UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Platform for generating personalized digests from Youtube content

Author:
Victoria USACHOVA

Supervisor:
Dmytro PRYIMAK

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2022

Declaration of Authorship

I, Victoria USACHOVA, declare that this thesis titled, "Platform for generating personalized digests from Youtube content" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Platform for generating personalized digests from Youtube content

by Victoria USACHOVA

Abstract

Nowadays, people consume information from various sources, making it harder to systematize and prioritize news notifications. This statement is relevant to one of the biggest media platforms - Youtube. Despite having a notification system for subscriptions, YouTube doesn't have a convenient digest system that could be stored somewhere and checked whenever the user wants to. The main platform for delivering digest was chosen to be the Telegram messenger. This work describes the development process and technologies overview of the personalized digest system from youtube content. Demo version of the application is available here. The code of this application could be find here.

Contents

List of Figures

List of Abbreviations

API	**A**pplication **P**rogramming **I**nterface
SRS	**S**oftware **R**equirement **S**pecification
CS	**C**loud **S**torage
QA	**Q**uality **A**ttributes
UC	**U**se **C**ase

Chapter 1

Introduction

1.1 Motivation

Nowadays we consume a variety of information from internet media platforms on different topics such as news, entertainment, and science. And one of the biggest and most popular informational resources is Youtube where billions of people chose to spend time. Although Youtube has a notification system for new videos published from the subscription list it has no compact everyday digest. For now, there is a list of all new videos, streams, shorts, and posts from channels based on their publish time which is not a convenient and usable feature. Instead, users can receive an informative everyday digest with the most important information based on their preferences which will save time to choose appropriate content and don't miss videos from their subscriptions. Moreover, in this case, users can check their everyday digest anytime through the day, and even after, while default Youtube notifications come ones and are mostly ignored or forgotten. Such an approach helps to consume content smartly and don't miss any important information.

1.2 Choosing messenger as a platform

Telegram messenger is an application that users visit on a daily basis which makes it relevant to send regular digests in it. Also, this is the most lightweight solution that doesn't require any downloads or remembering external sites. Another advantage of using Telegram is a secure, encrypted connection and ease of integrating Youtube API functionality along with its own API. Furthermore, the application is available on all major platforms and even has its web version for a browser.

1.3 Goals

- Determine functional and non-functional requirements of the system
- Develop efficient system architecture which satisfies given requirements
- Chose appropriate technologies for the system
- Integrate external APIs into the solution
- Develop a platform with personalized digest

Chapter 2

Existing Solutions

2.1 YouTube DigestGram

YouTube DigestGram is an active telegram bot that sends notifications after new videos are uploaded to the listed channels. So the user must enter one by one link for each preferred channel which is a big disadvantage of this solution. Also, there are available commands for removing particular channels from the feed or stopping sending digests. Digest looks like a short notification, which comes the next day after the video is published, about a new video and does not contain any information about comments, likes, dislikes, etc. As proposed in this work solution offers users to login into their accounts, so there is no need to manage subscriptions in Telegram. The significant advantage is the ability to change a scheduled time and get a short overview for each video. Besides DigestGram sends each video in a separate message which is quite distractive and could be improved by combining all the information in one message for each day.

2.2 Built-in Youtube solution

Built-in Youtube solution for new videos has a few major flows. The scheduled digest which is available in the notification tab of the app is a temporary notification of all the Youtube notifications. It combines many different types of notifications so it could include too much information that isn't prioritized somehow. Also, this digest isn't stored anywhere and expires soon. With the subscriptions tab, there is a similar problem: it combines all push notifications including live streams, shorts, posts, and comments notifications. Moreover, the list of new videos is sorted simply by creation time and doesn't have any additional information with them such as the number of likes, number in trends, top comments, and so on. Both native solutions don't store digests or new videos list for each day and the user hasn't the possibility to look back. However, one of the pros of it is the possibility to schedule its notifications time to a user-defined time.

2.3 Pipedream

Pipedream is a low code integration platform for developers that allows you to connect APIs fast. It has a great number of APIs it could make connections between APIs for Telegram Bot and YouTube Data API for example. There are trigger functions for each of the external platforms. There is also a trigger workflow on new videos from subscribed channels and then a list update function with Telegram Bot could be performed. This workflow allows anyone to receive notifications about new videos right in a telegram bot. But the main and obvious cons of this platform are that it is for developers mostly and requires a lot of configurations, authentication, and limitations to be set up manually. Also, no personalization is available.

Chapter 3

Functional requirements and quality attributes

3.1 Functional requirements

Before designing and implementing a solution it is necessary to define functional and non-functional requirements to have a clear vision of systems specifications and have priorities. Both of them should be clear and understandable.

Nonfunctional requirements are known as quality attributes; they describe common system characteristics, such as security or availability. Functional requirements describe the behavior of the software product in certain situations. Functional requirements can be presented as a text, a diagram, or in any of the following convenient forms:

- Software requirements specification document (SRS)
- Use cases
- User stories
- Functional decomposition
- Software prototypes

The UC diagram itself shows which functionality and opportunities the system provides and which persons execute them. However, there is no details about individual use cases except their names. This problem could be solved by the Use Case Specification. The use case specification includes the individual use cases, defining several details for each of them. First of all, we're going to describe the details, and then we'll create a part of the specification of our Use Case model.

Let's further consider the second form of functional requirements, use case text specification for full functionality details, and use case diagrams as demonstrative and graphical alternatives.

1. This use case provides a login procedure into the system for a new user.

   The main actors are the system and the user.

   Basic flow: after joining the chat it suggests the user authenticate via an existing Google account. After authentication and giving read-only access to a subscriptions list service stores a new user along with his/her token in the system.

   Alternative flow: user fails to log in or doesn't give access to the app. Then the corresponding message about the necessity of giving a permit pops up in a chat.

2. The second use case is scheduled; the system, triggered by a batch-oriented workflow, sends an everyday digest at a fixed time into telegram chat so the user can actually view this information.

   Basic flow: user can view a digest with full information considering previous activities and clickable video links after a time trigger sends it at a fixed time. Digest contains top-viewed videos for the last day with the title, channel title, number of likes, number of views, top comment, and number in trends.

   Alternative flow: there are no new videos from a subscription then the user receives information about trending videos at that time.

3. The third use case allows users to change schedule time.

   Basic flow: user can change from evening to morning time and vice versa.

   Alternative flow: the user isn't logged in; then the user receives a notification about it and instructions to log in first.

4. The fourth use case logs out the user from the system.

   Basic flow: user can log out from the system then no digests would be sent and the user becomes in-active in the system until the new login.

5. The fifth use case shows current trends for the user.

   Basic flow: user asks for a currently trending video and the system sends top-3 videos in trends for the user's country. The message contains the same information about videos as scheduled digests.

Below depicted visualization of use cases:

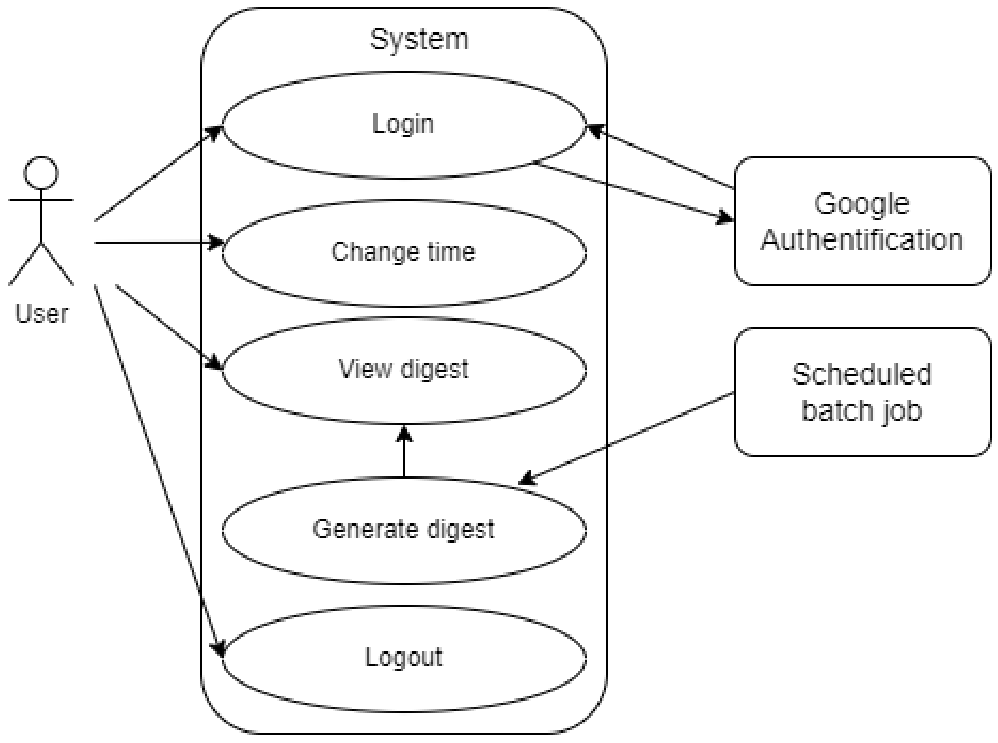


FIGURE 3.1: Use Case Diagram of the System

3.2 Non-functional requirements

Non-functional requirements, also known as quality attributes, describe the characteristics that a system should have. The most known non-functional requirements are:

- Usability
- Security
- Maintainability
- Performance
- Availability
- Scaleability

When developing the system, the main attributes were selected to be scalability and availability. Scalability defines a software system's capability to handle growth in some dimension of its operations. One of the main dimensions:

- The number of simultaneous user or external requests a system can process
- The amount of data a system can effectively process and manage.
- The value that can be derived from the data a system stores through predictive analytics

The last two should be especially taken into account. Because the system daily receives information about videos, channels, and user preferences which could be used in the future for large-scale analysis to improve recommendations. Moreover, with an increase in user counts data, and workload regarding videos and channels volume increases rapidly. These factors indicate that scalability should be one of the main focuses of the system.

Another important non-functional requirement for the system is availability. Availability and scalability are in general highly compatible partners. As we scale our systems through replicating resources, we create multiple instances of services that can be used to handle requests from any users. If one of our instances fails, the others remain available. It could be maintained manually by running extra instances or with the help of serverless solutions.

Chapter 4

External APIs

4.1 YouTube Data API and OAuth 2.0

With the YouTube Data API, it is possible to add a variety of YouTube features to the application. Using the API we can get information about recently published videos by a particular channel, user subscriptions, currently trending videos, and more details about videos like tags, comments, and likes. There are two main types of resources that we can retrieve using the API, such as activities and subscriptions; They can require different levels of authorization. Subscriptions of the user are sensitive data so authorization via OAuth 2.0 protocol is performed in the application. For this purpose authorization with a Google account is required.

An **activity** resource request aims for information about an action that a specified channel, or user, has taken on the YouTube platform recently. For our purposes, this type of request is used to get videos uploaded by a given channel for the last day. So in a request, channelId, part, and publishedAfter parameters are specified to get only recent videos for a given channel.

A **subscription** resource contains information about a YouTube user subscription. Requests of this resource type require a higher level of authentication. These requests are used to get subs of the user by his unique identifier. For that tokens received via OAuth 2.0 protocol are used.

Let's take a closer look at the requested parameters and response to the activities query. The part parameter should contain all the information we want to get in response to recent channel activity. In our case, it is content details and a snippet of the published video with all the related information, such as publishing time, title, etc. Also, channelId is a required field to identify the channel. The other parameter which is not required but is better to use is maxResults because otherwise, it becomes easy to exceed the Youtube quota for requests.

Youtube API uses JSON format to respond so we can obtain all the needed information. There are available publish times, channel and video ids, titles, descriptions, and other details. Furthermore, we can use video resource methods to get more information regarding the video, such as comments, likes, and numbers in trends.

To authorize a request tokens are stored for each user. In this case, there is no need for re-authorization for the user. For every Youtube Data API request, they are sent.

The YouTube Data API uses the OAuth 2.0 protocol for authorizing access to private user data. It is necessary to get a user's data with read-only scope. So during authorization users should agree to share their subs in read-only mode. Figure 4.3 shows the main flow of the authentification process, and the list below explains some core OAuth 2.0 concepts:

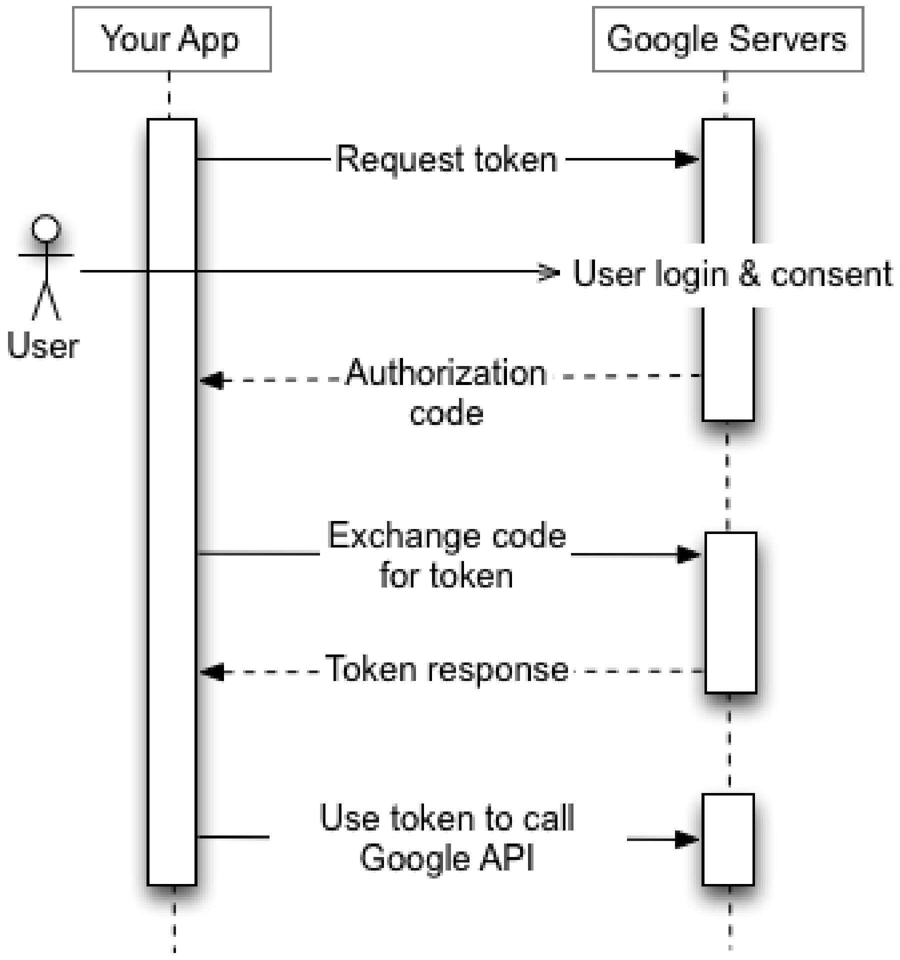


FIGURE 4.1: OAuth 2.0 Token exchange

- During the first user's attempts to make use of functionality in the application that requires the user to be logged in to a Google Account or YouTube account. For that special authorization link is created, using client secret of the Google Cloud Project. After the user clicks the link and login into the account application initiates the OAuth 2.0 authorization process.
- Application directs the user to Google's authorization server. The link to that page specifies the scope of access that the application is requesting for the user's account. For our needs, it is read-only scope. The scope specifies the resources that the platform can retrieve, when acting as the authenticated user.
- If the user agrees to authorize an application to access those scope of resources and fills in the login form, Google returns an access token and refresh token to the application. With the refresh token, it becomes possible to update credentials after the expiration of the access token.

4.2 Telegram API

A Telegram bot is a valuable addition to smart communication, especially in professional areas. Compared to the most popular messaging services, Telegram offers additional advantages in exchanging information with the special queries and actions of a bot. Telegram was chosen to be the main platform for delivering digest, and there are the main advantages of using Telegram in general:

- Free. It is a free messenger not depending on the number of messages, the creation of a bot is also free of charge. So it is a great opportunity to use active users of Telegram.
- Secure. Telegram is considered to be quite secure majorly for the reason messages are sent across the platform in encrypted form.
- Available. The telegram messenger is available on all most popular platforms such as Android, iOS, Windows phones with desktop apps for Mac, Linux, and Windows. Moreover, it also has a web version
- User-friendly interface. It provides a user-friendly interface that simplifies the process of creating and managing chatbots.
- Video preview. It can analyze Youtube video links and add image previews for it.

Creating a Telegram Bot at first it is needed to receive an authentification token from a BotFather. Further development is possible via pure Telegram API or lightweight Python wrappers for it. Telegram API and its ext. module for Bots are used in the application because it satisfies all current needs such as sending responses, scheduling messages, or broadcasting.

Chapter 5

Solution overview

5.1 Architecture

According to the functional requirements of the system, we can highlight the main components of the system on a high level:

- Python server which performs the main logic of the system such as:
  1. register users in the system and store their OAuth tokens for future queries
  2. makes it possible to change the scheduled time for the digest
  3. send current trending videos by a request
  4. performs logout from the system and makes the user in-active
- Scheduled Python tasks to update users' subscriptions and create daily digests.
- Cloud Storage as storage of ready-to-send digests.
- User Data Storage is the main database storing users' data such as their Youtube id, chat id, country, subscriptions, etc.
- Telegram Server, which is a proxy between users' queries and our application.

All this functionality is split up into a couple of different services depicted on a detailed architecture (see figure 5.2).

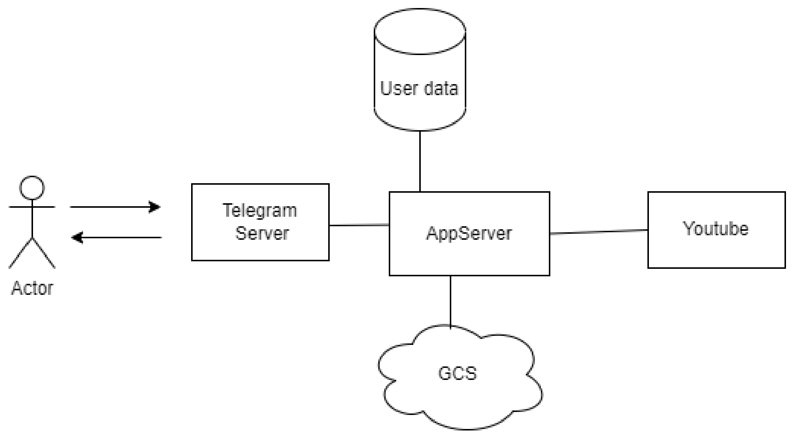


FIGURE 5.1: Main components of the system

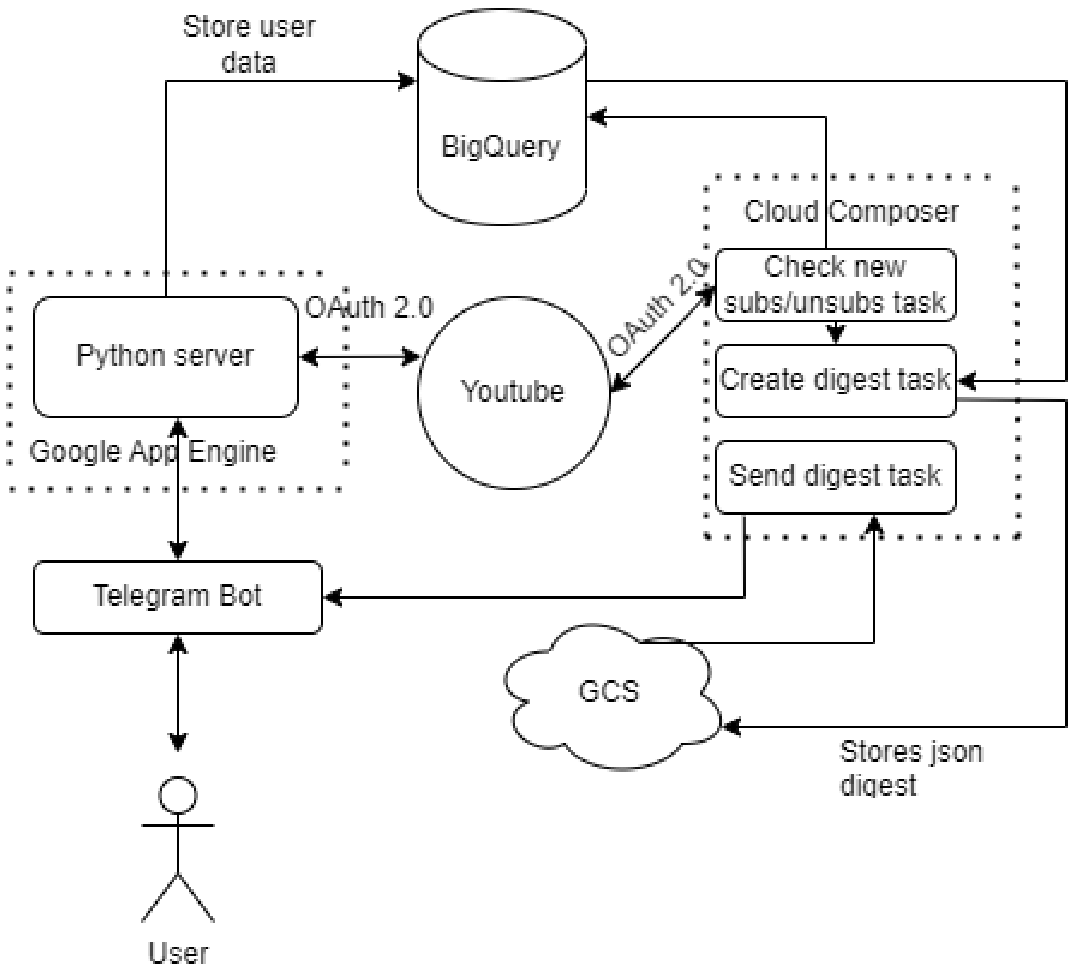


FIGURE 5.2: Complete Architecture

In figure 5.2 there is depicted the complete architecture of the system with concrete technologies and all the necessary services to execute the main logic of the application.

5.2 Choice of Techonogies

5.2.1 BigQuery

The main reason behind using Big Query is its analytical querying capabilities. It allows running complex analytical queries on large data sets, including merging, filtering, and modification operations, not only simple CRUD queries. So in a given application, we are working with a large amount of historical data from which we can potentially gain daily, weekly, and monthly reports. Also, the nature of the data disposes us to use OLAP type of storage rather than OLTP. Because of the fact that we could have a significant amount of historical data and have to run batch jobs to receive exhaustive reports.

Also, one of the considerable advantages of using Big Query as the main storage is that it is a cloud data warehouse platform, which is hyper-scalable, capable of executing SQL queries over petabytes of data, and automatically scaling on-demand to match current needs. And as was discussed before, high scalability is one of the priorities and a non-functional requirement we want to achieve in this system.

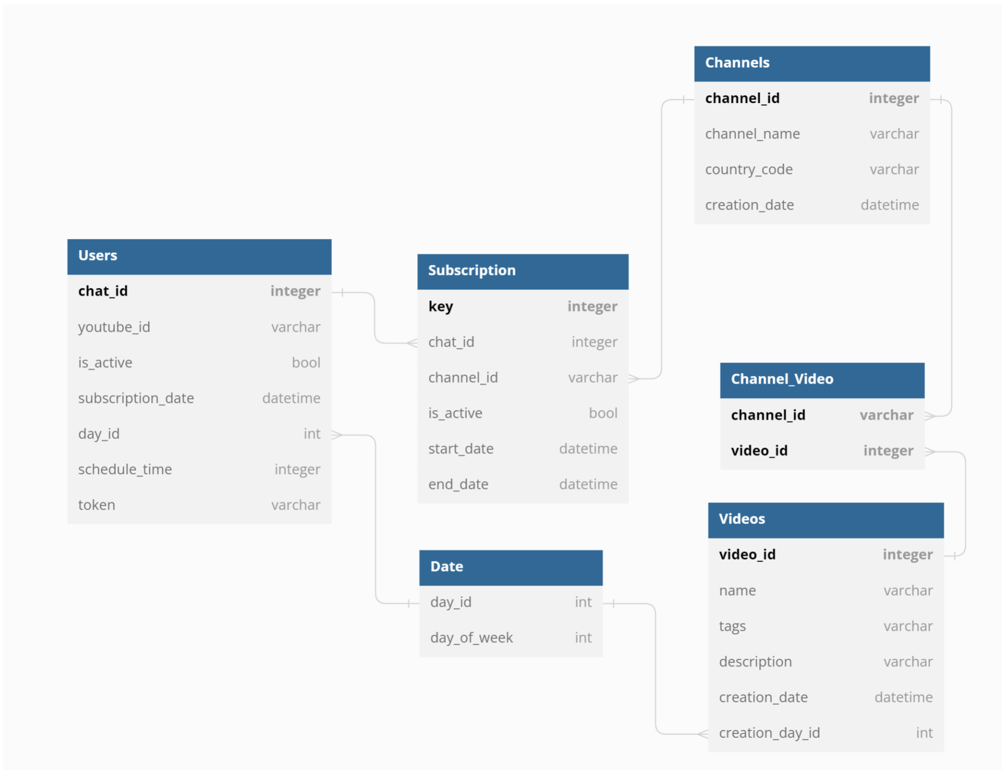


FIGURE 5.3: ER Diagram of BigQuery

Moreover, it has efficient integration and communication with other Google products such as Cloud Composer, GCS, Google Cloud Functions, and so on, which is a significant advantage compared to other solutions such as SnowFlake.

Let's move on to logical data modeling, figure 5.3 represents the entity relational diagram of the databases with main Entities such as Users, Channels, Videos, Subscriptions, and Date. Each Subscription table entry represents the subscription of the user to the channel in a certain period of time. So the whole history of subscriptions is stored using start date and end date attributes, while is active says either the given entry is the actual record or one from the archive.

This approach allows tracking the history of users' subscriptions and preferences changes which in turn makes it possible to analyze the activity of the user, interest among the users in a particular channel, etc. Another entity that simplifies the potential analytics is Date table which represents the day of the week. The Channel Video table simply represents the many-to-many relation between a channel and a published video.

Besides relations important step during designing is to define partitioning and clustering. BigQuery will store separately the different partitions at a physical level. When partitioning a table and then executing a query, it is also BigQuery that determines which partition to access and minimizes the data that must be read. It saves time and cost of the query execution. In most cases and in this particular design creation dates and subscription dates should be the partitioning columns because that would be the main condition in the where clause. When you partition a table and then execute a query, it is also BigQuery that determines which partition to access and minimizes the data that must be read.

5.2.2 Airflow

Airflow is used for the scheduling and orchestration of data pipelines or workflows. In our system, we need to perform a couple of scheduled batch jobs one after another, such as checking new subscriptions and unsubscriptions of users, checking new videos on channels, and actually creating and storing prepared digests overviews on GCS in JSON format, then sending them to users and archiving digests.

Airflow needed to handle the workflow by creating DAG for the listed scheduled tasks. It supports Python, and BigQuery, GCS operators, simplifying the creation of jobs. Also, it is run in a Cloud Composer environment which simplifies the setting up. Furthermore, it supports scaling, both vertical and horizontal which is a priority of the system. It is done by controlling the number of nodes in the Google Kubernetes Engine cluster that will be used to run the environment.

So, there is a cluster set-up to distribute the daemons across multiple machines. It gives room to scale and also increases availability as one of the worker nodes goes down the cluster would be still operational. Figure 5.4 depicts the DAG of tasks to update tables in the main storage and store digests to GCS. There is a separate scheduled DAG to send the digest for the user and after archive it. It is done in a separate workflow because the first part takes a variable amount of time but the second part should be delivered in a precise time. So to simplify the process and to satisfy all the needs it was implemented in two separate DAGs.

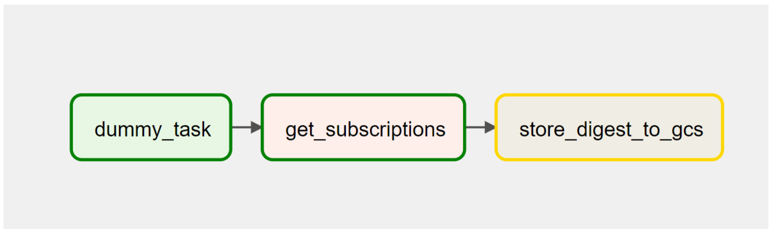
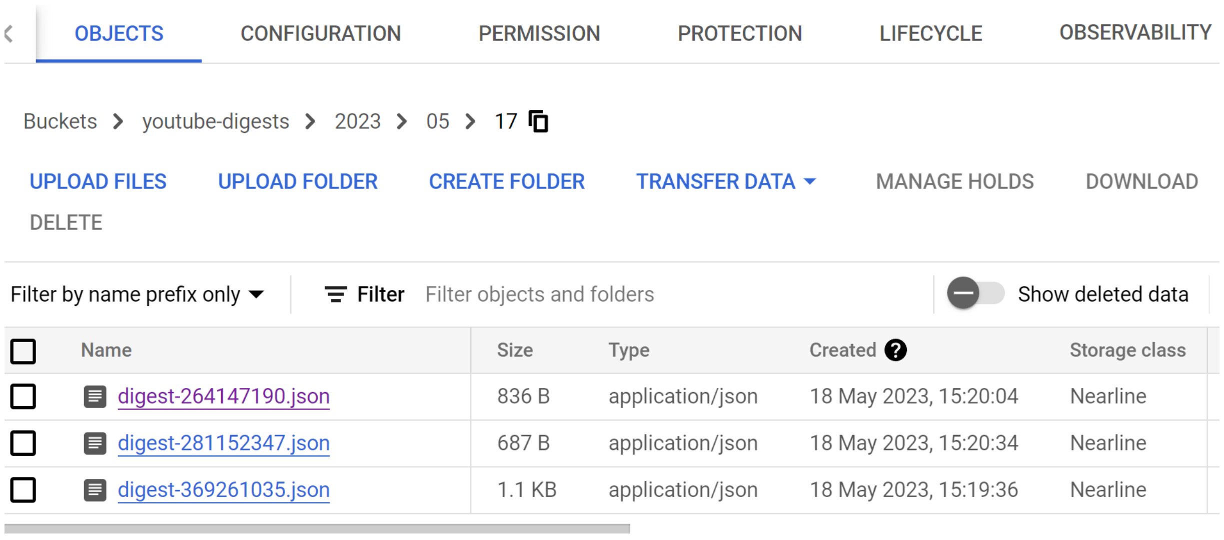


FIGURE 5.4: Airflow DAG

5.2.3 GCS

GCS is object storage for storing digest reports in a JSON format. It is needed to store ready digests before the scheduled time comes. Those JSON files are partitioned in CS by the date of a report because it is the main characteristic by which the server would request data. It is not providing small latency while requesting reports, but it is not a problem in our case because we have plenty of time to do each day before the specified time. More importantly, this storage can handle large amounts of data, is cheap compared to other alternatives, and is efficient and easy to implement communication with other Google Cloud services.

Since this storage for digests is mostly used like a archive for now Nearline storage class is used for the digests bucket.



Name	Size	Type	Created	Storage class
digest-264147190.json	836 B	application/json	18 May 2023, 15:20:04	Nearline
digest-281152347.json	687 B	application/json	18 May 2023, 15:20:34	Nearline
digest-369261035.json	1.1 KB	application/json	18 May 2023, 15:19:36	Nearline

FIGURE 5.5: Google Cloud Storage with ready daily digests

5.2.4 Python Server on Google App Engine

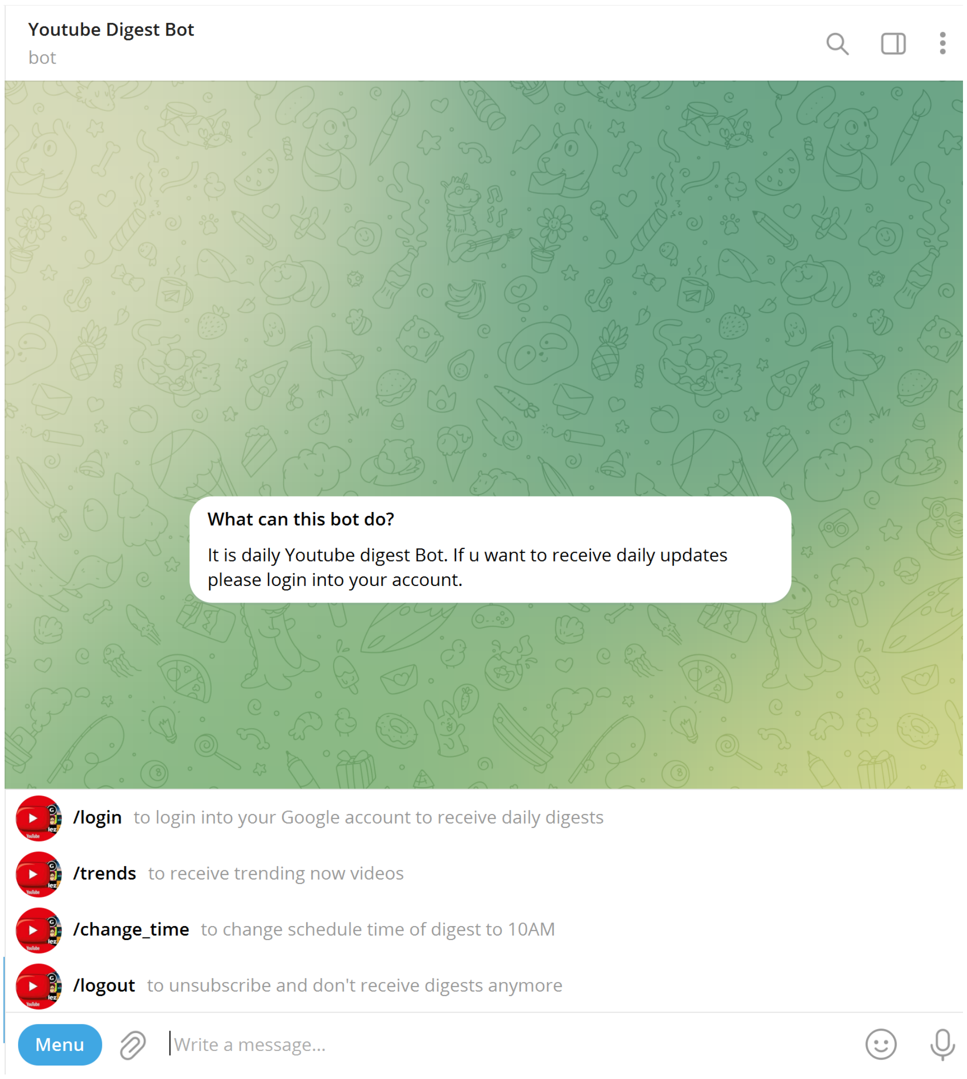
Python server to get data about the user's subscriptions receives permission from them during the first login. Then for each daily request tasks use tokens and refresh tokens to get data via the OAuth 2.0 protocol described in previous chapters. During the first attempt to log in, the user is given a special authorization link with an encoded chat id parameter in it, so after successful registration, the application can store its chat id and Google OAuth token and then redirect the user to Telegram chat. As a logical complement, the application is deployed on Google App Engine. For that purpose, the deployment configuration is defined in yaml file. Gunicorn is the WSGI server to which we are configuring our application to run on. Currently, it runs with four worker processes.

Chapter 6

Digest overview

6.1 Digest Overview

This section will discuss how Digest Bot looks in Telegram and what user experience is. In the very beginning when the user opens a chat with the bot, there is a menu with available commands and their description Figure 6.1.

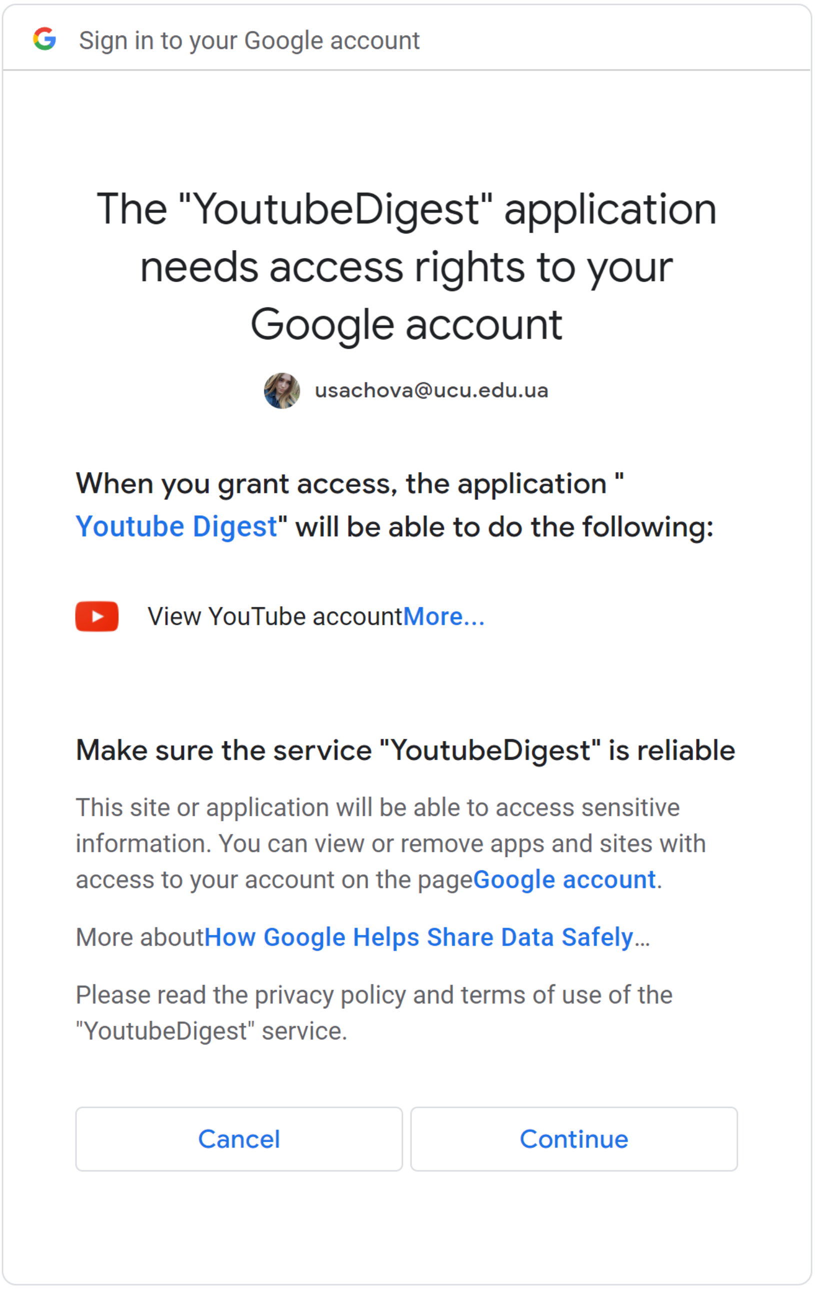

Youtube Digest Bot
bot

What can this bot do?
It is daily Youtube digest Bot. If u want to receive daily updates please login into your account.

/login to login into your Google account to receive daily digests
/trends to receive trending now videos
/change_time to change schedule time of digest to 10AM
/logout to unsubscribe and don't receive digests anymore

FIGURE 6.1: Basic commands of the bot

After that to receive daily digests user should use the login command. In the next message, he/she will get an invite link for Google authorization. After clicking on this link user is redirected to the OAuth window which asks to log in and then to grant access to the Youtube account in read-only mode. Important note: as the application is in test mode and isn't yet verified by Google it is possible to see a warning window from Google. So it may require you to click proceed word in this window to confirm that you trust the application. On figure 6.2 default OAuth window is depicted. Right after successfully login via the account the user data is stored in the main storage and the user is redirected to Telegram Bot again as it is on Figure 6.3.



Sign in to your Google account

The "YoutubeDigest" application needs access rights to your Google account

usachova@ucu.edu.ua

When you grant access, the application " Youtube Digest" will be able to do the following:

View YouTube accountMore...

Make sure the service "YoutubeDigest" is reliable

This site or application will be able to access sensitive information. You can view or remove apps and sites with access to your account on the pageGoogle account.

More aboutHow Google Helps Share Data Safely...

Please read the privacy policy and terms of use of the "YoutubeDigest" service.

FIGURE 6.2: Granting access window during Google Authorization

After completing authorization and returning to Bot user would receive a digest at a scheduled time which by default is 8 PM. If it is needed using change time command digest time can be changed to the morning time which is 10 AM for now. In addition, the user can get currently trending videos on Youtube in the country

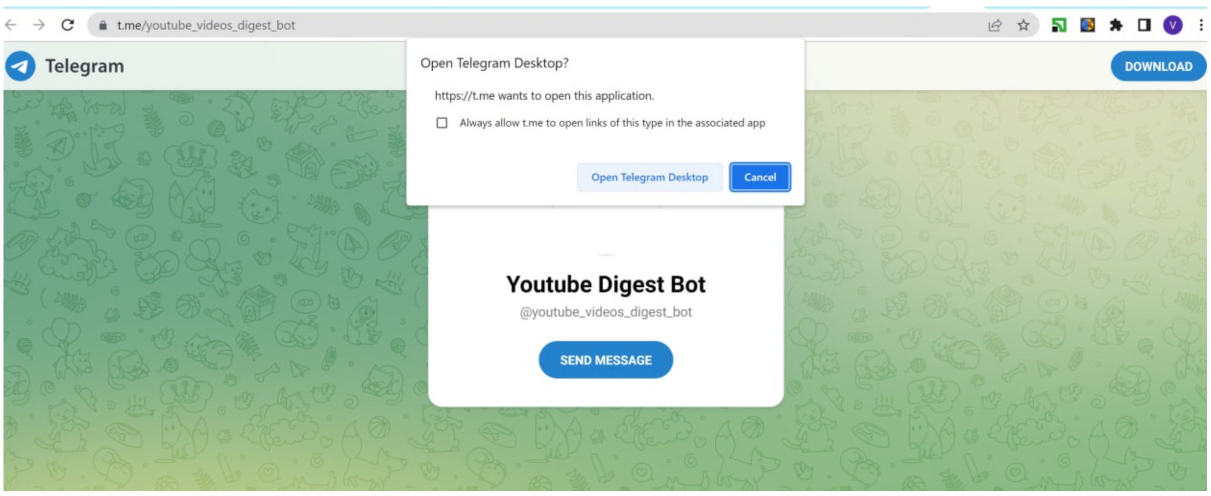

t.me/youtube_videos_digest_bot
Telegram
DOWNLOAD
Open Telegram Desktop?
https://t.me wants to open this application.
Always allow t.me to open links of this type in the associated app
Open Telegram Desktop
Cancel
Youtube Digest Bot
@youtube_videos_digest_bot
SEND MESSAGE

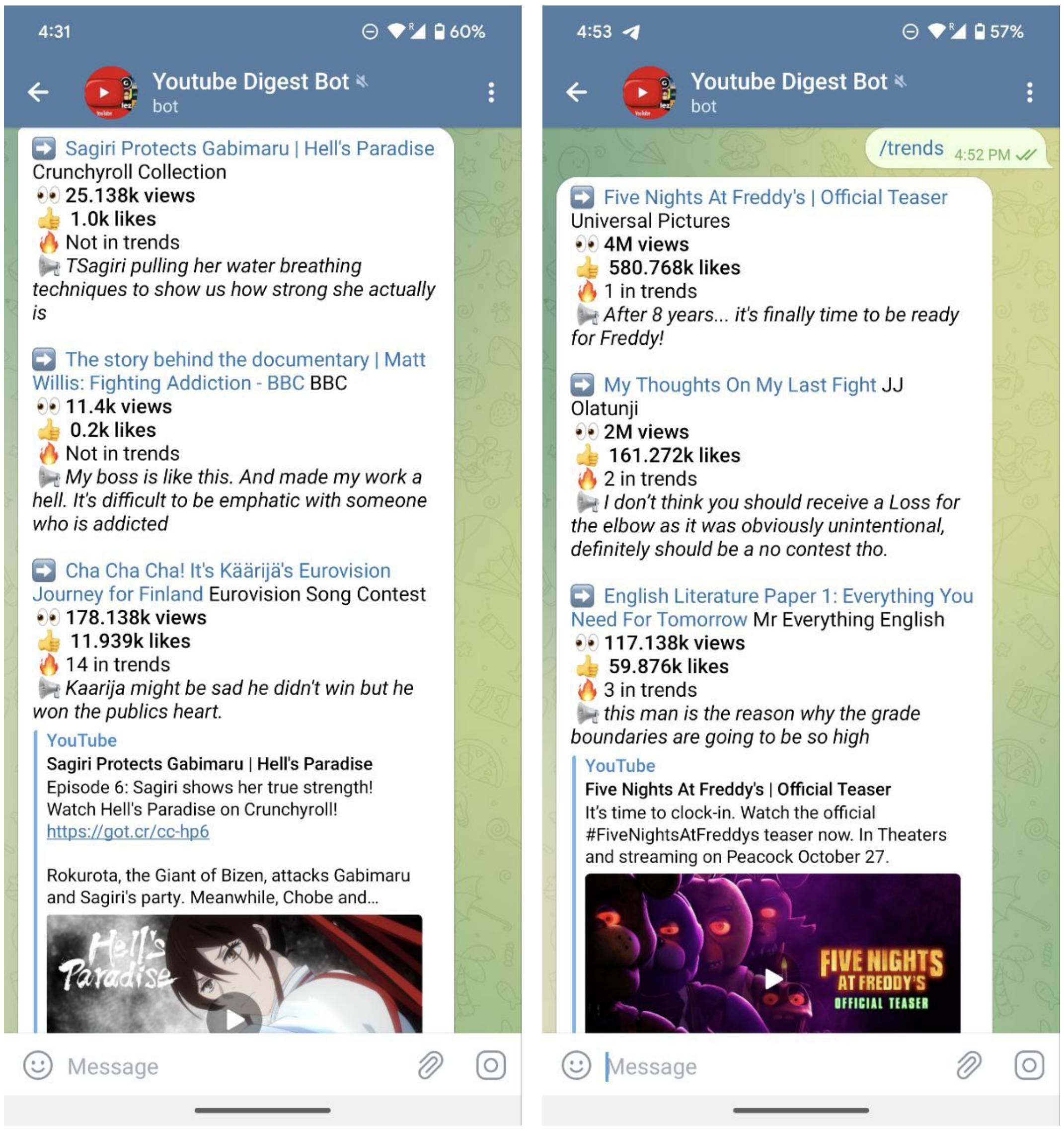
FIGURE 6.3: Redirection to Telegram after successful authorization

which is attached to his/her account. In case the user isn't logged in it is still possible to get trends but not necessarily in a user's country. On the next two figures Figure 6.4 and Figure 6.5 we can observe how digest and trends report look like.

As discussed in previous chapters Airflow scheduled tasks create and store digest for all users on GCS and then a separate task is sending users at the agreed time. With trending videos the flow is different, for this action simply the Python server requests Youtube Data API and sends a message to the user. But for both actions, the general view is pretty much the same. The information available about each video is:

- Title
- Channel title
- Number of views
- Number of likes
- Position in trends
- Top comment
- links to all the videos

The list is not exhaustive and could be expanded as Data API offers almost all possible data about videos and channels.


4:31 60%

Youtube Digest Bot
bot

➡️ Sagiri Protects Gabimaru | Hell's Paradise
Crunchyroll Collection
👀 25.138k views
👍 1.0k likes
🔥 Not in trends
📢 *TSagiri pulling her water breathing techniques to show us how strong she actually is*

➡️ The story behind the documentary | Matt Willis: Fighting Addiction - BBC BBC
👀 11.4k views
👍 0.2k likes
🔥 Not in trends
📢 *My boss is like this. And made my work a hell. It's difficult to be emphatic with someone who is addicted*

➡️ Cha Cha Cha! It's Käärijä's Eurovision Journey for Finland Eurovision Song Contest
👀 178.138k views
👍 11.939k likes
🔥 14 in trends
📢 *Kaarija might be sad he didn't win but he won the publics heart.*

YouTube
Sagiri Protects Gabimaru | Hell's Paradise
Episode 6: Sagiri shows her true strength!
Watch Hell's Paradise on Crunchyroll!
https://got.cr/cc-hp6

Rokurota, the Giant of Bizen, attacks Gabimaru and Sagiri's party. Meanwhile, Chobe and...

Message

4:53 57%

Youtube Digest Bot
bot

/trends 4:52 PM

➡️ Five Nights At Freddy's | Official Teaser
Universal Pictures
👀 4M views
👍 580.768k likes
🔥 1 in trends
📢 *After 8 years... it's finally time to be ready for Freddy!*

➡️ My Thoughts On My Last Fight JJ Olatunji
👀 2M views
👍 161.272k likes
🔥 2 in trends
📢 *I don't think you should receive a Loss for the elbow as it was obviously unintentional, definitely should be a no contest tho.*

➡️ English Literature Paper 1: Everything You Need For Tomorrow Mr Everything English
👀 117.138k views
👍 59.876k likes
🔥 3 in trends
📢 *this man is the reason why the grade boundaries are going to be so high*

YouTube
Five Nights At Freddy's | Official Teaser
It's time to clock-in. Watch the official #FiveNightsAtFreddys teaser now. In Theaters and streaming on Peacock October 27.

Message

FIGURE 6.4: Digest and Trending video appearance example

6.2 Future improvements

This section will describe possible future improvements in appearance, architecture, and additional functionalities.

- The first thing to improve is giving the user more freedom in choosing digest schedule time. This requires scheduling more almost identical Airflow tasks but isn't changing the general architecture.
- Personalization could be improved in the future based on statistics we have in BigQuery storage. This data could be helpful in sorting videos from subscriptions and suggesting previous videos or other channel activities. Having all the data historical data in the main storage it becomes possible to improve personalization and offer to the user weekly/monthly digests and so on.
- The digest format could be improved according to the user's preferences. For example adding or removing a number of likes, top comments, etc.
- The other architectural improvement which could be implemented in the future is the creation of a staging area in the main storage which would prevent many possible inconsistencies in case of multiple failures of the scheduled jobs.

Chapter 7

Conclusions

In this work, I determined the functional and non-functional requirements of the system according to the needs of the demo version and the potential growth of the application as well. Considering availability and scaleability as main non-functional requirements was designed efficient architecture and chosen appropriate technologies, storage, and ways of communication. The possibilities of each of the technologies were considered in terms of requirements. Also, external APIs were reviewed and successfully integrated into the system so the personalized digest platform can service users. As a result, a working demo application is given in the writing.

In conclusion, we have achieved most of what was planned for the application, even though there still are a few finishing touches left.

Bibliography

API Reference (n.d.). URL: `https://developers.google.com/youtube/v3/docs/` (visited on 06/01/2022).

Gorton, Ian (2022). *Foundations of Scalable Systems: Designing Distributed Architectures*. O'Reilly Media. ISBN: 1098106067.

Granger, Romain (June 1, 2022). *How to Use Partitions and Clusters in BigQuery Using SQL*. URL: `https://towardsdatascience.com/how-to-use-partitions-and-clusters-in-bigquery-using-sql-ccf84c89dd65`.

Implementing OAuth 2.0 Authorization (n.d.). URL: `https://developers.google.com/youtube/v3/guides/authentication` (visited on 06/01/2022).

Kukhnavets, Pavel (Aug. 23, 2020). *Defining Functional and Nonfunctional Requirements*. URL: `https://hygger.io/blog/functional-and-nonfunctional-requirements/`.

Matuts, Olga (Dec. 20, 2019). *Functional Requirements and Quality Attributes*. URL: `https://welldoneby.com/blog/functional-requirements-and-quality-attributes-your-short-guide/`.

Use Case Specification (n.d.). URL: `https://www.ictdemy.com/software-design/uml/uml-use-case-specification` (visited on 06/01/2022).

Using OAuth 2.0 to Access Google APIs (n.d.). URL: `https://developers.google.com/identity/protocols/oauth2` (visited on 06/01/2022).