

Building a GUI for viewing data from Apache Kafka



LTTS
GLOBAL
ENGINEERING
ACADEMY



L&T Technology Services

Mentor Name: Vimal komath

Ps.no: 954963



Document History

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	Approved By	Remarks/Revision Details
1	23/04/2021	Nandyala Naveen Kumar Reddy (99003604)			
2	23/04/2021	Satyasindhu Thummala (99003534)			
3	23/04/2021	Vikram Hedge (99003597)			

Contents

1.APACHE KAFKA.....	4
1.1 KAFKA	4
WHAT IS KAFKA?.....	4
WHY KAFKA?.....	4
WHY IS KAFKA SO POPULAR?.....	5
1.2 MAIN COMPONENTS OF KAFKA	5
1.3 KAFKA CLUSTER.....	6
1.4 BENEFITS OF KAFKA	9
2.SOFTWARE TOOLS AND PACKAGES INSTALLED	9
3.BUILDING OF KAFKA SERVER.....	10
4.GUI (GRAPHICAL USER INTERFACE)	11
4.1 GUI OVERVIEW	11
4.2 GUI EXAMPLES	12
4.3 ADVANTAGES OF GUI	12
5 IMPLEMENTATION.....	13
5.1 CLIENT AND KAFKA RUNNING ON THE SAME LOCAL MACHINE	13
5.1.1 GUI CLICK BUTTON.....	13
5.1.2 KAFKA PRODUCER CODING	13
5.1.3 KAFKA CONSUMER CODING.....	14
5.1.4 INTERFACE OF GUI AND KAFKA PRODUCER.....	15
5.1.5 INTERFACE OF GUI AND KAFKA CONSUMER	16
5.1.6 FINAL OUTPUT	16
5.2 CLIENT AND KAFKA RUNNING ON THE DIFFERENT MACHINES.....	17
5.2.1 OUTPUT OF KAFKA PRODUCER ON DIFFERENT MACHINE	18
5.2.2 OUTPUT OF KAFKA CONSUMER ON DIFFERENT MACHINE	18
6.PROJECT OUTCOMES	19
7.CHALLENGES FACED.....	19
8.CHALLENGES OVERCOME	19
9.CONTRIBUTIONS	20
INDIVIDUAL CONTRIBUTIONS.....	20
TEAM CONTRIBUTION	20
10.LINKS	20

1. Apache Kafka

1.1 Kafka

What is Kafka?

Kafka is a distributed streaming platform that is used to publish and subscribe to streams of records.

It is a messaging system that is designed to be fast, scalable, and durable. It is an open-source stream processing platform. It aims at providing a high-throughput, low-latency platform for handling real-time data feeds.

Apache describes Kafka as a distributed streaming platform that lets us:

1. Publish and subscribe to streams of records.
2. Store streams of records in a fault-tolerant way.
3. Process streams of records as they occur.

Why Kafka?

In simple words Kafka is a messaging system which transfers data between source system and target system, but imagine if we are having many source systems and target systems the interaction and transfer of data from source systems and target system will be more complicated. This is where the Kafka server plays an important role.

In short, Kafka is used for stream processing, website activity tracking, metrics collection and monitoring, log aggregation, real-time analytics, CEP, ingesting data into Spark, ingesting data into Hadoop, CQRS, replay messages, error recovery, and guaranteed distributed commit log for in-memory computing (microservices).

The functionalities that it provides are well-suited for many applications and thus we use Kafka for:

1. Building real-time streaming data pipelines that can get data between systems and applications.
2. Building real-time streaming applications to react to the stream of data.

It's also used by other companies like Spotify, Uber, Tumbler, Goldman Sachs, PayPal, Box, Cisco, Cloudflare, and Netflix.

Why Is Kafka So Popular?

Kafka has operational simplicity. Kafka is easy to set up and use, and it is easy to figure out how Kafka works. However, the main reason Kafka is very popular is its excellent performance. It is stable, provides reliable durability, has a flexible publish-subscribe/queue that scales well with N-number of consumer groups, has robust replication, provides producers with tunable consistency guarantees, and it provides preserved ordering at the shard level (i.e., Kafka topic partition).

In addition, Kafka works well with systems that have data streams to process and enables those systems to aggregate, transform, and load into other stores. But none of those characteristics would matter if Kafka was slow.

The most important reason Kafka is popular is Kafka's exceptional performance.

1.2 Main components of Kafka

- **Kafka Producer**

Kafka producers write to Topic. It is Kafka client that publishes records to the Kafka cluster. The producer is thread safe and sharing a single producer instance across threads will generally be faster than having multiple instances.

The producer consists of a pool of buffer space that holds records that haven't yet been transmitted to the server as well as a background I/O thread that is responsible for turning these records into requests and transmitting them to the cluster. Failure to close the producer after use will leak these resources.

- **Kafka ecosystem**

A messaging system is a system that is used for transferring data from one application to another so that the applications can focus on data and not on how to share it.

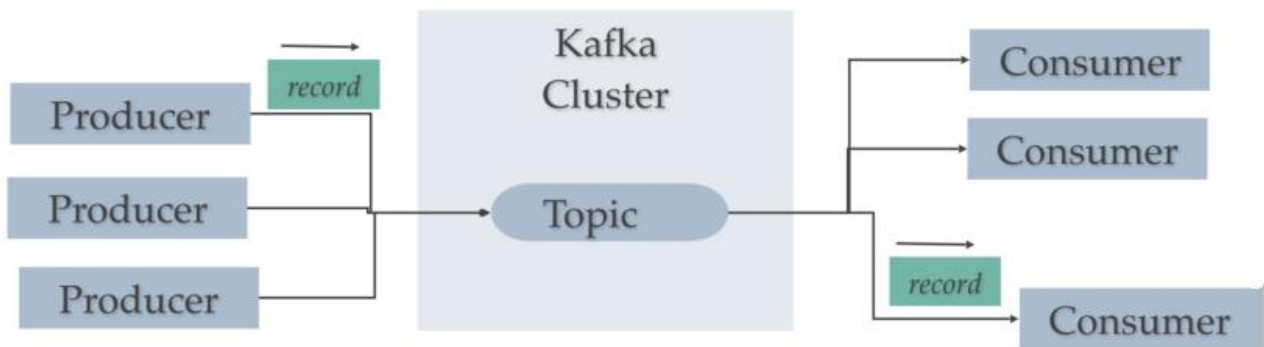
- **Kafka Consumer**

Kafka consumers read from Topics, It is a client that consumes records from a Kafka cluster

This client transparently handles the failure of Kafka brokers, and transparently adapts as topic partitions it fetches migrate within the cluster. This client also interacts with the broker to allow groups of consumers to load balance consumption using consumer groups.

Consumer group

Consumer groups give Kafka the flexibility to have the advantages of both message queuing and publish-subscribe models.



1.3 Kafka cluster

Basics of Kafka

- Kafka runs as a cluster on one or more servers.
- The Kafka cluster stores a stream of records in categories called topics.
- Each record consists of a key, a value, and a timestamp.

Cluster

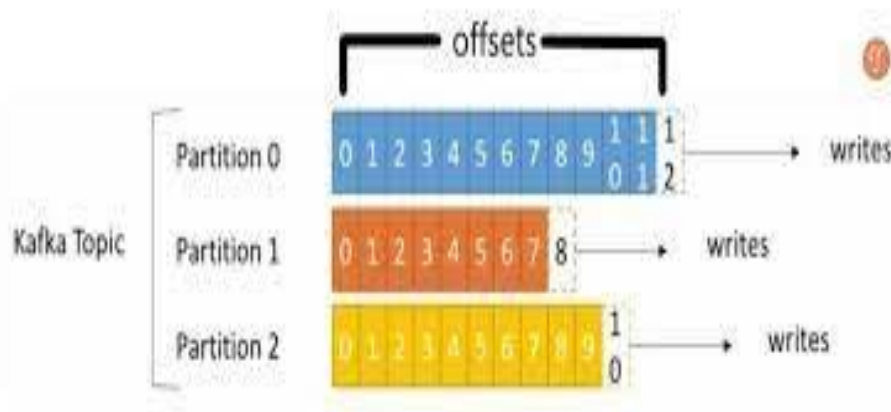
When Kafka has more than one broker, it is called a Kafka cluster. A Kafka cluster can be expanded without downtime. These clusters are used to manage the persistence and replication of message data.

Topics and Logs

Topic is unique name given to the records which are published

It can have many topics and each topic can have multi-subscribers

For each topic, the Kafka cluster maintains a partition log that looks like this:



Partitions and partition offsets:

Any topic may have many partitions it is decided by the publisher based on the data in the topics so that it can handle an arbitrary amount of data. In the above diagram, the topic is configured into three partitions (partition {0,1,2}). Partition 0 has 13 offsets, Partition 1 has 10 offsets, and Partition 2 has 13 offsets.

Partition offsets

Each partitioned message has a unique sequence ID called an offset. For example, in Partition 1, the offset is marked from 0 to 9.

Brokers

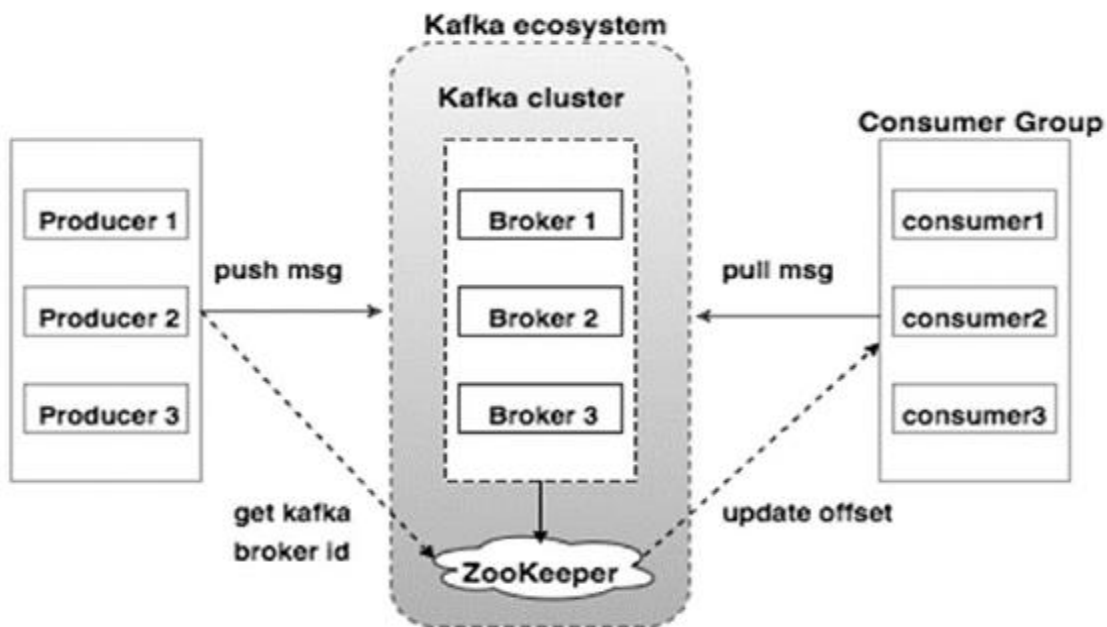
- A Kafka broker receives messages from producers and stores them on disk keyed by unique **offset**.
- A Kafka broker allows consumers to fetch messages by topic, partition and offset.
- Kafka brokers can create a Kafka cluster by sharing information between each other directly or indirectly using Zookeeper.
- If there is more than one broker then it is called cluster.

Zookeeper

Kafka brokers are stateless, so they use Zookeeper for maintaining their cluster state. It notifies the producer and consumer about the presence or failure of a broker based on which producer and consumer makes a decision and starts coordinating their tasks with some other broker.

Replicas

Replicas are nothing but backups of a partition. If the replication factor of the above topic is set to 4, then Kafka will create four identical replicas of each partition and place them in the cluster to make them available for all its operations. Replicas are never used to read or write data. They are used to prevent data loss.



1.4 Benefits of Kafka

Four main benefits of Kafka are:

1. **Reliability.** Kafka is distributed, partitioned, replicated, and fault tolerant. Kafka replicates data and is able to support multiple subscribers. Additionally, it automatically balances consumers in the event of failure.
2. **Scalability.** Kafka is a distributed system that scales quickly and easily without incurring any downtime.
3. **Durability.** Kafka uses a distributed commit log, which means messages persists on disk as fast as possible providing intra-cluster replication, hence it is durable.
4. **Performance.** Kafka has high throughput for both publishing and subscribing messages. It maintains stable performance even when dealing with many terabytes of stored messages.

2. Software tools and packages installed

- Python PyCharm
- Python
- Java
- Pip3 install pyqt5
- Pip3 install pyqt5-tools

3. Building of Kafka server

STEP 1: GET KAFKA

Download the latest Kafka release and extract it

- Scala 2.13 - [kafka_2.13-2.7.0.tgz](#) (asc, sha512)

STEP 2: START THE KAFKA ENVIRONMENT

Our local environment must also have java installed to set Kafka server

Command for starting a Zookeeper server

.bin\windows\zookeeper-server-start.bat. \config\zookeeper. Properties

Command for starting a broker server

.bin\windows\kafka-server-start.bat. \config\server. Properties

Once all services have successfully launched, basic Kafka environment will be running and ready to use.

Note: Use different command prompts for each command

STEP 3: CREATE A TOPIC TO STORE YOUR EVENTS

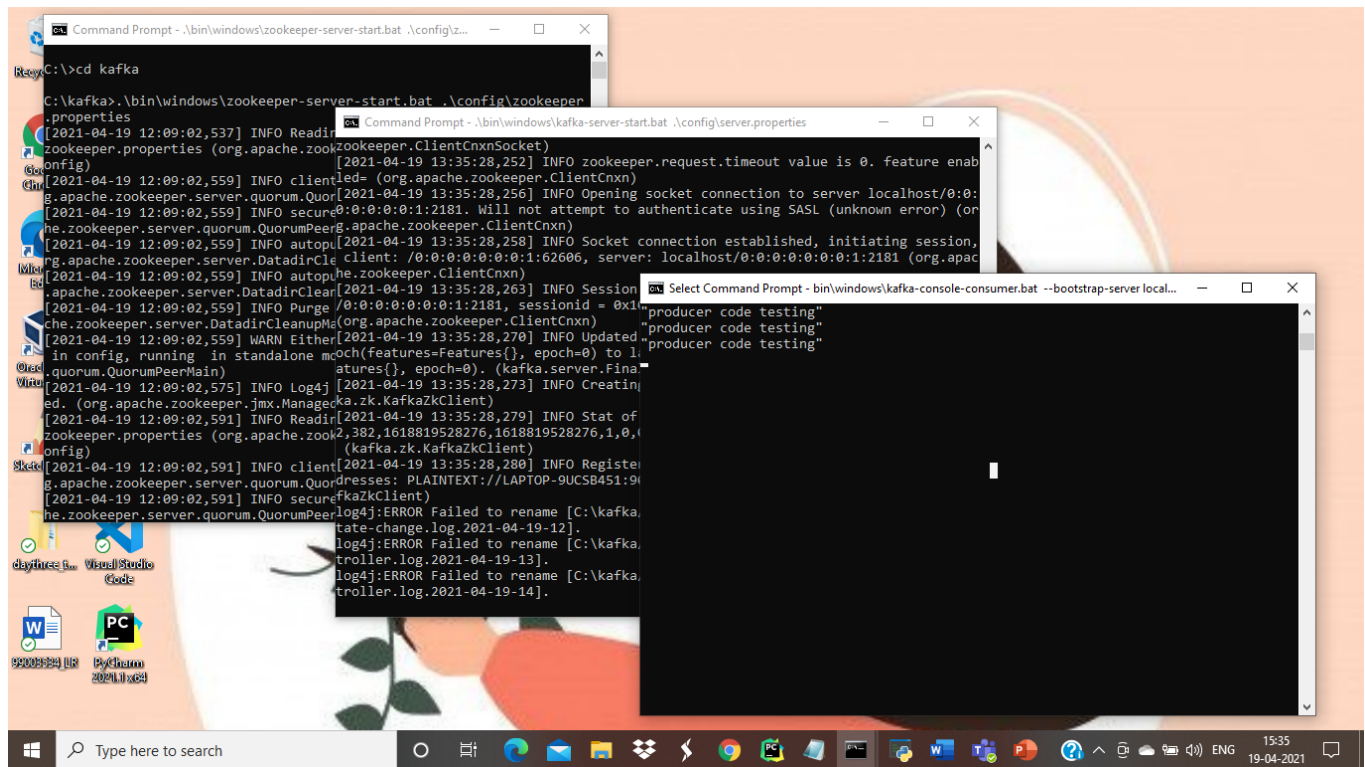
Note: use a new command prompt.

Command for creating topic, partitions and replicas and addressing the topic

.bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic team

Step 4: To check the data stored in the topic given by the producer

bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic test --from-beginning



4.GUI (graphical user interface)

4.1 GUI overview

A GUI (graphical user interface) is a system of interactive visual components for computer software. A GUI displays objects that convey information, and represent actions that can be taken by the user. The objects change color, size, or visibility when the user interacts with them.

GUI includes GUI objects like icons, cursors, and buttons. These graphical elements are sometimes enhanced with sounds, or visual effects like transparency and drop shadows. Using these objects, a user can use the computer without having to know commands.

GUI uses a combination of technologies and devices to provide a platform that users can interact with, for the tasks of gathering and producing information.

4.2 GUI Examples

Some popular, modern graphical user interface examples include Microsoft Windows, macOS, Ubuntu Unity, and GNOME Shell for desktop environments, and Android, Apple's iOS, BlackBerry OS, Windows 10 Mobile, Palm OS-WebOS, and Firefox OS for smartphones.

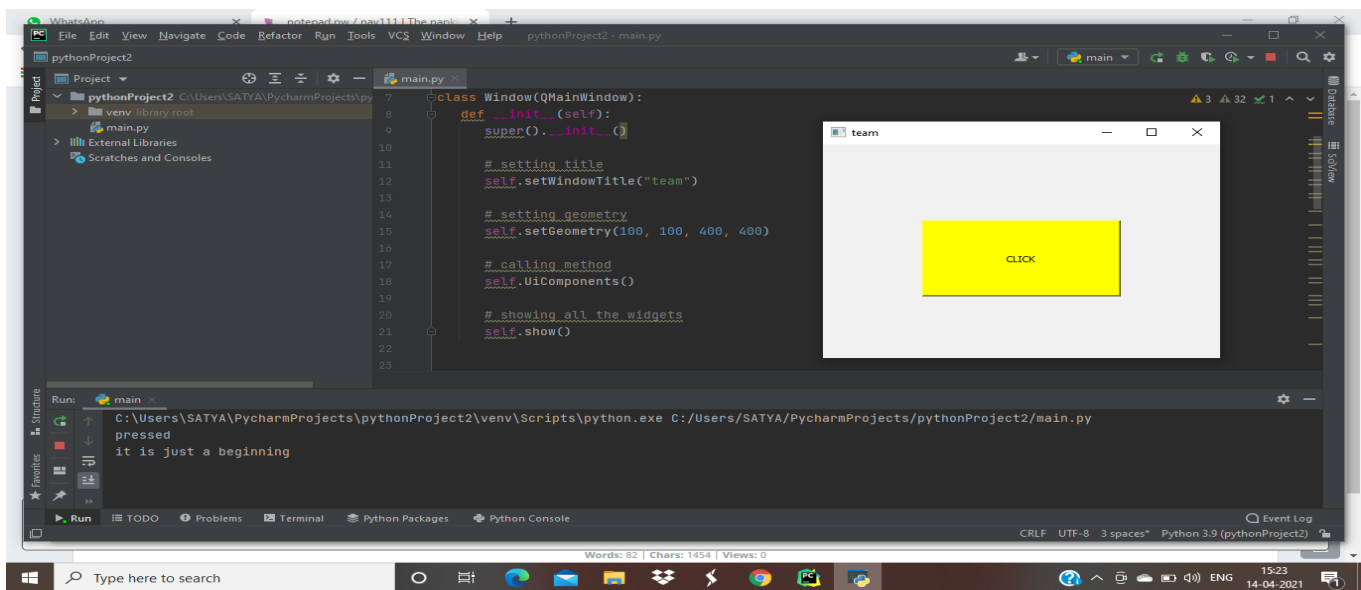
4.3 Advantages of GUI

- GUIs enable interaction through clarity and control
- Effective GUIs facilitate a seamless user journey
- Good GUI design can be shaped to anticipate audience needs
- GUIs capture attention, and keep it
- GUIs deliver consistency . . . consistently
- GUIs foreground storytelling and organization with visual hierarchies
- Effective GUIs heighten engagement with integrated contextualized clue

5 Implementation

5.1 Client and Kafka running on the same local machine

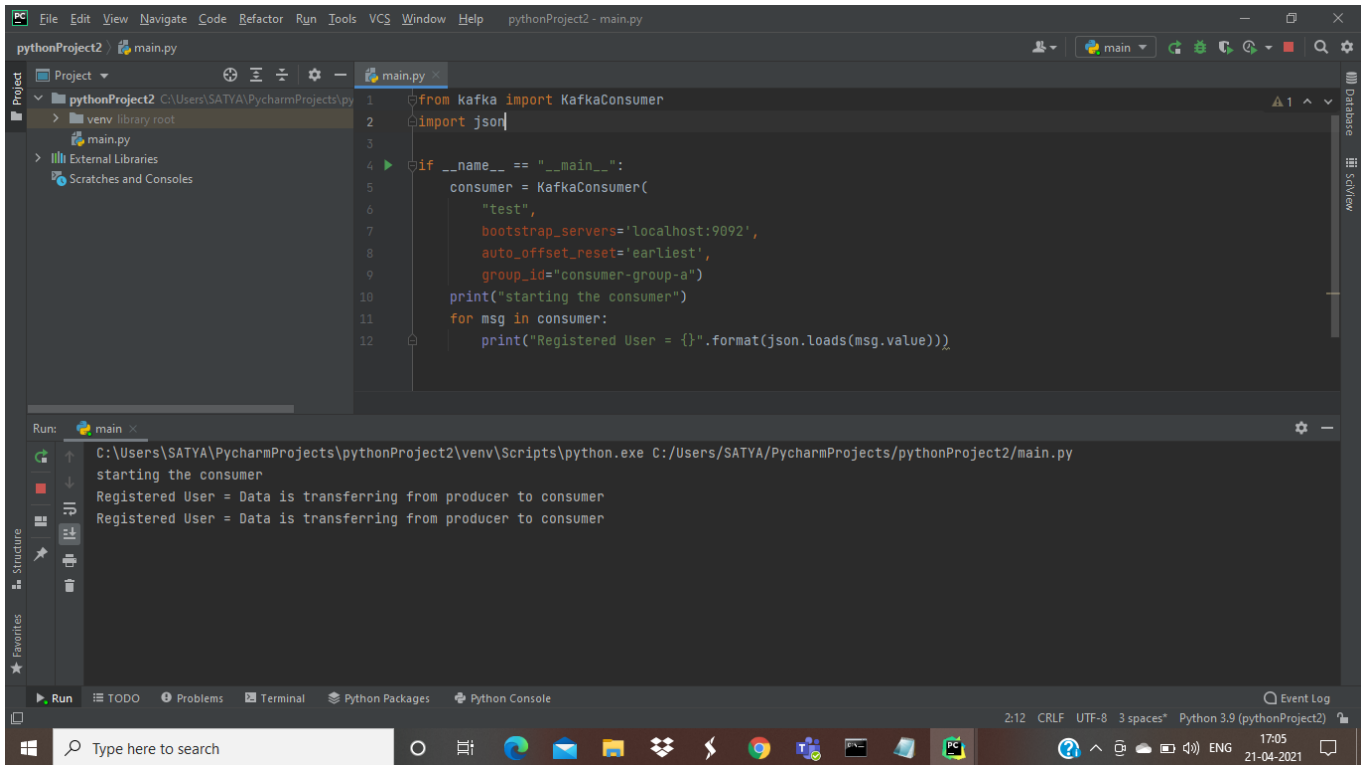
5.1.1 GUI click button



5.1.2 Kafka Producer coding

Configurations needed by producer to interface with Kafka server

- **bootstrap_server**
it indicates the address of topic specified
- **topic**
it is the unique name given by the producer to the data/records that are transferring to Kafka server
- **value_serializer**
before transferring the data, it should be serialized
- **call of send method**
send method as to be initialized for sending data



The screenshot shows the PyCharm IDE with a Python project named 'pythonProject2'. The main.py file contains the following code:

```
1 from kafka import KafkaConsumer
2 import json
3
4 if __name__ == "__main__":
5     consumer = KafkaConsumer(
6         "test",
7         bootstrap_servers='localhost:9092',
8         auto_offset_reset='earliest',
9         group_id="consumer-group-a")
10    print("starting the consumer")
11    for msg in consumer:
12        print("Registered User = {}".format(json.loads(msg.value)))
```

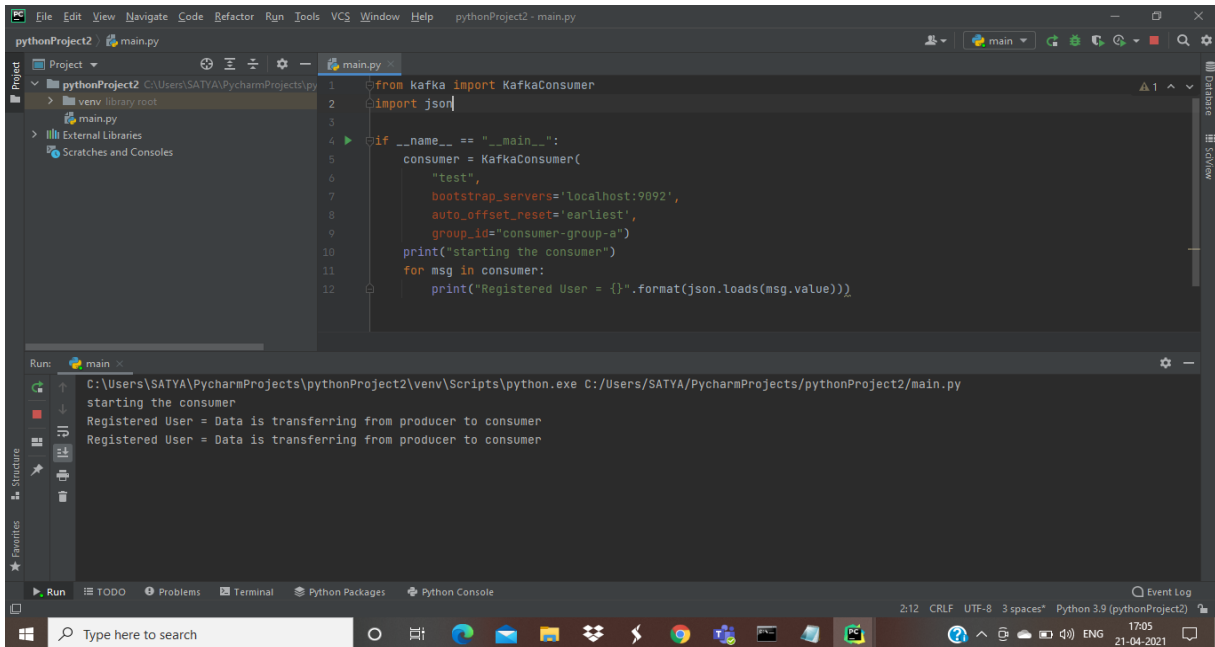
The Run console shows the output of the program:

```
C:\Users\SATYA\PycharmProjects\pythonProject2\venv\Scripts\python.exe C:/Users/SATYA/PycharmProjects/pythonProject2/main.py
starting the consumer
Registered User = Data is transferring from producer to consumer
Registered User = Data is transferring from producer to consumer
```

5.1.3 Kafka Consumer coding

Configurations needed by consumer to interface with Kafka server

- **Bootstrap_server**
It indicates the address of the topic from which data is getting consumed
- **Topic**
Unique topic has to be mentioned to know exactly from which topic data is consuming
- **Auto_offset_reset**
It auto starts the reading the topic from first message it is storing Offset and reset
- **Group_id**
Every consumer will be a part of consumer group
Group id should be created to indicate to which group consumer belongs to



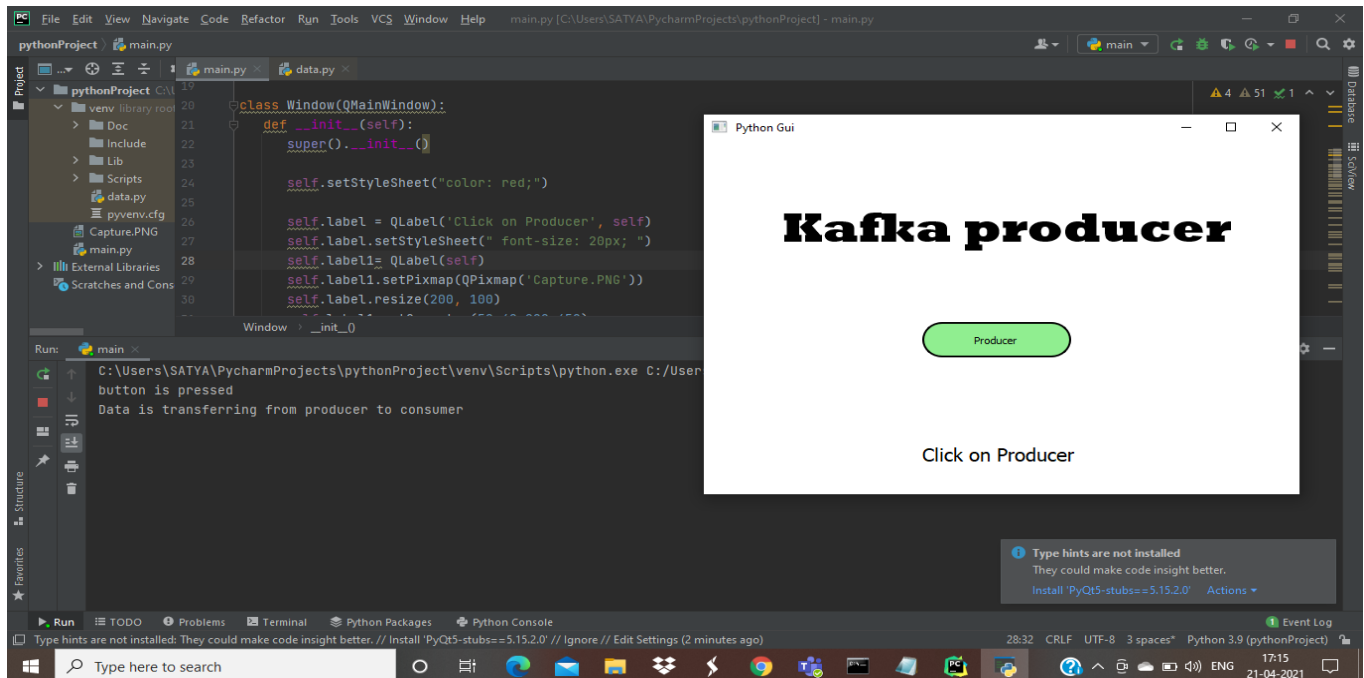
```
pythonProject2 - main.py
1 from kafka import KafkaConsumer
2 import json
3
4 if __name__ == "__main__":
5     consumer = KafkaConsumer(
6         "test",
7         bootstrap_servers='localhost:9092',
8         auto_offset_reset='earliest',
9         group_id="consumer-group-a")
10    print("starting the consumer")
11    for msg in consumer:
12        print("Registered User = {}".format(json.loads(msg.value)))
```

Run: main

C:\Users\SATYA\PycharmProjects\pythonProject2\venv\Scripts\python.exe C:/Users/SATYA/PycharmProjects/pythonProject2/main.py

starting the consumer
Registered User = Data is transferring from producer to consumer
Registered User = Data is transferring from producer to consumer

5.1.4 Interface of GUI and Kafka producer



```
pythonProject - main.py
19 class Window(QMainWindow):
20     def __init__(self):
21         super().__init__()
22         self.setStyleSheet("color: red;")
23
24         self.label = QLabel('Click on Producer', self)
25         self.label.setStyleSheet("font-size: 20px; ")
26         self.label1 = QLabel(self)
27         self.label1.setPixmap(QPixmap('Capture.PNG'))
28         self.label.resize(200, 100)
```

Run: main

C:\Users\SATYA\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/SATYA/PycharmProjects/pythonProject/main.py

button is pressed
Data is transferring from producer to consumer

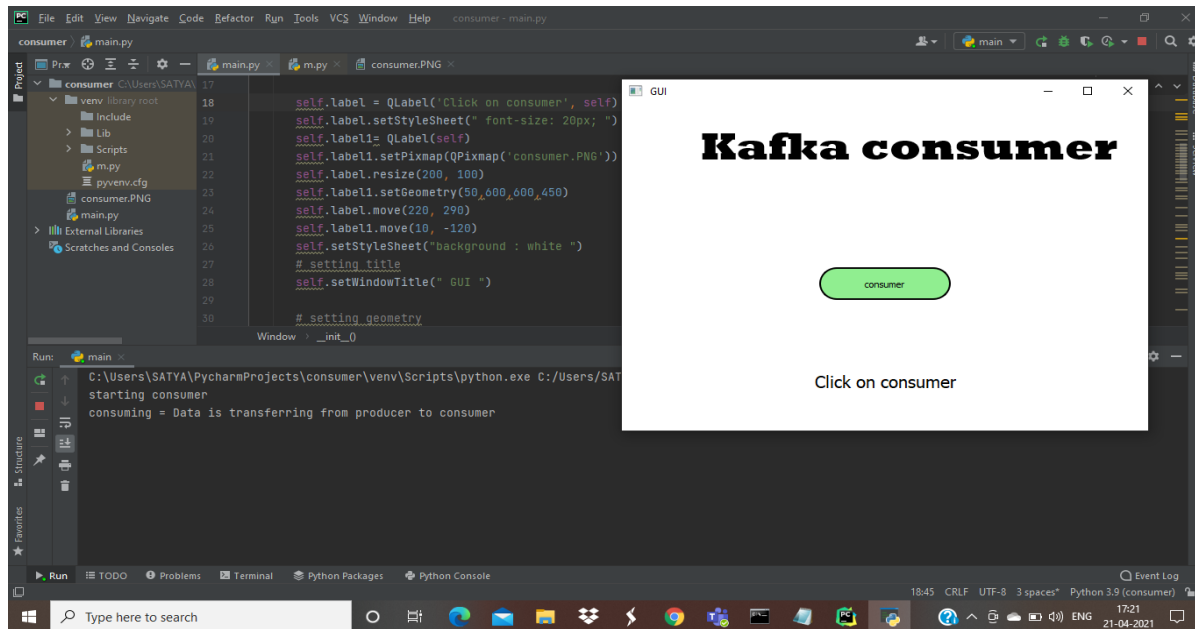
Python Gui

Kafka producer

Producer

Click on Producer

5.1.5 Interface of GUI and Kafka consumer

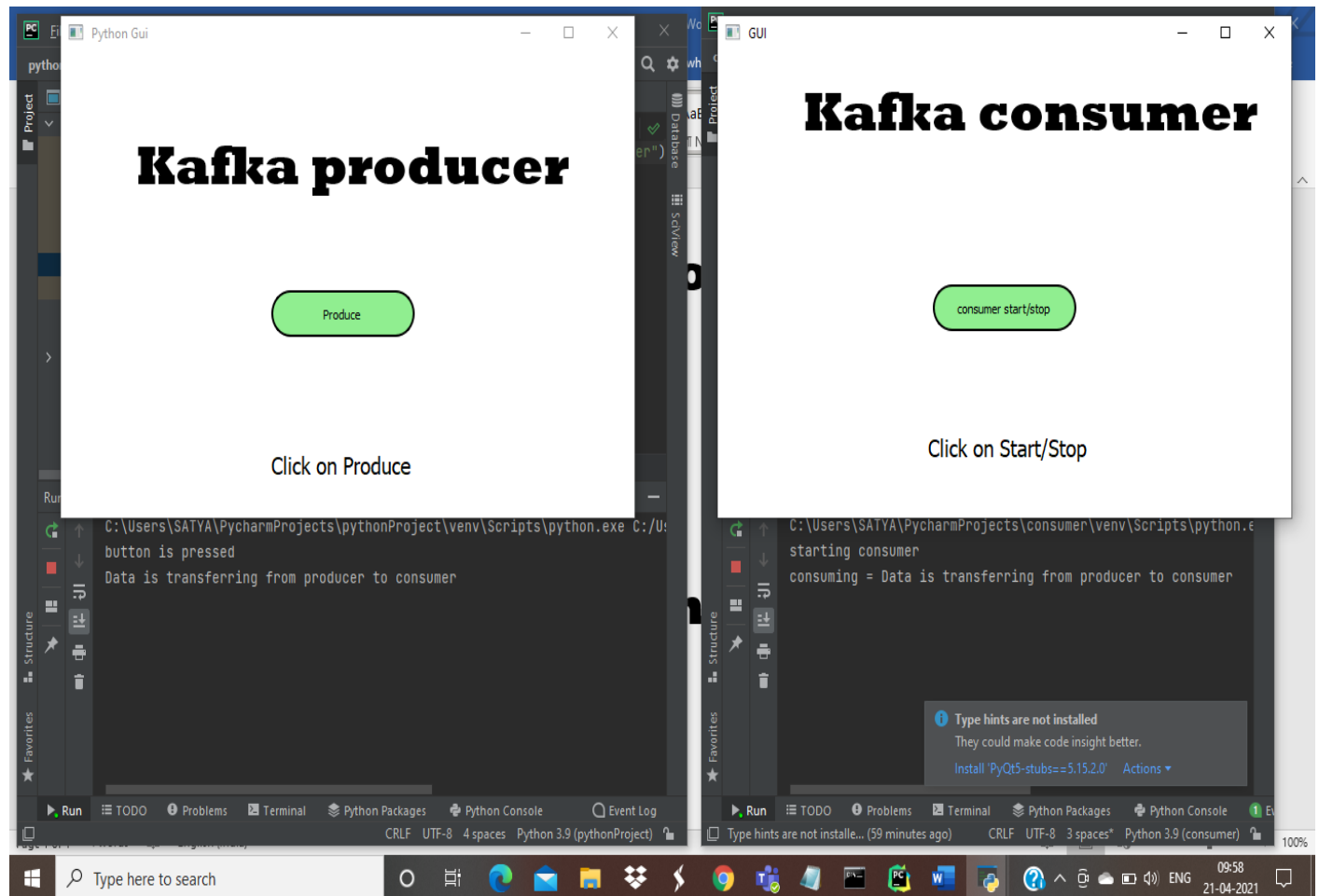


5.1.6 Final output

A separate source system (Kafka producer) is built and it is interfaced with GUI and once we execute the Kafka producer, GUI with a producer click button will appear when we click on the producer the data from the producer will be transferred to the Kafka server.

Identical to the Kafka producer (source system) we also built a Kafka consumer (target system) to consume data from Kafka server and it is interfaced with GUI and when we execute the Kafka consumer, A GUI with a consumer click button will appear when we click on the consumer, Kafka consumer gets activated and it starts consuming data from the Kafka server.

So, whenever the data is transferred from the producer it is transferred to Kafka server and it is consumed consumer.



5.2 Client and Kafka Running on The Different Machines

To connect to the Kafka cluster and the client running on different machines both should be running on the same network

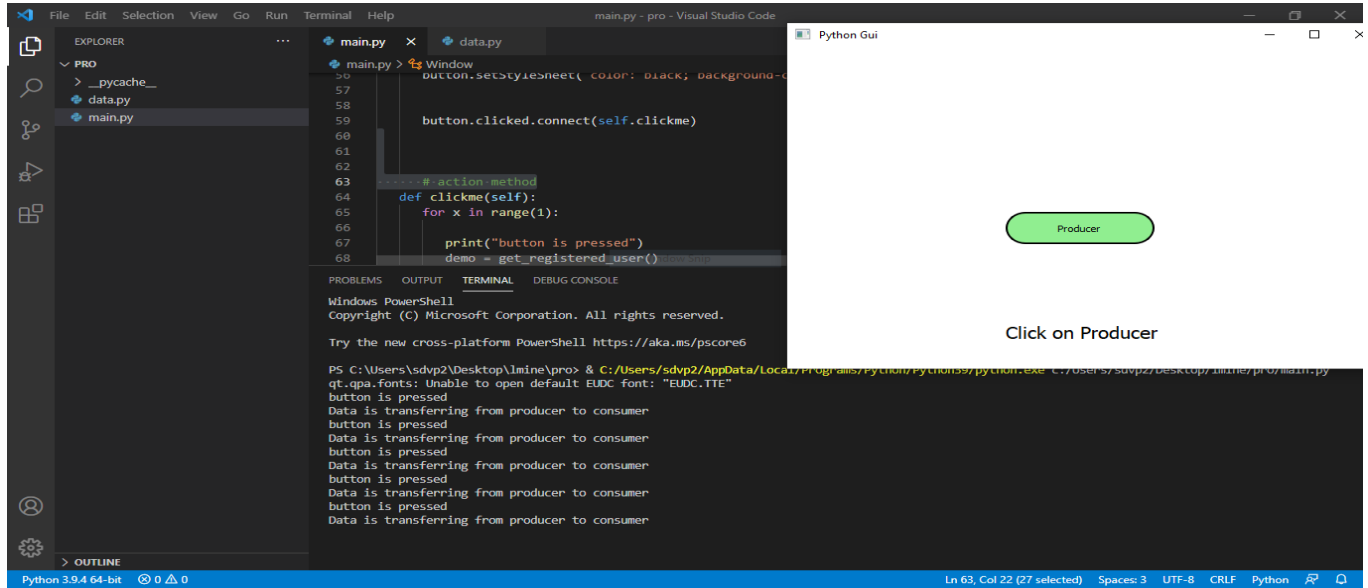
Implementation of client and Kafka on different server:

Both Kafka producer and Kafka server will be run in one system and

On the other hand, Kafka consumer(client) will be running on different system.

When the Kafka producer is executed in one machine the output from the producer will be consumed by Kafka consumer(client) in another machine.

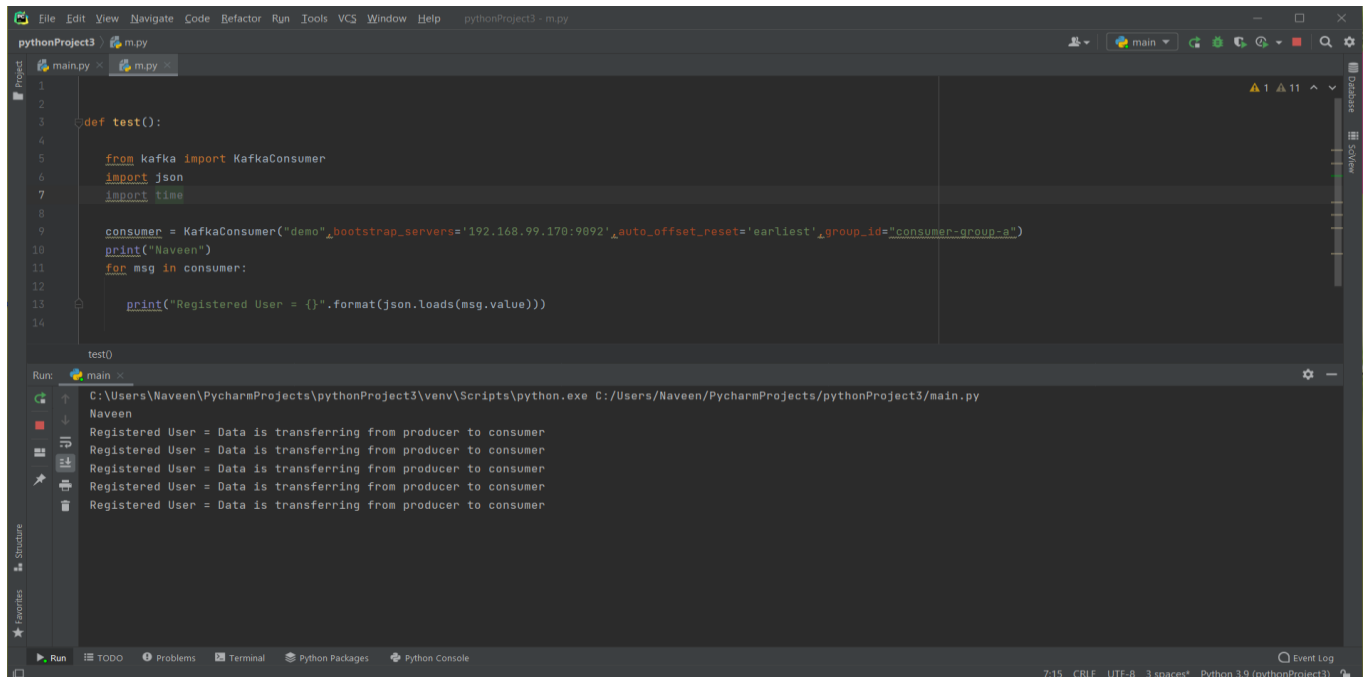
5.2.1 Output of Kafka Producer on different machine



The screenshot shows a Visual Studio Code editor with a Python GUI application and its terminal output. The GUI has a green button labeled "Producer" and a text label "Click on Producer". The terminal output shows the following sequence of events:

```
PS C:\Users\sdvp2\Desktop\lmine\pro> & C:\Users\sdvp2\AppData\Local\Programs\Python\Python39\python.exe C:\Users\sdvp2\Desktop\lmine\pro\main.py
qt.qpa.fonts: Unable to open default EUDC font: "EUDC.TTE"
button is pressed
Data is transferring from producer to consumer
button is pressed
Data is transferring from producer to consumer
button is pressed
Data is transferring from producer to consumer
button is pressed
Data is transferring from producer to consumer
button is pressed
Data is transferring from producer to consumer
```

5.2.2 Output of Kafka consumer on different machine



The screenshot shows a PyCharm editor with a Python script and its terminal output. The script defines a test function that creates a Kafka consumer and prints the registered user. The terminal output shows the following sequence of events:

```
C:\Users\Naveen\PycharmProjects\pythonProject3\venv\Scripts\python.exe C:/Users/Naveen/PycharmProjects/pythonProject3/main.py
Naveen
Registered User = Data is transferring from producer to consumer
Registered User = Data is transferring from producer to consumer
Registered User = Data is transferring from producer to consumer
Registered User = Data is transferring from producer to consumer
Registered User = Data is transferring from producer to consumer
```

6.Project outcomes

- Our final aim of the project is achieved
- A separate GUI for both source system and target system is built and interfaced with Kafka producer and Kafka consumer
- And the data is getting transferred from the Kafka producer to the Kafka consumer through Kafka server
- Both Kafka cluster and client are implemented on the different machines and data is transferring from one machine to the other system.

7.Challenges Faced

- Building GUI
- Setting a Kafka server and solving errors related to producer and consume coding
- Interfacing GUI with and producer, consumer
- Implementing on different machines and transferring data between them.

8.Challenges Overcome

- Referred different websites and videos related to Apache Kafka and building GUI.
- And team work by proper coordination and sharing the work.

9. Contributions

Individual contributions

Nandyala Naveen Kumar Reddy (99003604)

- Worked on GUI and complete setup of Kafka server, coding environment and implementing Kafka client on different machines.

Satyasindhu Thummala (99003534)

- Worked on Kafka architecture, Kafka producer coding, interfacing codes with GUI.

Vikram Hedge (99003597)

- Worked on Kafka consumer coding, interfacing codes with GUI and implementing Kafka cluster on the different machines

Team contribution

- We all together worked on implementation and background work for solving both programming errors and execution errors.

10. Links

GitHub link

https://github.com/99003534/Shadow_Project

SharePoint link

<https://lnttsgroup.sharepoint.com/sites/GEA/Global%20Engineering%20Academy/GENESIS%20%20Jan%202020%20Submission/Forms/AllItems.aspx?ct=1619609898956&or=OWA%2DNT&cid=17469d57%2Df4bb%2De6aa%2Dbaf9%2D53ffc23b062b&originalPath=aHR0cHM6Ly9sbnR0c2dyb3VwLnNoYXJlcG9pbmQuY29tLzpmOi9zL0dFQS9HbG9iYWwlmjBFbmdpbmVlcmluZyUyMEFjYWRLbXkvRWWhKZTZBZzFGcmhCZ2hFczVpLXJ3b3dCR1JNU0FPcm54ZVNLX3EyVks3QmxLUT9ydGltZT1QNkdrSmpvSzJVZw&viewid=c6456521%2D0829%2D4568%2Db8d1%2D27a784675a0f&id=%2Fsites%2FGEA%2FGlobal%20Engineering%20Academy%2FGENESIS%20%20Jan%202020%20Submission%2FSubmission%2FMYSORE%2F2002MYSEMB%5FF1%2F99003534%2FShadow%20Project>

