

```
1 import streamlit as st
2 from streamlit_option_menu import option_menu
3 import pymongo
4 from pymongo import MongoClient # Added this import
5 import psycopg2
6 import pandas as pd
7 from googleapiclient.discovery import build
8
9
10
11
12 #API key connection
13 def Api_connect():
14     Api_Id="AIzaSyCRlE3-Ly-iitIQrSWM_M3zF-E6BNR9IXQ"
15
16     api_service_name = "youtube"
17     api_version = "v3"
18     youtube = build(api_service_name,api_version,developerKey=Api_Id)
19     return youtube
20
21 youtube=Api_connect()
22
23
24 #CHANNEL INFORMATION
25 def get_channel_info(channel_id):
26
27     request = youtube.channels().list(
28         part = "snippet,contentDetails,Statistics",
29         id = channel_id)
30
31     response1=request.execute()
32
33     for i in range(0,len(response1["items"])):
34         data = dict(
35             Channel_Name = response1["items"][i]["snippet"]["title"],
36             Channel_Id = response1["items"][i]["id"],
37             Subscription_Count= response1["items"][i]["statistics"]
38             ["subscriberCount"],
39             Views = response1["items"][i]["statistics"]["viewCount"],
40             Total_Videos = response1["items"][i]["statistics"]
41             ["videoCount"],
42             Channel_Description = response1["items"][i]["snippet"]
43             ["description"],
44             Playlist_Id = response1["items"][i]["contentDetails"]
45             ["relatedPlaylists"]["uploads"],
46         )
47         return data
48
49
50 #PLAYLISTS INFORMATION
51 def get_playlist_info(channel_id):
52     All_data = []
53     next_page_token = None
54     next_page = True
55     while next_page:
56
57         request = youtube.playlists().list(
58             part="snippet,contentDetails",
59             channelId=channel_id,
60             maxResults=50,
```

```

57         pageToken=next_page_token
58     )
59     response = request.execute()
60
61     for item in response['items']:
62         data={'PlaylistId':item['id'],
63             'Title':item['snippet']['title'],
64             'ChannelId':item['snippet']['channelId'],
65             'ChannelName':item['snippet']['channelTitle'],
66             'PublishedAt':item['snippet']['publishedAt'],
67             'VideoCount':item['contentDetails']['itemCount']}
68         All_data.append(data)
69     next_page_token = response.get('nextPageToken')
70     if next_page_token is None:
71         next_page=False
72     return All_data
73
74 #CHANNEL VIDEO INFORMATION
75 def get_channel_videos(channel_id):
76     video_ids = []
77     # get Uploads playlist id
78     res = youtube.channels().list(id=channel_id,
79                                 part='contentDetails').execute()
80     playlist_id = res['items'][0]['contentDetails']['relatedPlaylists']['uploads']
81     next_page_token = None
82
83     while True:
84         res = youtube.playlistItems().list(
85             part = 'snippet',
86             playlistId = playlist_id,
87             maxResults = 50,
88             pageToken = next_page_token).execute()
89
90         for i in range(len(res['items'])):
91             video_ids.append(res['items'][i]['snippet']['resourceId']['videoId'])
92         next_page_token = res.get('nextPageToken')
93
94         if next_page_token is None:
95             break
96     return video_ids
97
98 #VIDEO_INFORMATION
99 def get_video_info(video_ids):
100
101     video_data = []
102
103     for video_id in video_ids:
104         request = youtube.videos().list(
105             part="snippet,contentDetails,statistics",
106             id= video_id)
107         response = request.execute()
108
109         for item in response["items"]:
110             data = dict(Channel_Name = item['snippet']['channelTitle'],
111                         Channel_Id = item['snippet']['channelId'],
112                         Video_Id = item['id'],
113                         Title = item['snippet']['title'],
114                         Tags = item['snippet'].get('tags'),
115                         Thumbnail = item['snippet']['thumbnails']['default']['url'],
116                         Description = item['snippet']['description'],

```

```

117         Published_Date = item['snippet']['publishedAt'],
118         Duration = item['contentDetails']['duration'],
119         Views = item['statistics']['viewCount'],
120         Likes = item['statistics'].get('likeCount'),
121         Comments = item['statistics'].get('commentCount'),
122         Favorite_Count = item['statistics']['favoriteCount'],
123         Definition = item['contentDetails']['definition'],
124         Caption_Status = item['contentDetails']['caption']
125     )
126     video_data.append(data)
127     return video_data
128 #COMMENT INFORMATION
129 def get_comment_info(video_ids):
130     Comment_Information = []
131     try:
132         for video_id in video_ids:
133
134             request = youtube.commentThreads().list(
135                 part = "snippet",
136                 videoId = video_id,
137                 maxResults = 50
138             )
139             response5 = request.execute()
140
141             for item in response5["items"]:
142                 comment_information = dict(
143                     Comment_Id = item["snippet"]["topLevelComment"]
144                     ["id"],
145                     Video_Id = item["snippet"]["videoId"],
146                     Comment_Text = item["snippet"]
147                     ["topLevelComment"]["snippet"]["textOriginal"],
148                     Comment_Author = item["snippet"]
149                     ["topLevelComment"]["snippet"]["authorDisplayName"],
150                     Comment_Published = item["snippet"]
151                     ["topLevelComment"]["snippet"]["publishedAt"])
152
153                 Comment_Information.append(comment_information)
154
155     except:
156         pass
157
158     return Comment_Information
159
160 #MongoDB Connection
161 client =
162     pymongo.MongoClient("mongodb+srv://sabarishraja:youtube@youtube.u3ogmr.mongodb.net/
163     ?retryWrites=true&w=majority")
164 db = client["Youtube_data"]
165
166 # upload to MongoDB
167
168 def channel_details(channel_id):
169     ch_details = get_channel_info(channel_id)
170     pl_details = get_playlist_info(channel_id)
171     vi_ids = get_channel_videos(channel_id)
172     vi_details = get_video_info(vi_ids)
173     com_details = get_comment_info(vi_ids)
174
175     coll1 = db["channel_details"]

```

```

170 coll1.insert_one({"channel_information":ch_details,"playlist_information":pl_detail
s,"video_information":vi_details,
171                     "comment_information":com_details})
172
173     return "upload completed successfully"
174
175 #Table creation for channels,playlists, videos, comments
176 def channels_table():
177     mydb = psycopg2.connect(host="localhost",
178                             user="postgres",
179                             password="postgresql",
180                             database= "Youtube_data",
181                             port = "5432"
182                             )
183     cursor = mydb.cursor()
184
185     drop_query = "drop table if exists channels"
186     cursor.execute(drop_query)
187     mydb.commit()
188
189     try:
190         create_query = '''CREATE TABLE IF NOT EXISTS channels(
191                             Channel_Name VARCHAR(100),
192                             Channel_Id VARCHAR(80) PRIMARY KEY,
193                             Subscription_Count BIGINT,
194                             Views BIGINT,
195                             Total_Videos INT,
196                             Channel_Description TEXT,
197                             Playlist_Id VARCHAR(50))'''
198
199         cursor.execute(create_query)
200         mydb.commit()
201     except:
202         st.write("Channels Table already created")
203
204
205     ch_list = []
206     db = client["Youtube_data"]
207     coll1 = db["channel_details"]
208
209     for ch_data in coll1.find({}, {"_id": 0, "channel_information": 1}):
210         ch_list.append(ch_data["channel_information"])
211
212     df = pd.DataFrame(ch_list)
213
214     for index, row in df.iterrows():
215         insert_query = '''INSERT INTO channels(Channel_Name,
216                                                 Channel_Id,
217                                                 Subscription_Count,
218                                                 Views,
219                                                 Total_Videos,
220                                                 Channel_Description,
221                                                 Playlist_Id)
222                             VALUES (%s, %s, %s, %s, %s, %s, %s)'''
223
224         values = (
225             row['Channel_Name'],
226             row['Channel_Id'],
227             row['Subscription_Count'],

```

```

228         row['Views'],
229         row['Total_Videos'],
230         row['Channel_Description'],
231         row['Playlist_Id']
232     )
233     try:
234         cursor.execute(insert_query, values)
235         mydb.commit()
236     except:
237         st.write("Channels values are already inserted")
238
239
240 def playlists_table():
241     mydb = psycopg2.connect(host="localhost",
242                             user="postgres",
243                             password="postgresql",
244                             database="Youtube_data",
245                             port="5432"
246                             )
247     cursor = mydb.cursor()
248     drop_query = "drop table if exists playlists"
249     cursor.execute(drop_query)
250     mydb.commit()
251
252     try:
253         create_query = '''create table if not exists playlists(PlaylistId
254                                                                    Title varchar(80),
255                                                                    ChannelId
256                                                                    varchar(100),
257                                                                    ChannelName
258                                                                    varchar(100),
259                                                                    PublishedAt
260                                                                    timestamp,
261                                                                    VideoCount int
262                                                                    )'''
263         cursor.execute(create_query)
264         mydb.commit()
265     except:
266         st.write("Playlists Table already created")
267
268     db = client["Youtube_data"]
269     coll1 = db["channel_details"]
270     pl_list = []
271     for pl_data in coll1.find({}, {"_id": 0, "playlist_information": 1}):
272         for i in range(len(pl_data["playlist_information"])):
273             pl_list.append(pl_data["playlist_information"][i])
274     df2 = pd.DataFrame(pl_list)
275
276     for index, row in df2.iterrows():
277         insert_query = '''INSERT into playlists(PlaylistId,
278                                                                    Title,
279                                                                    ChannelId,
280                                                                    ChannelName,
281                                                                    PublishedAt,
282                                                                    VideoCount)
283                                                                    VALUES(%s,%s,%s,%s,%s,%s,%s)'''

```

```

284     values =(
285         row['PlaylistId'],
286         row['Title'],
287         row['ChannelId'],
288         row['ChannelName'],
289         row['PublishedAt'],
290         row['VideoCount'])
291
292     try:
293         cursor.execute(insert_query,values)
294         mydb.commit()
295     except:
296         st.write("Playlists values are already inserted")
297
298 def videos_table():
299
300     mydb = psycopg2.connect(host="localhost",
301                             user="postgres",
302                             password="postgresql",
303                             database= "Youtube_data",
304                             port = "5432"
305                             )
306     cursor = mydb.cursor()
307
308     drop_query = "drop table if exists videos"
309     cursor.execute(drop_query)
310     mydb.commit()
311
312
313
314     try:
315         create_query = '''create table if not exists videos(
316                             Channel_Name varchar(150),
317                             Channel_Id varchar(100),
318                             Video_Id varchar(50) primary key,
319                             Title varchar(150),
320                             Tags text,
321                             Thumbnail varchar(225),
322                             Description text,
323                             Published_Date timestamp,
324                             Duration interval,
325                             Views bigint,
326                             Likes bigint,
327                             Comments int,
328                             Favorite_Count int,
329                             Definition varchar(10),
330                             Caption_Status varchar(50)
331                             )'''
332
333         cursor.execute(create_query)
334         mydb.commit()
335     except:
336         st.write("Videos Table already created")
337
338     vi_list = []
339     db = client["Youtube_data"]
340     coll1 = db["channel_details"]
341     for vi_data in coll1.find({},{"_id":0,"video_information":1}):
342         for i in range(len(vi_data["video_information"])):
343             vi_list.append(vi_data["video_information"][i])

```

```
344     df3 = pd.DataFrame(vi_list)
345
346
347     for index, row in df3.iterrows():
348
349         insert_query = '''INSERT INTO videos(Channel_Name,
350                                             Channel_Id,
351                                             Video_Id,
352                                             Title,
353                                             Tags,
354                                             Thumbnail,
355                                             Description,
356                                             Published_Date,
357                                             Duration,
358                                             Views,
359                                             Likes,
360                                             Comments,
361                                             Favorite_Count,
362                                             Definition,
363                                             Caption_Status)
364                                     VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s,
365                                     %s, %s)'''
366         values = (
367             row['Channel_Name'],
368             row['Channel_Id'],
369             row['Video_Id'],
370             row['Title'],
371             row['Tags'],
372             row['Thumbnail'],
373             row['Description'],
374             row['Published_Date'],
375             row['Duration'],
376             row['Views'],
377             row['Likes'],
378             row['Comments'],
379             row['Favorite_Count'],
380             row['Definition'],
381             row['Caption_Status']
382         )
383
384         try:
385             cursor.execute(insert_query, values)
386             mydb.commit()
387         except:
388             st.write("videos values already inserted in the table")
389
390 def comments_table():
391
392     mydb = psycopg2.connect(host="localhost",
393                             user="postgres",
394                             password="postgresql",
395                             database="Youtube_data",
396                             port="5432"
397                             )
398     cursor = mydb.cursor()
399
400     drop_query = "drop table if exists comments"
401     cursor.execute(drop_query)
402     mydb.commit()
```

```

403
404     try:
405         create_query = '''CREATE TABLE if not exists comments(Comment_Id
varchar(100) primary key,
406                             Video_Id varchar(80),
407                             Comment_Text text,
408                             Comment_Author varchar(150),
409                             Comment_Published timestamp)'''
410         cursor.execute(create_query)
411         mydb.commit()
412
413     except:
414         st.write("Comments Table already created")
415
416     com_list = []
417     db = client["Youtube_data"]
418     coll1 = db["channel_details"]
419     for com_data in coll1.find({},{"_id":0,"comment_information":1}):
420         for i in range(len(com_data["comment_information"])):
421             com_list.append(com_data["comment_information"][i])
422     df4 = pd.DataFrame(com_list)
423
424     for index, row in df4.iterrows():
425         insert_query = '''
426             INSERT INTO comments (Comment_Id,
427                                   Video_Id ,
428                                   Comment_Text,
429                                   Comment_Author,
430                                   Comment_Published)
431             VALUES (%s, %s, %s, %s, %s)
432
433         ...
434         values = (
435             row['Comment_Id'],
436             row['Video_Id'],
437             row['Comment_Text'],
438             row['Comment_Author'],
439             row['Comment_Published']
440         )
441         cursor.execute(insert_query,values)
442         mydb.commit()
443     try:
444         cursor.execute(insert_query,values)
445         mydb.commit()
446     except:
447         st.write("This comments are already exist in comments table")
448
449 def tables():
450     channels_table()
451     playlists_table()
452     videos_table()
453     comments_table()
454
455     return "Tables created successfully"
456
457 # ... (previous code)
458
459 def show_channels_table():
460     ch_list = []
461     db = client["Youtube_data"]

```



```

462     coll1 = db["channel_details"]
463     for ch_data in coll1.find({}, {"_id": 0, "channel_information": 1}):
464         ch_list.append(ch_data["channel_information"])
465     channels_table=st.dataframe(ch_list)
466
467     return channels_table
468
469 def show_playlists_table():
470     pl_list = []
471     db = client["Youtube_data"]
472     coll1 = db["channel_details"]
473     for pl_data in coll1.find({}, {"_id": 0, "playlist_information": 1}):
474         for i in range(len(pl_data["playlist_information"])):
475             pl_list.append(pl_data["playlist_information"][i])
476     playlists_table=st.dataframe(pl_list)
477     return playlists_table
478
479 def show_videos_table():
480     vi_list = []
481     db = client["Youtube_data"]
482     coll1 = db["channel_details"]
483     for vi_data in coll1.find({}, {"_id": 0, "video_information": 1}):
484         for i in range(len(vi_data["video_information"])):
485             vi_list.append(vi_data["video_information"][i])
486     videos_table=st.dataframe(vi_list)
487     return videos_table
488
489 def show_comments_table():
490     com_list = []
491     db = client["Youtube_data"]
492     coll1 = db["channel_details"]
493     for com_data in coll1.find({}, {"_id": 0, "comment_information": 1}):
494         for i in range(len(com_data["comment_information"])):
495             com_list.append(com_data["comment_information"][i])
496     comments_table=st.dataframe(com_list)
497     return comments_table
498
499 class HomePage:
500     def show(self):
501         st.title(":blue[YOUTUBE DATA HARVESTING] 🇮🇳")
502         if st.button(":white[welcome]"):
503             st.balloons()
504             st.write({"Name": "Sabarish", {"Batch No": "D104"}, {"Project":
"Youtube Data Harvesting"}})
505
506 # ... (previous code)
507
508 class ProjectPage:
509     def show(self):
510         st.title("Welcome to :red[Project Section] 🧡")
511         channel_id = st.text_input("Enter the Channel id")
512         channels = channel_id.split(',')
513         channels = [ch.strip() for ch in channels if ch]
514
515         if st.button("Collect and Store data"):
516             for channel in channels:
517                 ch_ids = []
518                 db = client["Youtube_data"]
519                 coll1 = db["channel_details"]
520                 for ch_data in coll1.find({}, {"_id": 0, "channel_information": 1}):

```

```

521         ch_ids.append(ch_data["channel_information"]["Channel_Id"])
522     if channel in ch_ids:
523         st.success("Channel details of the given channel id: " + channel
+ " already exist")
524     else:
525         output = channel_details(channel)
526         st.success(output)
527
528     if st.button("Migrate to SQL"):
529         Tables = tables()
530         st.success(Tables)
531
532     show_table = st.radio("SELECT THE TABLE FOR VIEW",
533                           (":green[channels]", ":orange[playlists]",
":red[videos]", ":blue[comments]"))
534
535     if show_table == ":green[channels]":
536         show_channels_table()
537     elif show_table == ":orange[playlists]":
538         show_playlists_table()
539     elif show_table == ":red[videos]":
540         show_videos_table()
541     elif show_table == ":blue[comments]":
542         show_comments_table()
543
544     # SQL connection
545     mydb = psycopg2.connect(host="localhost",
546                             user="postgres",
547                             password="postgresql",
548                             database="Youtube_data",
549                             port="5432")
550     cursor = mydb.cursor()
551
552     question = st.selectbox(
553         'please select your queries',
554         (
555             "1. What are the names of all the videos and their corresponding
channels?",
556             "2. Which channels have the most number of videos, and how many
videos do they have?",
557             "3. What are the top 10 most viewed videos and their respective
channels?",
558             "4. How many comments were made on each video, and what are their
corresponding video names?",
559             "5. Which videos have the highest number of likes, and what are
their corresponding channel names?",
560             "6. What is the total number of likes and dislikes for each video,
and what are their corresponding video names?",
561             "7. What is the total number of views for each channel, and what are
their corresponding channel names?",
562             "8. What are the names of all the channels that have published
videos in the year 2022?",
563             "9. What is the average duration of all videos in each channel, and
what are their corresponding channel names?",
564             "10. Which videos have the highest number of comments, and what are
their corresponding channel names?"
565         )
566     )
567 )

```

```
568         if question == '1. What are the names of all the videos and their
corresponding channels?':
569             query1 = "select Title as videos, Channel_Name as ChannelName from
videos;"
570             cursor.execute(query1)
571             mydb.commit()
572             t1=cursor.fetchall()
573             st.write(pd.DataFrame(t1, columns=["Video Title","Channel Name"]))
574
575         elif question == '2. Which channels have the most number of videos, and how
many videos do they have?':
576             query2 = "select Channel_Name as ChannelName,Total_Videos as NO_Videos
from channels order by Total_Videos desc;"
577             cursor.execute(query2)
578             mydb.commit()
579             t2=cursor.fetchall()
580             st.write(pd.DataFrame(t2, columns=["Channel Name","No Of Videos"]))
581
582         elif question == '3. What are the top 10 most viewed videos and their
respective channels?':
583             query3 = '''select Views as views , Channel_Name as ChannelName,Title as
VideoTitle from videos
584                             where Views is not null order by Views desc limit
10;'''
585             cursor.execute(query3)
586             mydb.commit()
587             t3 = cursor.fetchall()
588             st.write(pd.DataFrame(t3, columns = ["views","channel Name","video
title"]))
589         elif question == '4. Comments in each video':
590             query4 = "select Comments as No_comments ,Title as VideoTitle from
videos where Comments is not null;"
591             cursor.execute(query4)
592             mydb.commit()
593             t4=cursor.fetchall()
594             st.write(pd.DataFrame(t4, columns=["No Of Comments", "Video Title"]))
595
596         elif question == '5. Videos with highest likes':
597             query5 = '''select Title as VideoTitle, Channel_Name as ChannelName,
Likes as LikesCount from videos
598                             where Likes is not null order by Likes desc;'''
599             cursor.execute(query5)
600             mydb.commit()
601             t5 = cursor.fetchall()
602             st.write(pd.DataFrame(t5, columns=["video Title","channel Name","like
count"]))
603
604         elif question == '6. likes of all videos':
605             query6 = '''select Likes as likeCount,Title as VideoTitle from
videos;'''
606             cursor.execute(query6)
607             mydb.commit()
608             t6 = cursor.fetchall()
609             st.write(pd.DataFrame(t6, columns=["like count","video title"]))
610
611         elif question == '7. views of each channel':
612             query7 = "select Channel_Name as ChannelName, Views as Channelviews from
channels;"
613             cursor.execute(query7)
614             mydb.commit()
```

```

615         t7=cursor.fetchall()
616         st.write(pd.DataFrame(t7, columns=["channel name","total views"]))
617
618         elif question == '8. videos published in the year 2022':
619             query8 = '''select Title as Video_Title, Published_Date as VideoRelease,
Channel_Name as ChannelName from videos
620                         where extract(year from Published_Date) = 2022;'''
621             cursor.execute(query8)
622             mydb.commit()
623             t8=cursor.fetchall()
624             st.write(pd.DataFrame(t8,columns=["Name", "Video Publised On",
"ChannelName"]))
625
626             elif question == '9. average duration of all videos in each channel':
627                 query9 = "SELECT Channel_Name as ChannelName, AVG(Duration) AS
average_duration FROM videos GROUP BY Channel_Name;"
628                 cursor.execute(query9)
629                 mydb.commit()
630                 t9=cursor.fetchall()
631                 t9 = pd.DataFrame(t9, columns=['ChannelTitle', 'Average Duration'])
632                 T9=[]
633                 for index, row in t9.iterrows():
634                     channel_title = row['ChannelTitle']
635                     average_duration = row['Average Duration']
636                     average_duration_str = str(average_duration)
637                     T9.append({"Channel Title": channel_title , "Average Duration":
average_duration_str})
638                 st.write(pd.DataFrame(T9))
639
640             elif question == '10. videos with highest number of comments':
641                 query10 = '''select Title as VideoTitle, Channel_Name as ChannelName,
Comments as Comments from videos
642                         where Comments is not null order by Comments desc;'''
643                 cursor.execute(query10)
644                 mydb.commit()
645                 t10=cursor.fetchall()
646                 st.write(pd.DataFrame(t10, columns=['Video Title', 'Channel Name', 'NO
Of Comments']))
647
648
649 class AccountPage:
650     def show(self):
651         st.title("Welcome to :violet[Profile]👤")
652
653         choice = st.selectbox('Login/Signup', ['Login', 'Signup'])
654         if choice == 'Login':
655             email = st.text_input("Email Address")
656             password = st.text_input("Password", type='password')
657
658             st.button('Login')
659
660         else:
661             email = st.text_input("Email Address")
662             password = st.text_input("Password", type="password")
663
664             username = st.text_input("Enter your username")
665             st.button("Create my account")
666
667 class MultiPageApp:
668     def __init__(self):

```

```
669     self.pages = {
670         "Home": HomePage(),
671         "Project": ProjectPage(),
672         "Account": AccountPage()
673     }
674
675
676     def run(self):
677         st.title("Project 1")
678         options = st.sidebar.radio("Main Menu", ["Home", "Project", "Account"])
679         # Display the selected page
680         selected_page = self.pages.get(options, HomePage())
681         selected_page.show()
682
683 if __name__ == "__main__":
684     st.set_page_config(page_title="Streamlit MultiPage Example")
685
686     app = MultiPageApp()
687     app.run()
688
689
690
691
```