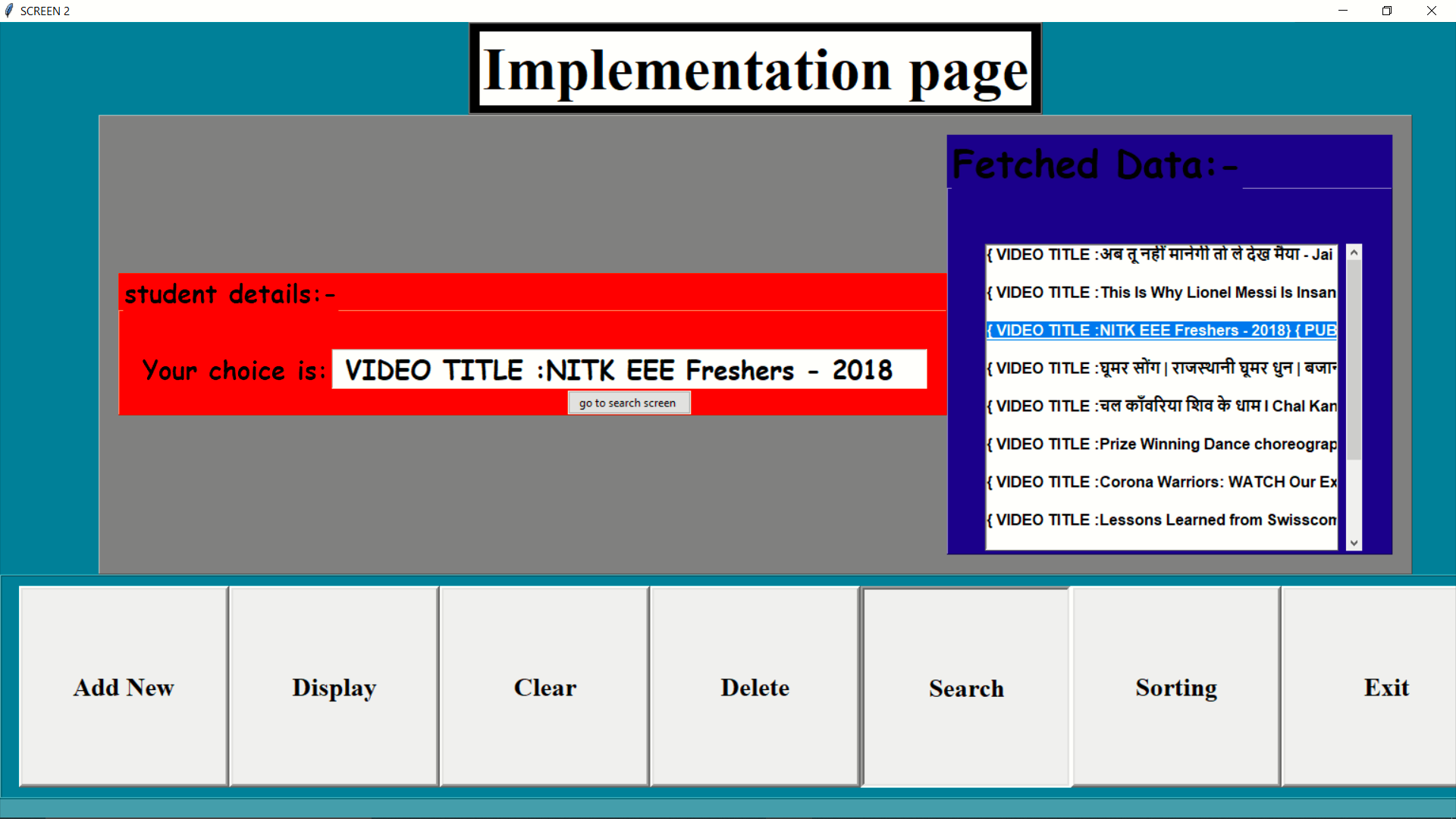
* **CLEAR BUTTON:-**

****

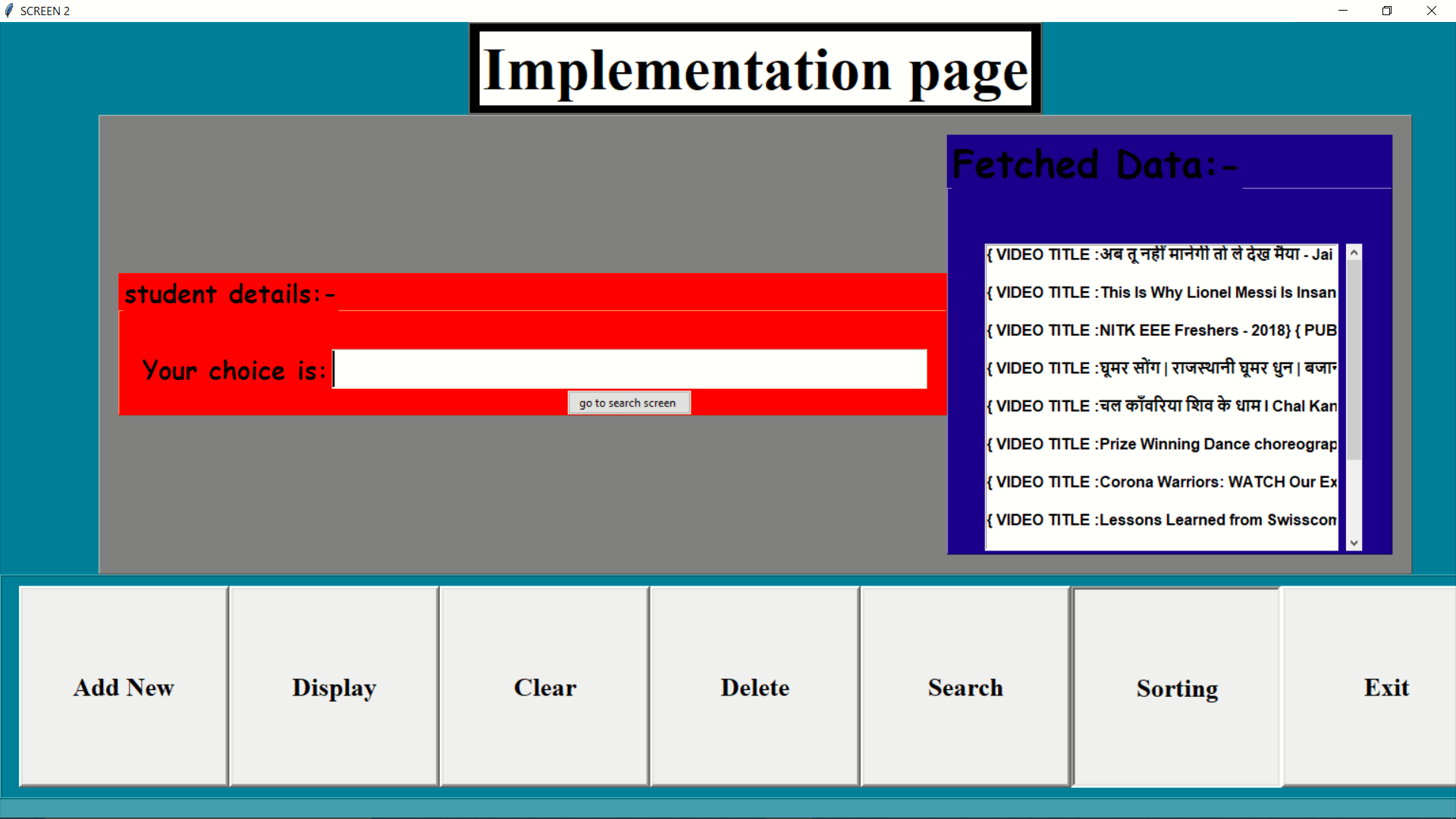
* **DELETE BUTTON :-**

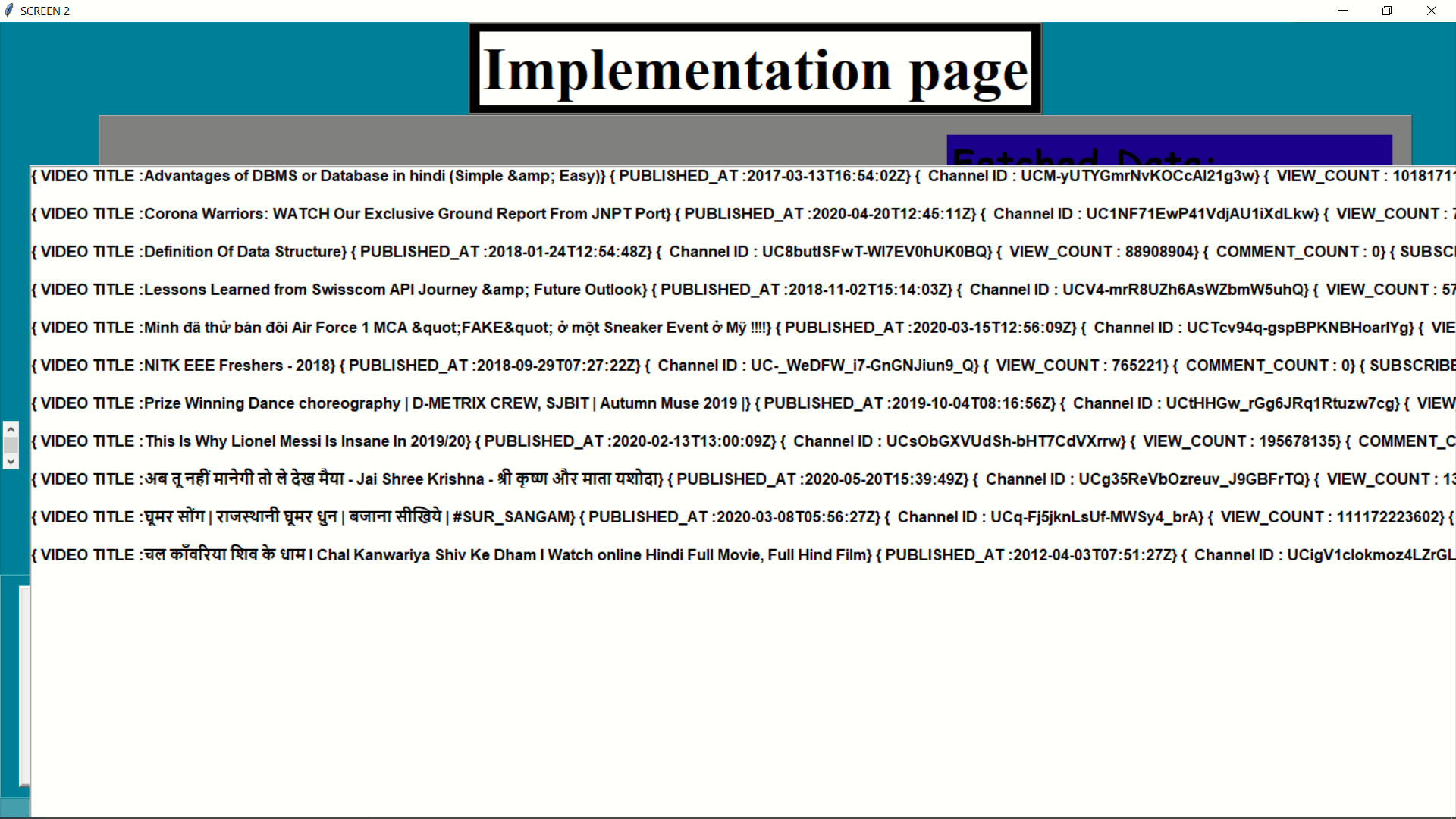
****

* **SEARCH BUTTON:-**

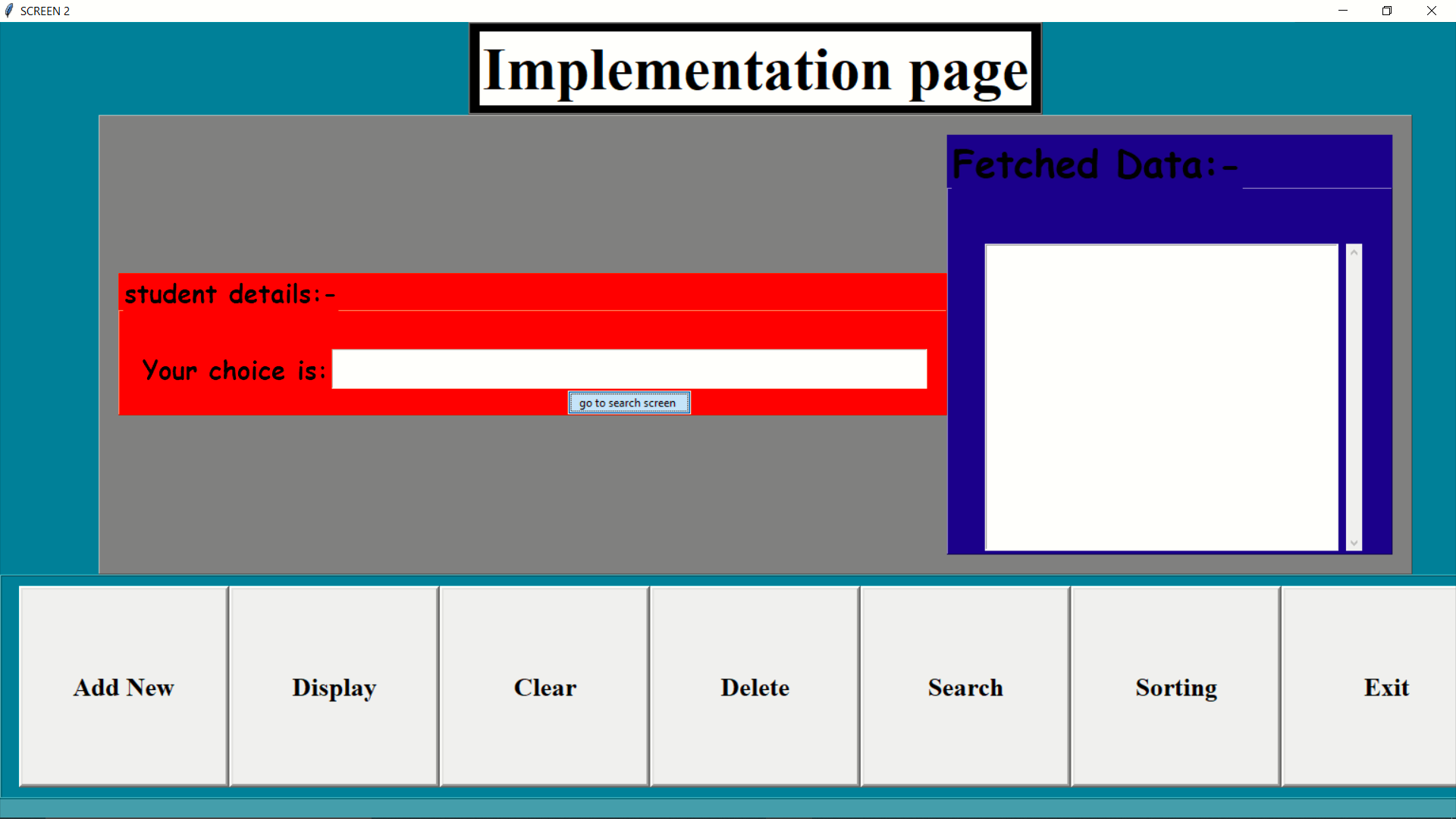
****

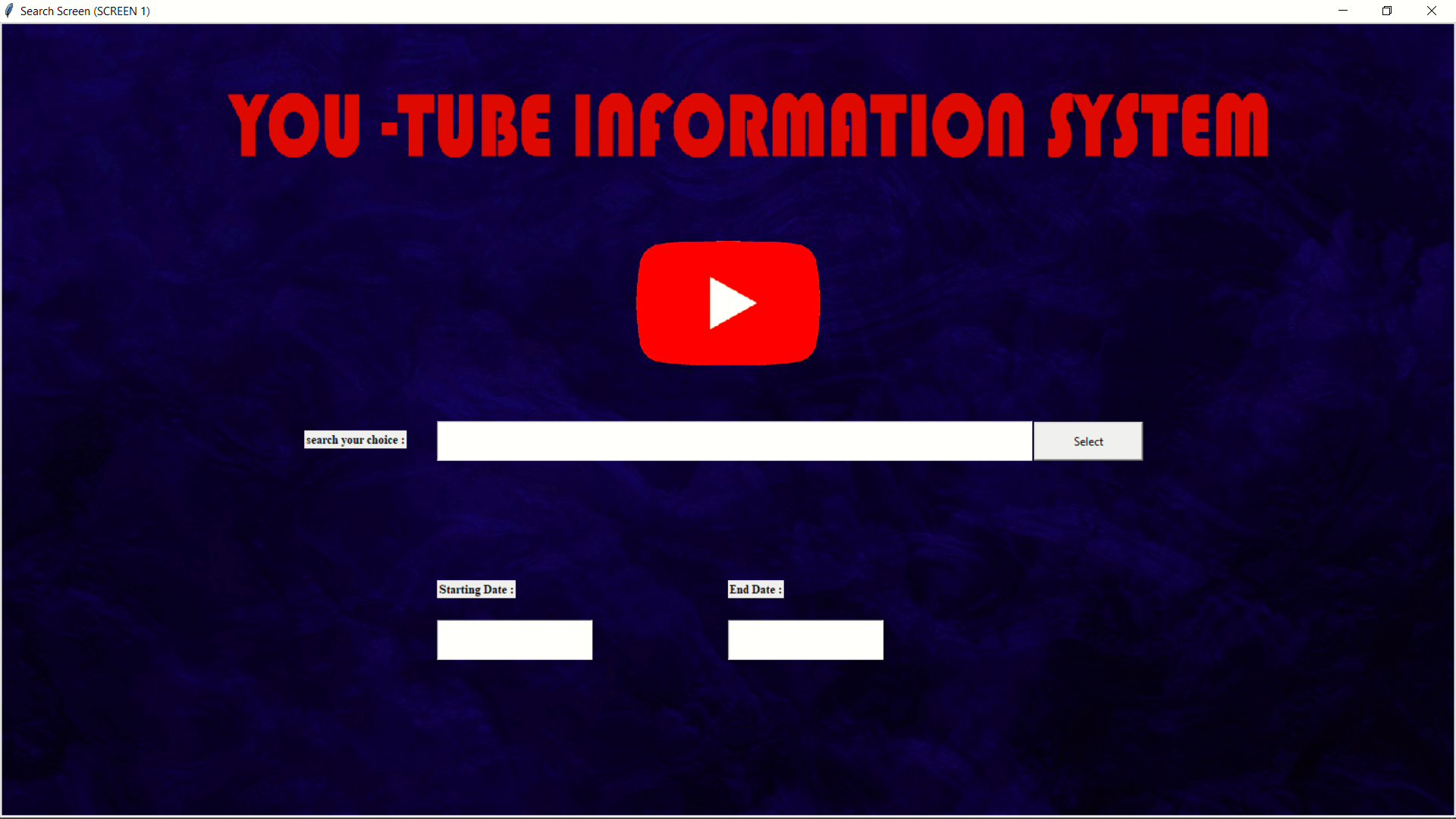
* **SORTING BUTTON:- (SORT WITH RESPECT TO TITLE)**

****

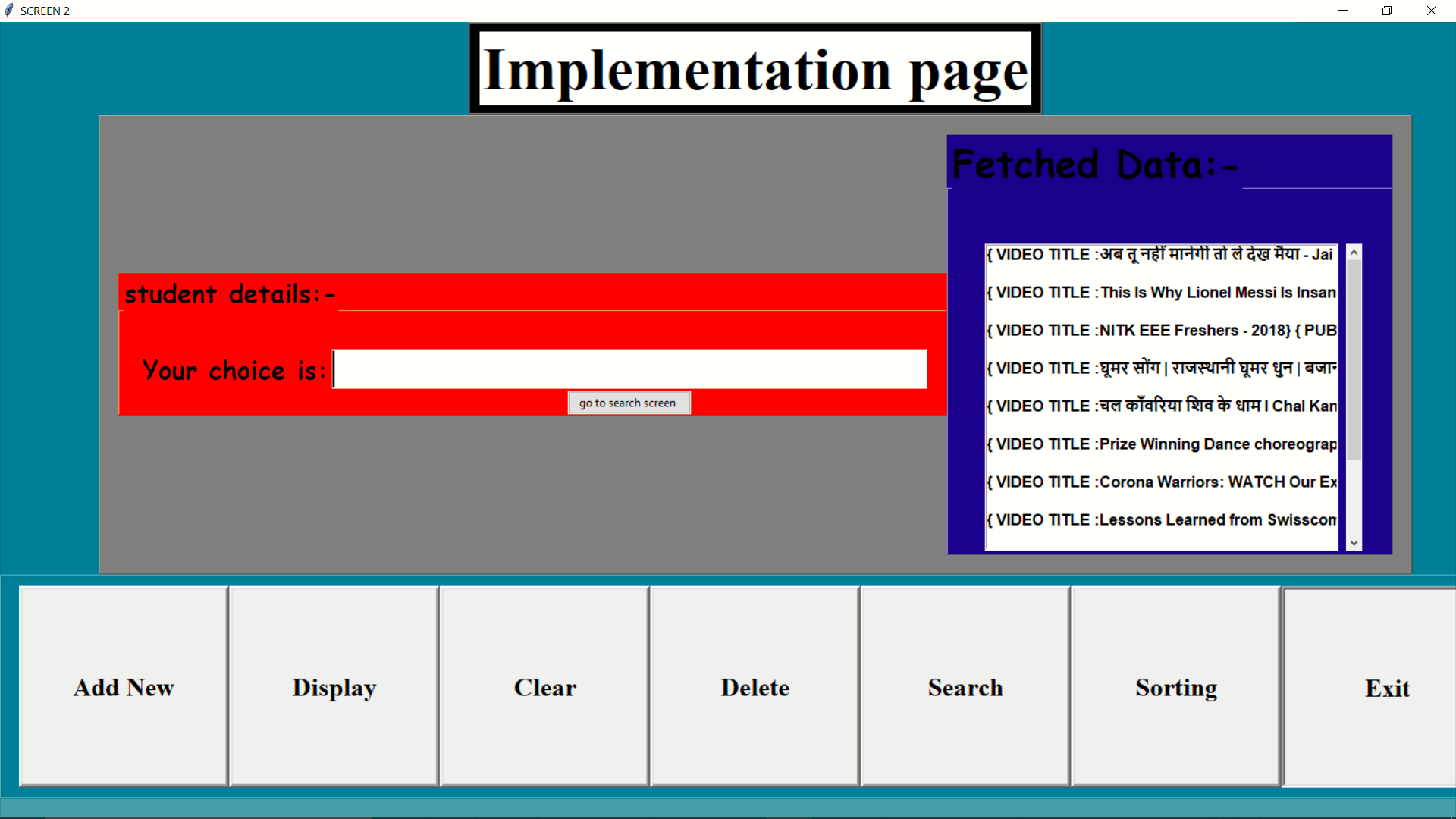
****

* **GO TO 1 ST SCREEN BUTTON :-**

****

****

* **EXIT BUTTON:-(END PROJECT)**

****

**FUTURE ASPECTS**

The future of APIs are conversational APIs. Current API’S are set to naturally evolve towards conversation APIs, interfaces that perform searches and provide responses to queries via conversational user interfaces. Innovation depends on early adoption of APIs as a means of launching products, and the integration of an open developer community within work processes.

**CONCLUSION**

In this project we have successfully implemented Youtube V3 api key in our project to see the youtube’s channel related data which is easy to access and user friendly. And here we also implemented Tkinter in our Frontend which is a Python binding to the Tk GUI toolkit. It is the standard python interfaces to the Tk GUI toolkit, with this also able to implement sqlite3 successfully in our backend.

**REFERENCES**

1. <https://www.w3schools.com>
2. <https://www.tutorialspoint.com/sqlite/sqlite_python.htm>