Real time Video Recommendation and Analytics

Team Members:

Allen Abraham (aa10770)

Thejaswini Dondemadahalli Manjunath (td2478)

Kuber Mehra (km6133)

Table of Content

ABSTRACT

In today's digital age, online video streaming platforms have become ubiquitous, catering to a diverse audience with a vast array of content. With the exponential growth of video consumption, the need for personalised recommendations and real-time analytics has become paramount. In response to this demand, our project aims to develop a YouTube streaming application that leverages cutting-edge technologies such as Kafka, PySpark, Flask, Docker, JQuery, and MongoDB to provide users with trending video recommendations and insights into their viewing behaviour.

INTRODUCTION

The advent of big data technologies has revolutionised the way we consume and analyse data, opening up new possibilities for real-time insights and personalised experiences. In the realm of online video streaming, platforms like YouTube generate massive amounts of data every second, including user interactions, video views, likes, dislikes, comments, and more. Harnessing this data to deliver tailored recommendations and analyse user behaviour in real-time presents a significant opportunity for enhancing user engagement and satisfaction.

Our project aims to address this challenge by developing a YouTube streaming application that utilises a combination of Kafka, PySpark, Flask, Docker, JQuery, and MongoDB. By integrating these technologies, we can ingest streaming data from Kafka topics, perform real-time analysis using PySpark, store and retrieve data from MongoDB, and present recommendations and insights to users through a user-friendly interface built with Flask and JQuery.

Key components of our application include:

1. Kafka: Serving as the messaging backbone, Kafka allows us to ingest streaming data from various sources, including user interactions and video metadata.
2. PySpark: With PySpark, we can perform real-time data processing and analytics, enabling us to identify trending videos, analyse user behaviour, and extract insights from large datasets.

3. Flask: Flask provides the backend framework for our application, allowing us to create RESTful APIs and serve dynamic web content to users.
4. Docker: Docker containers enable us to deploy and manage our application in a scalable and efficient manner, ensuring consistency across different environments.
5. JQuery: As the frontend framework, JQuery enables us to build a responsive and interactive user interface, providing users with an intuitive browsing experience.
6. MongoDB: MongoDB serves as the database for storing and retrieving user data, video metadata, recommendation logs, and other relevant information.

By integrating these technologies, our YouTube streaming application aims to revolutionise the way users discover and interact with online video content, providing personalised recommendations and real-time insights into their viewing behaviour. Through continuous iteration and improvement, we strive to create a seamless and engaging experience for users, driving engagement and retention on the platform.

LITERATURE SURVEY

The rapid growth of online video streaming platforms has spurred significant research interest in personalised video recommendation systems and real-time analytics. Several studies have explored various techniques and technologies to address the challenges and opportunities in this domain.

Related Work:

1. Real-Time Video Recommendation Systems: Researchers have proposed and implemented real-time recommendation systems that leverage streaming data from online video platforms. Techniques such as collaborative filtering, content-based filtering, and hybrid approaches have been employed to generate personalised recommendations for users in real-time (Jannach et al., 2015; Li et al., 2017).

2. Big Data Analytics for Video Streaming: Big data technologies such as Kafka, Spark, and MongoDB have been widely adopted for real-time analytics and processing of streaming video data. Studies have demonstrated the effectiveness of these technologies in handling large volumes of data and extracting valuable insights to enhance user experiences (Zaharia et al., 2012; Yu et al., 2016).

3. User Behaviour Analysis: Understanding user behaviour and preferences is crucial for delivering relevant video recommendations. Researchers have conducted extensive analyses of user interactions, viewing patterns, and engagement metrics to identify trends and patterns that can inform recommendation algorithms (Abbar et al., 2017; Chen et al., 2018).

Motivation:

The motivation behind our research stems from the increasing demand for personalised video recommendations and real-time analytics in the online streaming industry. With the proliferation of digital content and the rise of on-demand viewing, there is a pressing need for advanced recommendation systems that can adapt to users' preferences and provide relevant suggestions in real-time. By leveraging cutting-edge technologies and innovative approaches, we aim to address this need and enhance the user experience on online video platforms.

Problem Statement:

The primary challenge addressed in this research is the design and implementation of a YouTube streaming application that can deliver personalised video recommendations and real-time insights to users. This involves ingesting streaming data from Kafka topics, performing real-time analytics using PySpark, storing and retrieving data from MongoDB, and presenting recommendations and insights through a user-friendly interface. The key challenges include handling large volumes of streaming data, implementing efficient real-time processing algorithms, and ensuring scalability and reliability of the application.

Objective:

The objective of this research is to develop a robust and scalable YouTube streaming application that utilises state-of-the-art technologies to deliver personalised video recommendations and real-time insights to users. Specifically, our objectives include:

1. Ingesting streaming data from Kafka topics and performing real-time analytics using PySpark.
2. Storing and retrieving data from MongoDB for efficient data management.
3. Implementing recommendation algorithms to generate video recommendations based on user preferences and behaviour.
4. Presenting recommendations and insights through a user-friendly interface built with Flask and JQuery.
5. Evaluating the performance of the application in terms of recommendation accuracy, responsiveness, and user satisfaction.

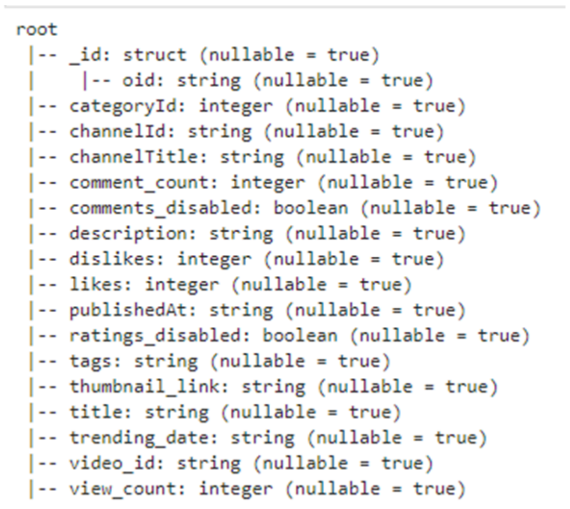
By achieving these objectives, we aim to contribute to the advancement of personalised video recommendation systems and real-time analytics in the online streaming industry.

PROPOSED METHODOLOGY

1. Data Preprocessing

A. Acquire YouTube Dataset

Obtain a dataset containing information about YouTube videos, including channel IDs, video IDs, titles, and other relevant metadata. This dataset could consist of a sample of 2 million videos to ensure comprehensive coverage.



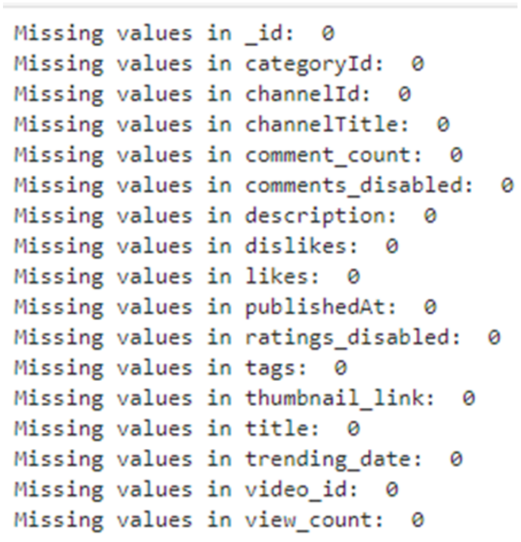
```
root
 |-- _id: struct (nullable = true)
 |    |-- oid: string (nullable = true)
 |-- categoryId: integer (nullable = true)
 |-- channelId: string (nullable = true)
 |-- channelTitle: string (nullable = true)
 |-- comment_count: integer (nullable = true)
 |-- comments_disabled: boolean (nullable = true)
 |-- description: string (nullable = true)
 |-- dislikes: integer (nullable = true)
 |-- likes: integer (nullable = true)
 |-- publishedAt: string (nullable = true)
 |-- ratings_disabled: boolean (nullable = true)
 |-- tags: string (nullable = true)
 |-- thumbnail_link: string (nullable = true)
 |-- title: string (nullable = true)
 |-- trending_date: string (nullable = true)
 |-- video_id: string (nullable = true)
 |-- view_count: integer (nullable = true)
```

Schema of YT dataset

B. Clean and Prepare Data

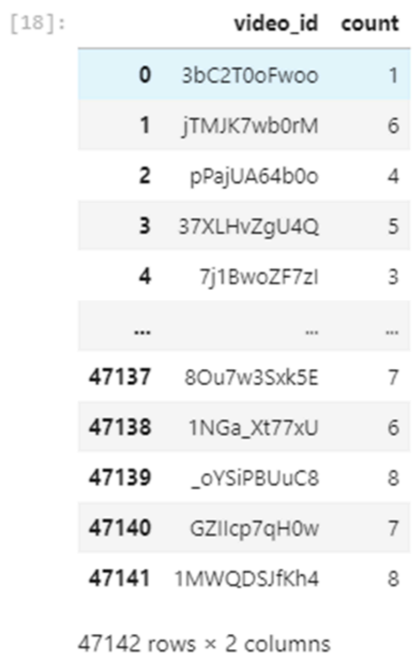
Perform data cleaning and preprocessing tasks such as removing duplicates, handling missing values, and formatting data for further analysis.

a. Checking and removing null values



```
Missing values in _id:  0
Missing values in categoryId:  0
Missing values in channelId:  0
Missing values in channelTitle:  0
Missing values in comment_count:  0
Missing values in comments_disabled:  0
Missing values in description:  0
Missing values in dislikes:  0
Missing values in likes:  0
Missing values in publishedAt:  0
Missing values in ratings_disabled:  0
Missing values in tags:  0
Missing values in thumbnail_link:  0
Missing values in title:  0
Missing values in trending_date:  0
Missing values in video_id:  0
Missing values in view_count:  0
```

b. Counting redundant video_ids:

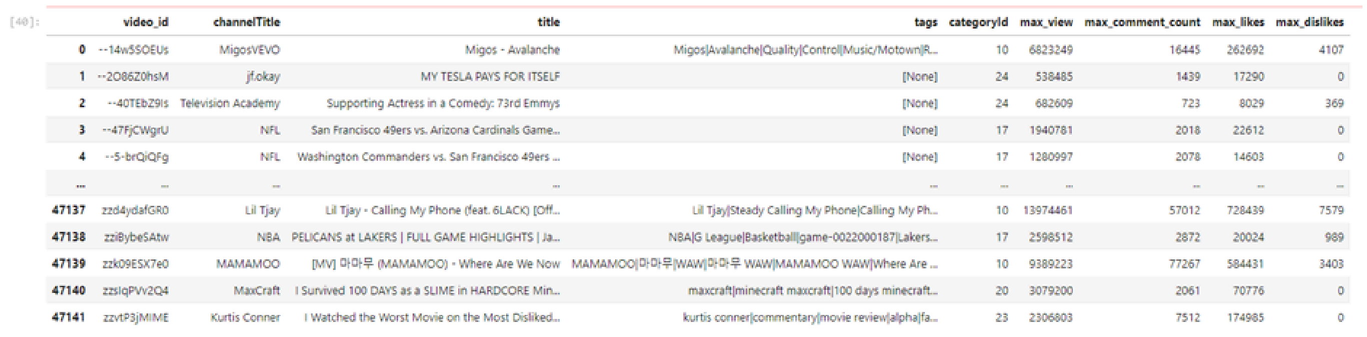


[18]:

	video_id	count
0	3bC2T0oFwoo	1
1	jTMJK7wb0rM	6
2	pPajUA64b0o	4
3	37XLHvZgU4Q	5
4	7j1BwoZF7zI	3
...	...	...
47137	8Ou7w3Sxk5E	7
47138	1NGa_Xt77xU	6
47139	_oYSiPBUuC8	8
47140	GZIlcp7qH0w	7
47141	1MWQDSJfKh4	8

47142 rows × 2 columns

Count of redundant dataset



[40]:

	video_id	channelTitle	title	tags	categoryId	max_view	max_comment_count	max_likes	max_dislikes
0	--14w5SOEUs	MigosVEVO	Migos - Avalanche	Migos|Avalanche|Quality|Control|Music/Motown|R...	10	6823249	16445	262692	4107
1	--2O86Z0hsM	jf.okay	MY TESLA PAYS FOR ITSELF	[None]	24	538485	1439	17290	0
2	--40TEbZ9Is	Television Academy	Supporting Actress in a Comedy: 73rd Emmys	[None]	24	682609	723	8029	369
3	--47FjCWgrU	NFL	San Francisco 49ers vs. Arizona Cardinals Game...	[None]	17	1940781	2018	22612	0
4	--5-brQiQFg	NFL	Washington Commanders vs. San Francisco 49ers ...	[None]	17	1280997	2078	14603	0
...	...	...	...	...	...	...	...	...	...
47137	zzd4ydafGR0	Lil Tjay	Lil Tjay - Calling My Phone (feat. 6LACK) [Off...	Lil Tjay|Steady Calling My Phone|Calling My Ph...	10	13974461	57012	728439	7579
47138	zziBybeSAtw	NBA	PELICANS at LAKERS | FULL GAME HIGHLIGHTS | Ja...	NBA|G League|Basketball|game-0022000187|Lakers...	17	2598512	2872	20024	989
47139	zzk09ESX7e0	MAMAMOO	[MV] 마마무 (MAMAMOO) - Where Are We Now	MAMAMOO|마마무|WAW|마마무 WAW|MAMAMOO WAW|Where Are ...	10	9389223	77267	584431	3403
47140	zzsIqPVv2Q4	MaxCraft	I Survived 100 DAYS as a SLIME in HARDCORE Min...	maxcraft|minecraft maxcraft|100 days minecraft...	20	3079200	2061	70776	0
47141	zzvtP3jMIME	Kurtis Conner	I Watched the Worst Movie on the Most Disliked...	kurtis conner|commentary|movie review|alpha|fa...	23	2306803	7512	174985	0

Final dataset

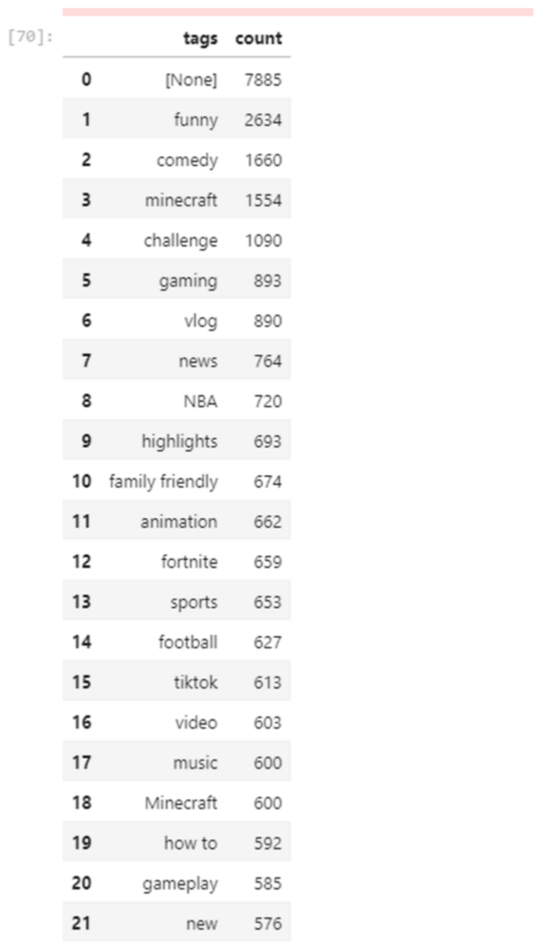
C. Create Tables

Organise the dataset into tables, including a "Videos" table containing video information, a "Channels" table for channel details, and additional tables as needed for tags, categories, etc.

2. Analysis

- Choosing max values for like, dislike, view_count, comment_count as this dataset contains week wise video depending on the trending chart, hence the latest values always had more views, like and dislike than the previous week
- Choosing columns for potential recommendations:
  "Tags" as well as "categoryId" were the two potential columns which were deemed to be considered to use for potential values that can be used to give further recommendations on the basis of user watch history, likes and dislikes. Hence we furthered looked into the distribution of both of these columns

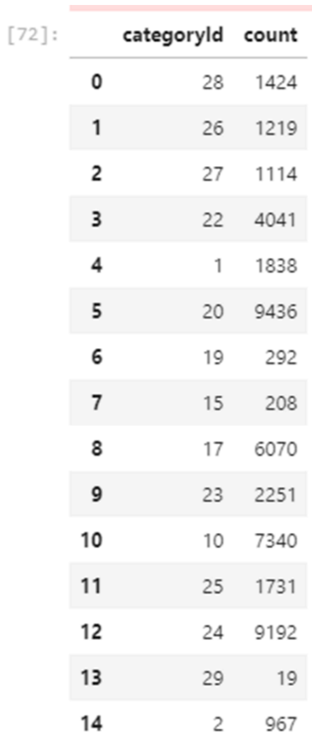
  While tags gives a lot of information about videos and what topic a person might like, the distribution of the videos in those tags were skewed hence we decided to drop the tags being a condition



[70]:

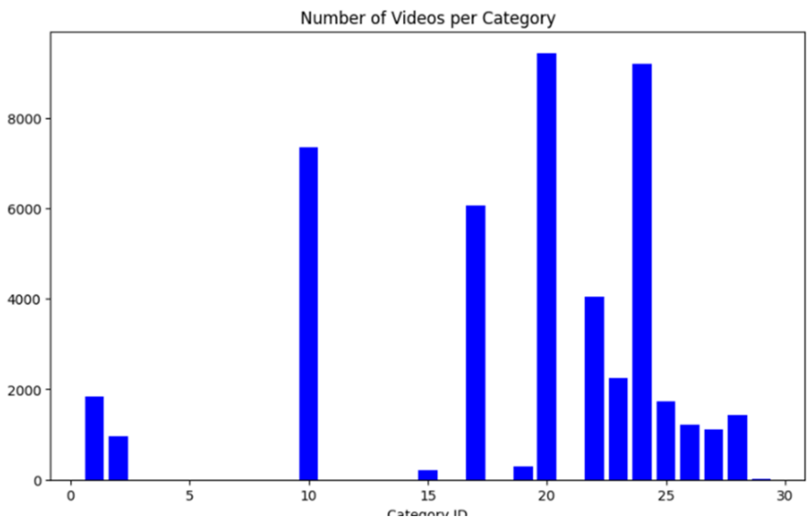
	tags	count
0	[None]	7885
1	funny	2634
2	comedy	1660
3	minecraft	1554
4	challenge	1090
5	gaming	893
6	vlog	890
7	news	764
8	NBA	720
9	highlights	693
10	family friendly	674
11	animation	662
12	fortnite	659
13	sports	653
14	football	627
15	tiktok	613
16	video	603
17	music	600
18	Minecraft	600
19	how to	592
20	gameplay	585
21	new	576

On the other hand categoryId seems to give better results than tags.



[72]:

	categoryId	count
0	28	1424
1	26	1219
2	27	1114
3	22	4041
4	1	1838
5	20	9436
6	19	292
7	15	208
8	17	6070
9	23	2251
10	10	7340
11	25	1731
12	24	9192
13	29	19
14	2	967

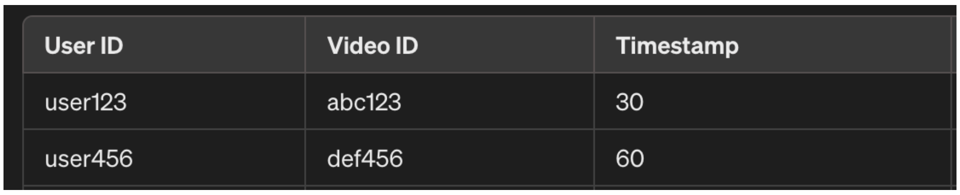


3. Data Collection

- User Interaction Logging: Track user interactions with the YouTube streaming application, including timestamps when users skip to specific segments of videos, as well as videos liked and disliked.
- Log Timestamps: Store logged timestamps in a database, associating each timestamp with the corresponding video ID and user ID.
- Log Liked and Disliked Videos: Record users' interactions with videos by logging liked and disliked videos, associating each interaction with the respective video and user.

Example Dataset Structure:

User Interaction Log Table



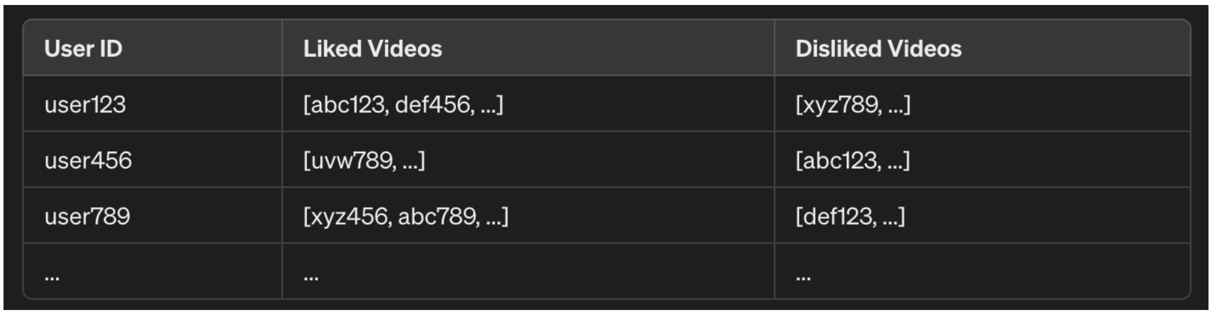
User ID	Video ID	Timestamp
user123	abc123	30
user456	def456	60

Liked and Disliked Videos Arrays:

For each user, maintain arrays of video IDs representing liked and disliked videos. Store these arrays in the user profile or interaction log, allowing for easy access and analysis of user preferences.

Example Dataset Structure:

User Profile Table:



User ID	Liked Videos	Disliked Videos
user123	[abc123, def456, ...]	[xyz789, ...]
user456	[uvw789, ...]	[abc123, ...]
user789	[xyz456, abc789, ...]	[def123, ...]
...	...	...

Liked Videos Array: Contains video IDs that the user has liked.

Disliked Videos Array: Contains video IDs that the user has disliked.

4. Computing Most Watched Timeframe

Aggregate Timestamps: Aggregate the logged timestamps for each video to determine the timeframe that is most watched by users.

Compute Most Watched Timeframe: Analyse the distribution of timestamps and identify the time interval with the highest concentration of user interactions.

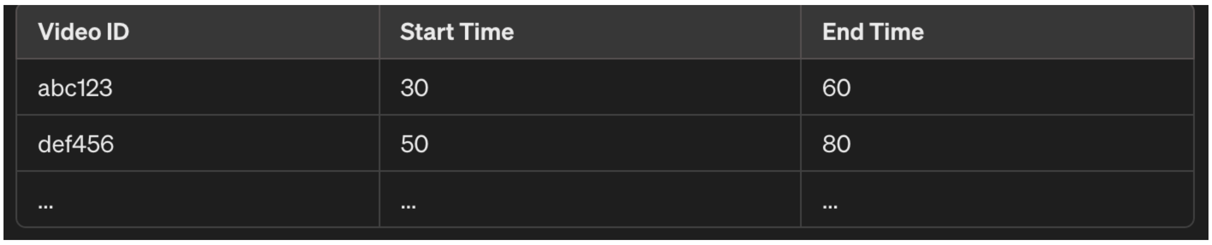
Example Logic:

For each video, calculate the frequency of timestamps and identify the time interval with the highest frequency.

Determine the start and end timestamps of the most watched timeframe based on the identified interval.

Example Output:

Most Watched Timeframe Table:

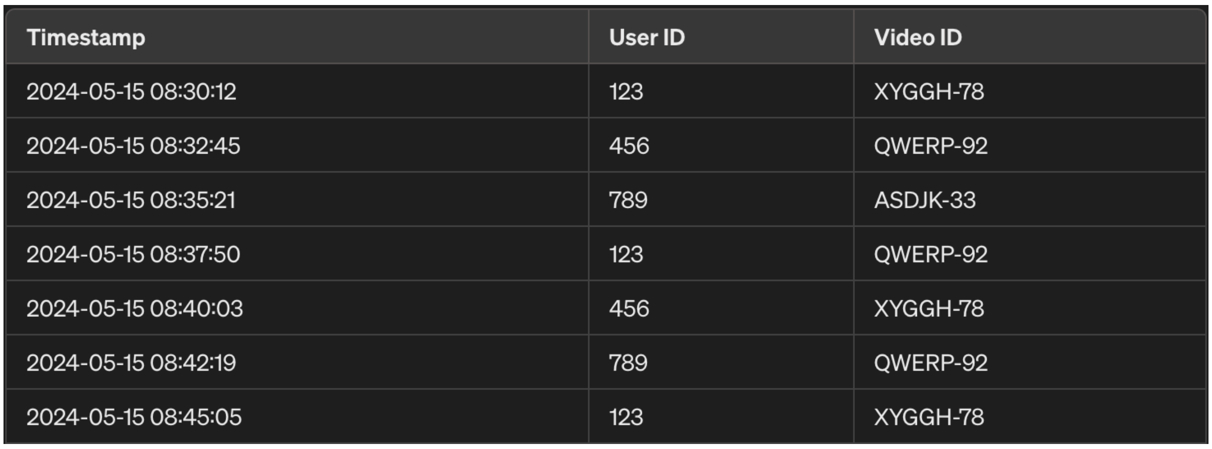


Video ID	Start Time	End Time
abc123	30	60
def456	50	80
...	...	...

5. Computing Trending videos list

- User interactions, such as video clicks, are recorded and sent to a Kafka topic named "videoRecommender" with five partitions.
- Spark consumes the data stream from the "VID_OPEN_TOPIC" topic to perform real-time analysis.
- Spark utilises a basic methodology involving grouping and counting to analyse user interactions with videos.
- By aggregating user interactions and counting the occurrences of each video, Spark identifies the top 20 trending videos.
- This process enables the platform to dynamically update the list of trending videos based on user engagement, ensuring relevant content is prominently featured.

The VID_OPEN_TOPIC kafka topic log data:



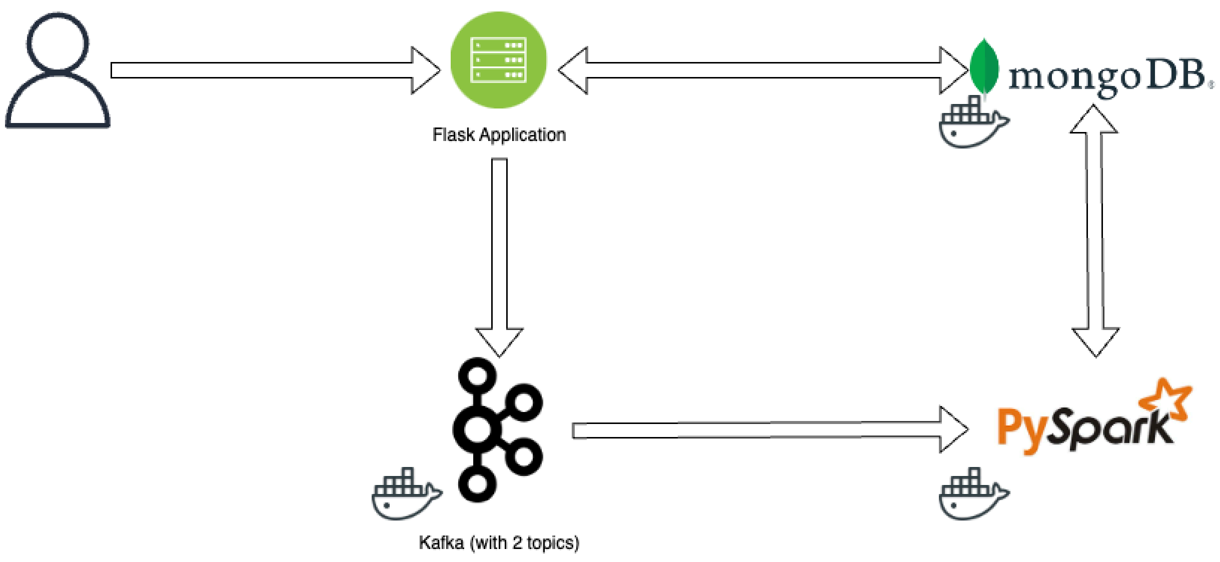
Timestamp	User ID	Video ID
2024-05-15 08:30:12	123	XYGGH-78
2024-05-15 08:32:45	456	QWERP-92
2024-05-15 08:35:21	789	ASDJK-33
2024-05-15 08:37:50	123	QWERP-92
2024-05-15 08:40:03	456	XYGGH-78
2024-05-15 08:42:19	789	QWERP-92
2024-05-15 08:45:05	123	XYGGH-78

6. Implementation:

- Develop Scripts: Write scripts using PySpark for data preprocessing and analysis.

- Set up Database: Configure MongoDB or another suitable database for storing video metadata, user interaction logs, and liked/disliked videos.
- Create Flask API: Build a Flask API to handle user interactions and retrieve recommendations, most watched timeframes, and liked/disliked videos.
- Frontend Development: Develop a user-friendly frontend interface using JQuery to display video recommendations, allow users to interact with the application, and view their liked/disliked videos.

ARCHITECTURE



The client/end-user interacts with a Flask web application. Flask helps us serve the frontend interface where users can submit data or requests and also handle the backend APIs which handle the requests from the client.

The Flask application has APIs which help with analytics. The data required for analytics are pushed into Kafka topics. Our Kafka system is configured with 2 Topics, which handle the following features:

1. Analysing most trending videos across the platform
2. Finding most viewed section of a video

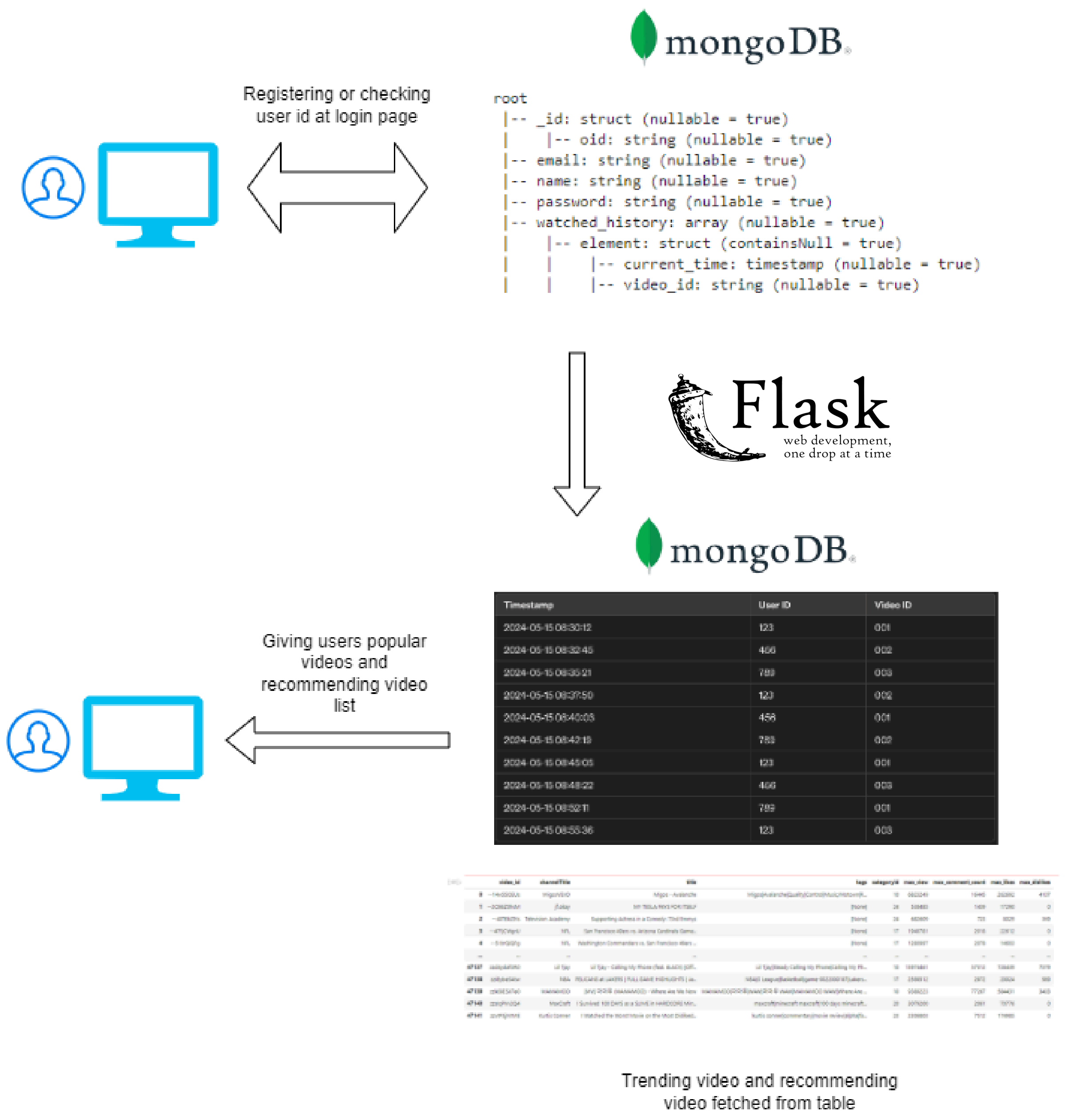
Kafka helps us to handle high-throughput data feeds and to manage the flow of data through topics

Data from Kafka is streamed to PySpark for complex processing, analysis, or transformation. PySpark allows for handling big data through its distributed processing capability. In our project, we use it to find the most viewed videos across the platform in the past 7 days (i.e. trending videos) and to compute the most viewed section of a video.
PySpark is also connected to MongoDb and can process and analyse data from the DB.

The results from the PySpark computation are stored in MongoDB. MongoDb is a NoSQL database, which was chosen for its high performance, high availability, and easy scalability. We store information on the video like video_id, like and dislike count, metadata, etc in addition to the results from PySpark.

Pyspark, Kafka and MongoDb are run in Docker containers to help ease development efforts and generate reproducible builds. The entire architecture is written out in a Docker compose file which helps to easily setup the entire architecture.

DATAFLOW

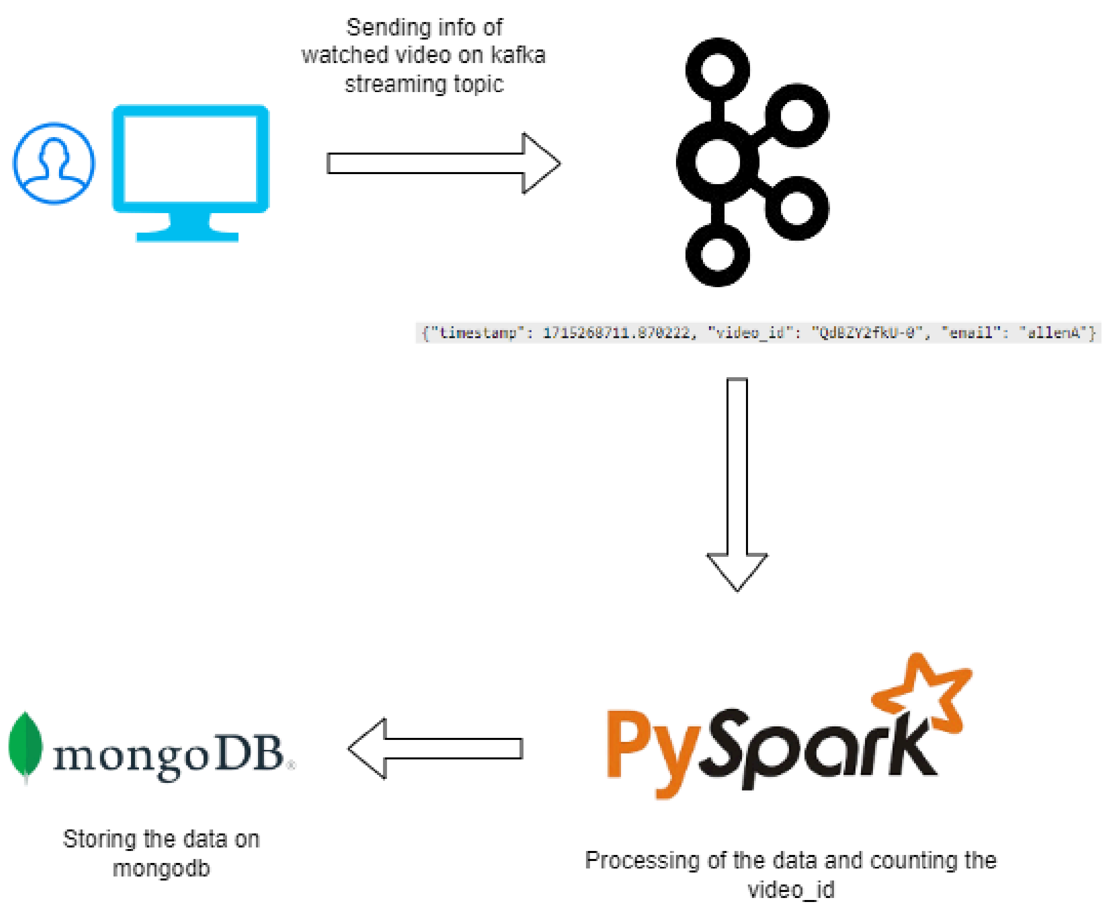

mongoDB
Registering or checking user id at login page
Flask web development, one drop at a time
mongoDB
Giving users popular videos and recommending video list
Trending video and recommending video fetched from table

```
root
 |-- _id: struct (nullable = true)
 |    |-- oid: string (nullable = true)
 |-- email: string (nullable = true)
 |-- name: string (nullable = true)
 |-- password: string (nullable = true)
 |-- watched_history: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- current_time: timestamp (nullable = true)
 |    |    |-- video_id: string (nullable = true)
```

Timestamp	User ID	Video ID
2024-05-15 08:30:12	123	001
2024-05-15 08:32:45	456	002
2024-05-15 08:35:21	789	003
2024-05-15 08:37:50	123	002
2024-05-15 08:40:03	456	001
2024-05-15 08:42:18	789	002
2024-05-15 08:45:05	123	001
2024-05-15 08:48:22	456	003
2024-05-15 08:52:11	789	001
2024-05-15 08:55:36	123	003

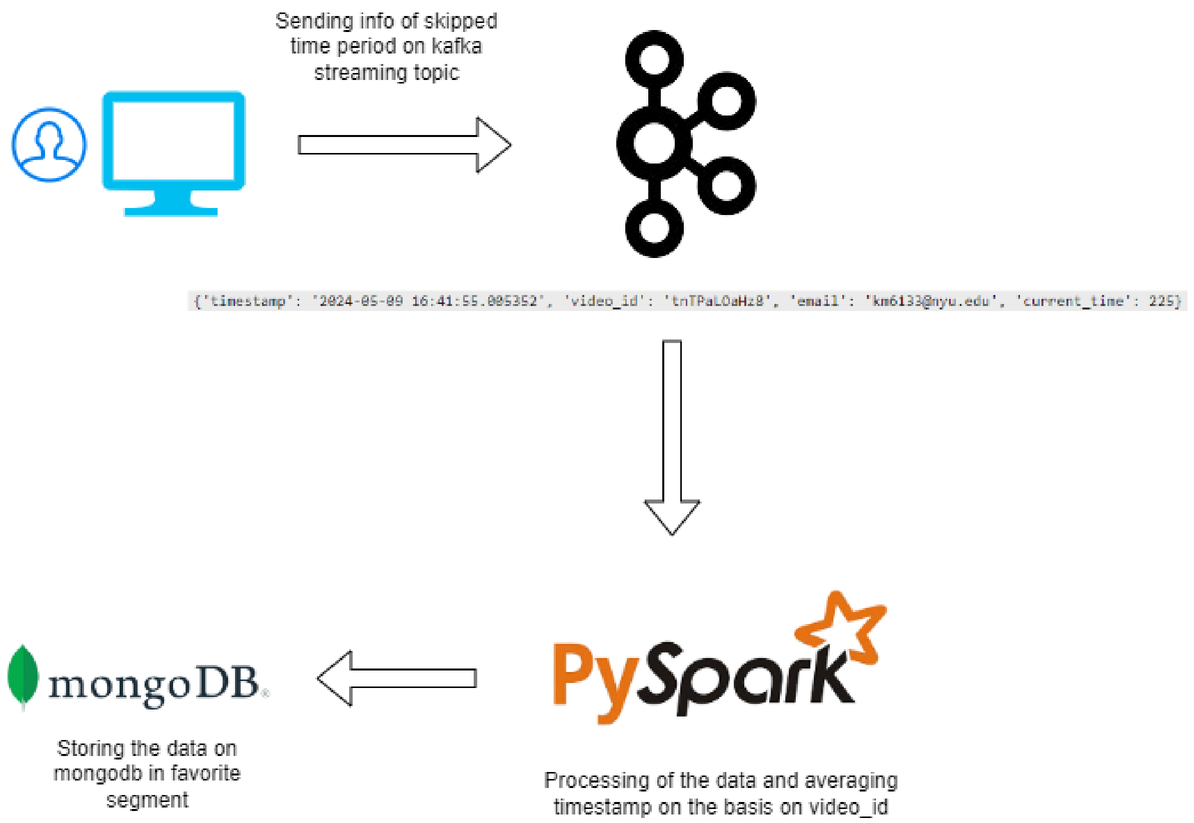
	video_id	channelTitle	title	tags	categoryId	max_view	max_comment_count	max_likes	max_dislikes
0	~14v2DOU...	MigosVEVO	Migos - Avalanche	MigosAvalanche|QualityControlMusic|MotownR...	10	[illegible]	16446	252602	4107
1	~0QR6Z0hAM	JTalkay	MY TESLA PAYS FOR ITSELF	[None]	28	539880	1409	17290	0
2	~4ZTEB2Tls	Television Academy	Supporting Actress in a Comedy: 73rd Emmys	[None]	24	682609	723	6029	309
3	~4TSCWgrU	NFL	San Francisco 49ers vs. Arizona Cardinals Game...	[None]	17	1048751	2018	22612	0
4	~5-bnQGQg	NFL	Washington Commanders vs. San Francisco 49ers ...	[None]	17	1230997	2078	14800	0
...	...	...	...	...	...	...	...	...	...
47137	[illegible]	Lil Tjay	Lil Tjay - Calling My Phone (feat. 6LACK) [Off...	Lil Tjay|Steady Calling My Phone|Calling My Ph...	10	[illegible]	37513	536639	7379
47138	[illegible]	NBA	PELICANS at LAKERS | FULL GAME HIGHLIGHTS | ...	NBA|G League|Basketball|game-0022000187|Lakers...	17	2358812	2973	20924	909
47139	[illegible]	MAMAMOO	[MV] 마마무 (MAMAMOO) - Where Are We Now	[illegible]	10	[illegible]	7737	584431	3403
47140	[illegible]	[illegible]	I Survived 100 DAYS as a SLIME in HARDCORE Min...	minecraft|minecraft minecraft|100 days minecraft...	20	[illegible]	2561	75776	0
47141	[illegible]	Kurtis Conner	I Watched the Worst Movie on the Most Disliked...	kurtis conner|commentary|movie review|[illegible]...	23	[illegible]	7512	119880	0

Dataflow at login and registration page


Sending info of watched video on kafka streaming topic
Storing the data on mongodb
Processing of the data and counting the video_id

```
{"timestamp": 1715268711.870222, "video_id": "QdBZY2fkU-0", "email": "allenA"}
```

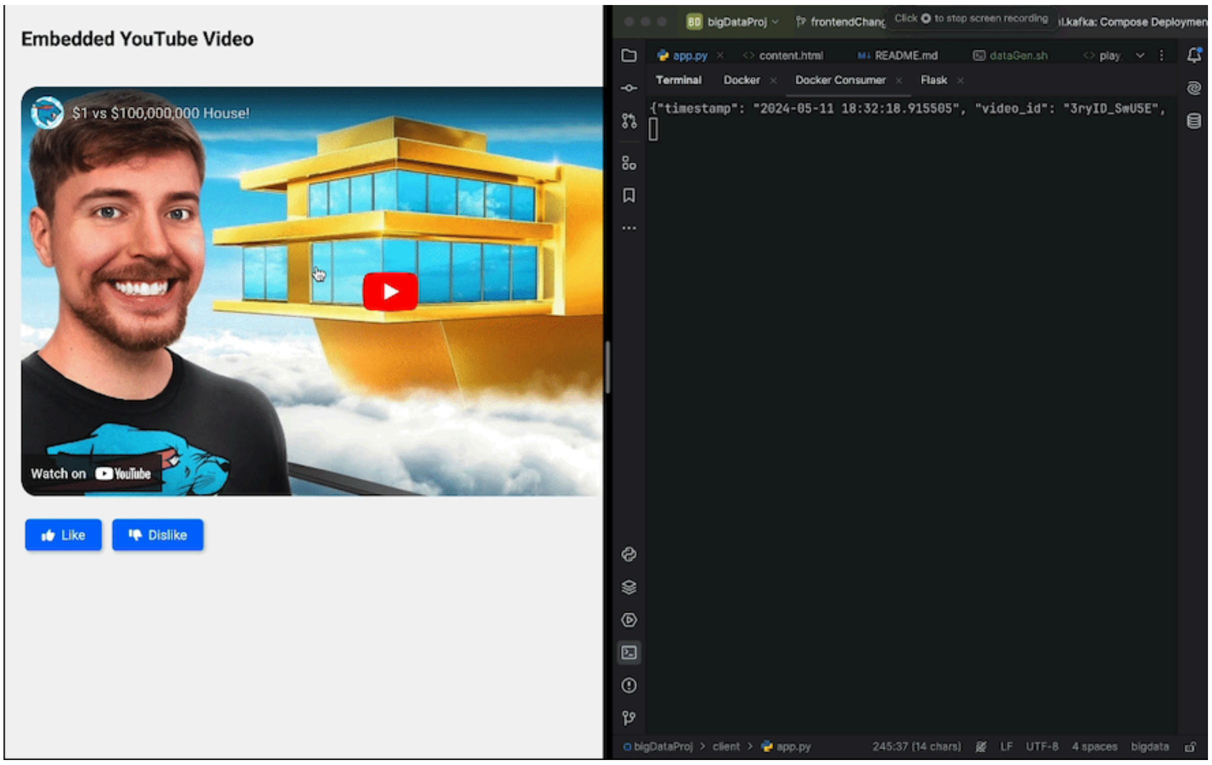
Data flow at content page


Sending info of skipped time period on kafka streaming topic
Storing the data on mongodb in favorite segment
Processing of the data and averaging timestamp on the basis on video_id

```
{'timestamp': '2024-05-09 16:41:55.005352', 'video_id': 'tnTPaLOaHz8', 'email': 'km6133@nyu.edu', 'current_time': 225}
```

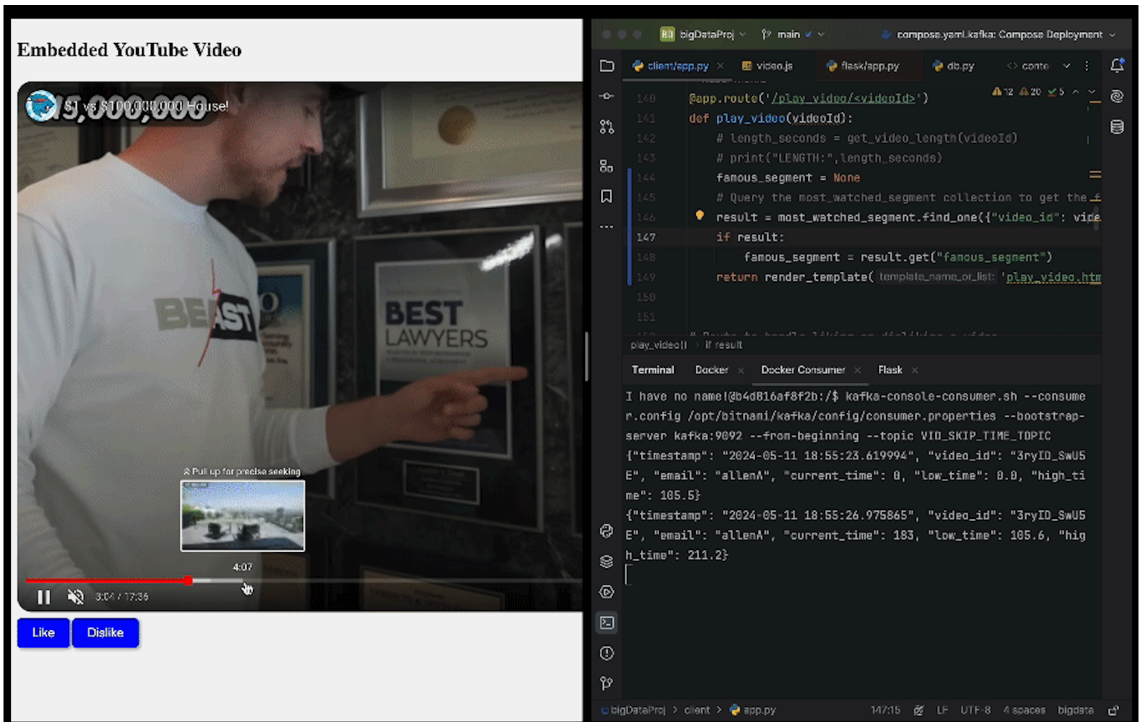
Data flow at play watch page

WORKFLOW



```
{"timestamp": "2024-05-11 18:32:18.915505", "video_id": "3ryID_SwUSE",
```

Logs being generated to Kafka when a user clicks on a video



```
@app.route('/play_video/<videoId>')
def play_video(videoId):
    # length_seconds = get_video_length(videoId)
    # print("LENGTH:",length_seconds)
    famous_segment = None
    # Query the most_watched_segment collection to get the f
    result = most_watched_segment.find_one({"video_id": vide
    if result:
        famous_segment = result.get("famous_segment")
    return render_template( template_name_or_list: 'play_video.htm
```

```
I have no name!@b4d816af8f2b:/$ kafka-console-consumer.sh --consume
r.config /opt/bitnami/kafka/config/consumer.properties --bootstrap-
server kafka:9092 --from-beginning --topic VID_SKIP_TIME_TOPIC
{"timestamp": "2024-05-11 18:55:23.619994", "video_id": "3ryID_SwU5
E", "email": "allenA", "current_time": 0, "low_time": 0.0, "high_ti
me": 105.5}
{"timestamp": "2024-05-11 18:55:26.975865", "video_id": "3ryID_SwU5
E", "email": "allenA", "current_time": 183, "low_time": 105.6, "hig
h_time": 211.2}
```

Logs being generated to Kafka when a user seeks a video. The log contains data about user current seek time and total video duration

Results

In our latest report, we highlight how our application successfully caters to both new and existing users with tailored video recommendations. For new users, the application immediately showcases a selection of trending videos, offering a snapshot of current popular content to engage them from the outset. As users interact more with the platform, the recommendation engine smartly adjusts by incorporating a mix of these trending videos and personalised selections based on the user's previously liked categories.
This seamless integration ensures that existing users receive a finely-tuned mix that not only includes what is currently popular but also aligns with their specific interests. The underlying technology that supports these features is a robust and scalable database architecture. This database is designed to efficiently handle large datasets, allowing for rapid querying and updates, which is critical as the volume of users and interactions grows.
Our system's ability to adapt to increasing data demands without performance degradation is a testament to its design and implementation. This ensures that all users experience fast, responsive interactions with the platform, fostering higher engagement and satisfaction. The combination of cutting-edge technology and user-focused recommendations continues to set our platform apart in a competitive digital landscape.

CONCLUSION AND FUTURE WORK

In conclusion, the development of our YouTube streaming application leveraging Kafka, PySpark, Flask, MongoDB, Docker, and JQuery presents a promising solution for real-time video recommendation and user interaction analysis. Through extensive data preprocessing, including the ingestion and processing of YouTube's vast dataset, we have laid the foundation for accurate and personalised video recommendations. The integration of Kafka facilitates seamless data streaming, enabling real-time analysis of user interactions and video trends. PySpark's robust processing capabilities empower us to compute trending videos and identify the most watched time frames efficiently. With Flask and MongoDB, we can store and retrieve user interaction data, providing valuable insights into user preferences and behaviours. Additionally, Docker ensures easy deployment and scalability, while JQuery offers a responsive and intuitive user interface for enhanced user experience.

Future Work

Moving forward, some avenues for future work can be explored to further enhance the functionality and effectiveness of our YouTube streaming application:

Enhanced Recommendation Algorithms: Develop and integrate advanced recommendation algorithms, such as collaborative filtering and content-based filtering, to improve the accuracy and relevance of video recommendations.

User Engagement Analytics: Expand the scope of user interaction analytics to include additional metrics such as session duration, click-through rates, and engagement patterns, enabling deeper insights into user behaviour.

Integration with External APIs: Integrate with external APIs, such as YouTube Data API, to access real-time video metadata and enrich the recommendation process with up-to-date information.

REFERENCES

[1] Jannach, D., Ludewig, M., & Lerche, L. (2015). A Survey of Real-time Recommendation Systems. ACM Computing Surveys (CSUR), 47(2), 1-47.

[2] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2012). Spark: Cluster Computing with Working Sets. In Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud'12).

[3] Yu, H., Wang, J., & Zhang, L. (2016). A Real-time Streaming Analytics System Based on Spark Streaming and Kafka. In Proceedings of the International Conference on Computer, Network, and Communication Engineering (ICCANCE) (pp. 187-192). ACM.