SUMMER INTERNSHIP
On
(DATA SCIENCE AND MACHINE LEARNING)

Submitted by

Name-Somnath Rana
Registration No-12324985
Program Name-MCA

School of Computer Application
Lovely Professional University, Phagwara

[June -July 2024]

Acknowledgement

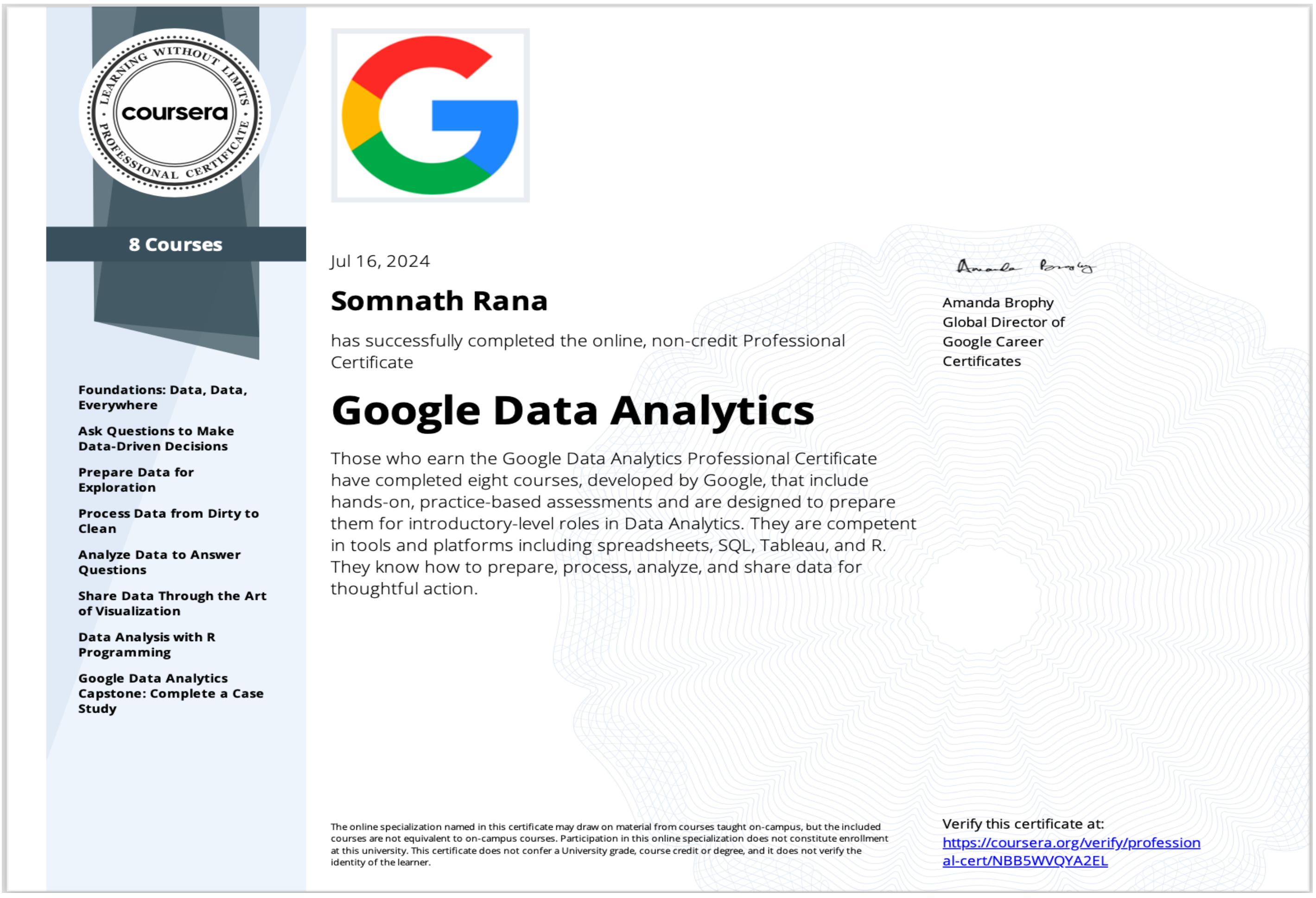
The internship opportunity with Coursera was a great chance for learning and professional development. Therefore, I consider myself a very lucky individual as I was given an opportunity to be a part of it. I am also grateful for having a chance to learn from professionals who led me through this internship period.

I express my deepest thanks to the Training and Placement Coordinator, School of Computer Application, Lovely Professional University for allowing me to grab this opportunity. I choose this moment to acknowledge his contribution gratefully by giving the necessary advice and guidance to make my internship a good learning experience.

Somnath Rana.
12324985

MOOC Course Certificate:

The "Google Data Analytics" course is an eight-course professional certification developed by Google and offered on Coursera.



LEARNING WITHOUT LIMITS · coursera · PROFESSIONAL CERTIFICATE

8 Courses

Foundations: Data, Data, Everywhere

Ask Questions to Make Data-Driven Decisions

Prepare Data for Exploration

Process Data from Dirty to Clean

Analyze Data to Answer Questions

Share Data Through the Art of Visualization

Data Analysis with R Programming

Google Data Analytics Capstone: Complete a Case Study

Jul 16, 2024

Somnath Rana

has successfully completed the online, non-credit Professional Certificate

Google Data Analytics

Those who earn the Google Data Analytics Professional Certificate have completed eight courses, developed by Google, that include hands-on, practice-based assessments and are designed to prepare them for introductory-level roles in Data Analytics. They are competent in tools and platforms including spreadsheets, SQL, Tableau, and R. They know how to prepare, process, analyze, and share data for thoughtful action.

[illegible]

Amanda Brophy
Global Director of Google Career Certificates

The online specialization named in this certificate may draw on material from courses taught on-campus, but the included courses are not equivalent to on-campus courses. Participation in this online specialization does not constitute enrollment at this university. This certificate does not confer a University grade, course credit or degree, and it does not verify the identity of the learner.

Verify this certificate at:
https://coursera.org/verify/professional-cert/NBB5WVQYA2EL

Table of contents

Introduction of the Course -

1. Overview of Data Science –

- What is Data Science?

Analysing and manipulating the data(text, image, audio or video), so that it can predict the further used cases and recommend reasonable suggestions accordingly through some recommendation systems.

- **Foundations: Data, Data, Everywhere** - An introduction to data analytics and its importance in the modern world.
- **Ask Questions to Make Data-Driven Decisions** - Developing the ability to ask the right questions to guide decision-making processes.
- **Prepare Data for Exploration** - Understanding how to prepare data for analysis.
- **Process Data from Dirty to Clean** - Learning techniques to clean and organize data.
- **Analyse Data to Answer Questions** - Using analytical techniques to derive insights from data.
- **Share Data Through the Art of Visualization** - Communicating data insights using visualization tools.
- **Data Analysis with R Programming** - Applying R programming to data analytics tasks.
- **Google Data Analytics Capstone: Complete a Case Study** - Putting all the learned skills into practice through a capstone project.

The program covers various tools and platforms such as spreadsheets, SQL, Tableau, and R, making graduates proficient in preparing, processing, analysing, and sharing data effectively.

Note:- That this certification is designed as an introduction to the field and is aimed at providing the foundational skills necessary for entry-level roles in data analytics.

2. Life Cycle of Data Science –

1. Understanding The Problem
2. Gathering Relevant Data
3. Data Preparation & EDA (Data Analysis)
4. Feature Engineering & Feature Extraction
5. Model Building & Deployment

1. What is Understanding the Problem

• Define the objectives clearly.
• Determine how to measure the project's success.
• Identify the problem type (e.g., classification, regression,)

2. What is Gathering Relevant Data

• Collect data from other sources.
• Ensure data is relevant to the problem.
• Consider the volume, variety, velocity, and veracity of the data.

3. what is Data Preparation & EDA

• EDA stands for Exploratory Data Analysis
• Clean the data (handle missing values, remove duplicates) from the data. • Perform exploratory analysis to understand the data.
• So that the data should be prepared for error

4. Feature Engineering & Feature Extraction

• We need to select exactly the feature which we want to select
• Create new features that can help improve
model performance.
• Reduce dimensionality if the feature space is too large.
• Select the most important features to be used for modelling.

5. Model Building & Deployment

• Choose appropriate algorithms and train models.
• Validate model performance using cross-validation.
• Deploy the model for real-time use or batch processing.

3. Major Roles in Data Domain –

1. Data Analyst
Skills: SQL, Excel, data visualization, basic statistical knowledge.

Data Scientist
Skills: Programming (Python/R), machine learning, advanced statistics, data wrangling.

Machine Learning Engineer
Skills: Deep learning, algorithm optimization, programming, software engineering.

4. Data Engineer
Skills: Database management, ETL processes, big data technologies, data pipelines.

Business Intelligence Analyst
Skills: Data analysis, BI tools (Tableau, Power BI), business knowledge.

Programming Languages:

1. Python
2. R Programming

Tools:

1. Anaconda
2. Google Colab/Jupyter
3. Git Version Control

Technical Learnings from the Google Data Analytics Professional Certificate-

During the completion of the Google Data Analytics Professional Certificate, key technical skills and concepts were acquired, including:

Programming and Software Tools

- **Languages:** Developed proficiency in Python and R for data manipulation, analysis, and model development.
- **Libraries and Frameworks:** Gained experience with essential libraries, including NumPy, Pandas, Scikit-Learn, TensorFlow, and Keras, to implement various data science and machine learning tasks.

Data Manipulation and Preprocessing

- **Data Cleaning:** Acquired techniques for handling missing data, removing outliers, and transforming raw data into structured formats.
- **Data Integration:** Mastered the merging and reshaping of datasets to prepare them for analysis.

Exploratory Data Analysis (EDA)

- **Visualization Techniques:** Learned to create insightful visualizations using tools such as Matplotlib, Seaborn, and Plotly to uncover patterns and trends in data.
- **Statistical Analysis:** Applied descriptive statistics to understand data distributions and relationships between variables.

Machine Learning Algorithms

- **Supervised Learning:** Implemented algorithms like Linear Regression, Decision Trees, and Random Forests to predict outcomes based on labelled data.
- **Unsupervised Learning:** Applied clustering techniques such as K-Means and Hierarchical Clustering to identify patterns in unlabelled data.
- **Deep Learning:** Developed an understanding of neural networks and their application to complex tasks such as image and text processing.

Model Evaluation and Optimization

- **Performance Metrics:** Evaluated models using metrics such as accuracy, precision, recall, and F1-score.

- **Cross-Validation and Tuning:** Implemented K-Fold Cross-Validation and performed hyperparameter tuning to optimize model performance.

Big Data and Data Engineering

- **Data Management:** Gained knowledge in using SQL and NoSQL databases for efficient data retrieval and storage.
- **Big Data Tools:** Acquired foundational knowledge of tools like Hadoop and Spark for managing and processing large datasets.
- **Data Pipelines:** Learned about the ETL (Extract, Transform, Load) process and automation of data workflows.

Time Series Analysis

- **Forecasting Models:** Applied ARIMA and LSTM models for time series forecasting.
- **Trend and Seasonality:** Analysed and adjusted for trends and seasonality in time series data.

Natural Language Processing (NLP)

- **Text Processing:** Gained experience with techniques such as tokenization and sentiment analysis.
- **Language Models:** Explored the use of advanced language models like BERT for text understanding and generation.

Model Deployment

- **Deployment Techniques:** Developed experience in deploying machine learning models using tools like Flask, Docker, and Kubernetes.
- **MLOps:** Understood the principles of MLOps for maintaining and scaling models in production environments

Ethics in AI

- **Bias and Fairness:** Learned methods for identifying and mitigating biases in machine learning models.
- **Data Privacy:** Considered the ethical implications and data privacy regulations, such as GDPR, during the development process.

▪ *About Anaconda – Introduction:*

- **Anaconda** is a distribution of the Python and R programming languages designed to simplify package management and deployment in data science and machine learning applications.
- It was created by **Anaconda, Inc.** (formerly known as Continuum Analytics).

Why Use Anaconda?

- **Simplified Package Management:** Easily manage Python and R packages using Conda, Anaconda's package manager.
- **Comprehensive Distribution:** Comes pre-installed with many data science libraries and tools, reducing the need to install them individually.
- **Environment Management:** Easily create and manage isolated environments for different projects.
- **Cross-Platform Compatibility:** Works across various operating systems (Windows, macOS, Linux).
- **Open-Source Tools:** Anaconda is free and open-source, offering a vast ecosystem of tools for data science.
- **Ease of Use:** Simplifies the setup and configuration process for data science and machine learning projects.

Environment in Anaconda:

What is an Environment?

- An environment is a collection of packages, primarily libraries, used for a specific project or purpose. It allows you to keep dependencies isolated and manageable.

Why Use Environments?

- **Maintain Multiple Versions:** Easily manage different versions of Python and packages for different projects.
- **Isolate Projects:** Keep your project dependencies separate to avoid conflicts between different projects.
- **Reproducibility:** Environments ensure that others can recreate your exact setup to achieve the same results.

2. *Data Cleaning and Preparation Techniques:*

1. **TRIM:** Removes all extra spaces from a string.
    - **Syntax:** ’=TRIM(cell_name)’

2. **PROPER:** Converts each word in a string to start with an uppercase letter.
    - **Syntax:** ’=PROPER(cell_name)’

3. **UPPER:** Converts all characters in a string to uppercase.
    - **Syntax:** ’=UPPER(cell_name)’

4. **LOWER:** Converts all characters in a string to lowercase.
    - **Syntax:** ‘=LOWER(cell_name)’

3. *Text Manipulation Functions:*

1. **LEFT:** Extracts a specified number of characters from the start of a string.
    - **Syntax:** ‘=LEFT(cell_name, number_of_characters)’

2. **MID:** Returns a specified number of characters from a text string starting at any position.
    - **Syntax:’** =MID(cell_name, start_position, number_of_characters)’

3. **SEARCH:** Searches for a specific character or set of characters in a string.
    - **Syntax:** ‘=SEARCH("text_to_find", cell_name)’

4. **LEN:** Returns the length of a text string (number of characters).
    - **Syntax:** ‘=LEN(cell_name)’

5. **RIGHT:** Extracts a specified number of characters from the end of a text string.
    - **Syntax:’** =RIGHT(cell_name, number_of_characters)’

4. *Text Combining:*

1. **Concatenation:** Combines two or more text strings into one.
    - **Syntax:’** =CONCATENATE(cell_name1, cell_name2, ...)’

2. **TEXTJOIN:** Combines text from multiple cells with a specified delimiter.
    - **Syntax:** ‘=TEXTJOIN("delimiter", TRUE/FALSE, cell_range)’

5. *Conditional and Logical Functions:*

1. **IF:** Tests a condition and returns one value if true and another if false.
    - **Syntax:** ‘=IF(condition, value_if_true, value_if_false)’

2. **IFERROR:** Returns a specified value if a formula evaluates to an error, otherwise returns the result of the formula.
    - **Syntax:** ‘=IFERROR(value, value_if_error)’

Introduction of Project –

The project you have outlined is a comprehensive IoT-based health monitoring system. It involves simulating health data, ingesting this data in real time, analysing it for anomalies using machine learning, and triggering alerts if anomalies are detected.

1. **Health Data Simulation:-**

- **Module: simulate_iot_data**
- **Purpose**: Simulate real-time health data such as heart rate, body temperature, and oxygen levels.

- **Functionality**: This module uses a random generator to produce data that mimics the vital signs of a person. The simulated data is then published to an MQTT topic (health/data) using the MQTT protocol.

2. Data Ingestion and Visualization
- **Module: data_ingestion**
- **Purpose**: Ingest the simulated health data from the MQTT broker, store it in memory, and visualize it using real-time plotting.
- **Functionality**: This module subscribes to the same MQTT topic (health/data) and processes incoming data. It extracts heart rate, temperature, and oxygen levels and appends them to corresponding lists. Matplotlib is used for real-time visualization of this data.

3. Anomaly Detection and Alerting
- **Module: anomaly_detection (within the ingestion module)**
- **Purpose**: Detect anomalies in the ingested health data and send alerts if abnormal patterns are detected.
- **Functionality**: The system uses the Isolation Forest algorithm from Scikit-learn to detect outliers in the health data. If an anomaly is detected, the system triggers an alert via the Twilio API, sending a notification to a predefined phone number.

4. User Dashboard
- **Module: Dashboard**
- **Purpose**: Provide an interactive user interface for controlling the system.
- **Functionality**: The dashboard allows users to simulate health data, start data ingestion, or exit the system. The interface interacts with the other modules, enabling users to initiate health data generation or begin monitoring and visualization.

5. Main Application

- **Module: main**
- **Purpose**: Serve as the entry point for the entire system, guiding the user through the different functionalities of the project.
- **Functionality**: The main function provides the user with a menu to either simulate health data, start data ingestion or exit the application.

Project Goals

- **Real-time Data Simulation**: Generate real-time health-related data that could simulate a real patient's vital signs.

- **Data Visualization**: Display incoming data in real-time plots to provide insight into the trends and behaviour of the health parameters.

- **Anomaly Detection**: Implement machine learning models to detect outliers in the health data and take appropriate action.

- **Alert System**: Send notifications when abnormal health patterns are detected, making it suitable for remote monitoring in health care.

Applications

- **Healthcare Monitoring**: This system can be used in healthcare settings to monitor patients' vitals in real-time.

- **Remote Patient Monitoring**: Particularly useful for telemedicine or home care, where a healthcare professional needs to monitor patients remotely.

- **IoT and Health Analytics**: The project demonstrates how IoT devices and cloud-based communication protocols like MQTT can be leveraged for health analytics.

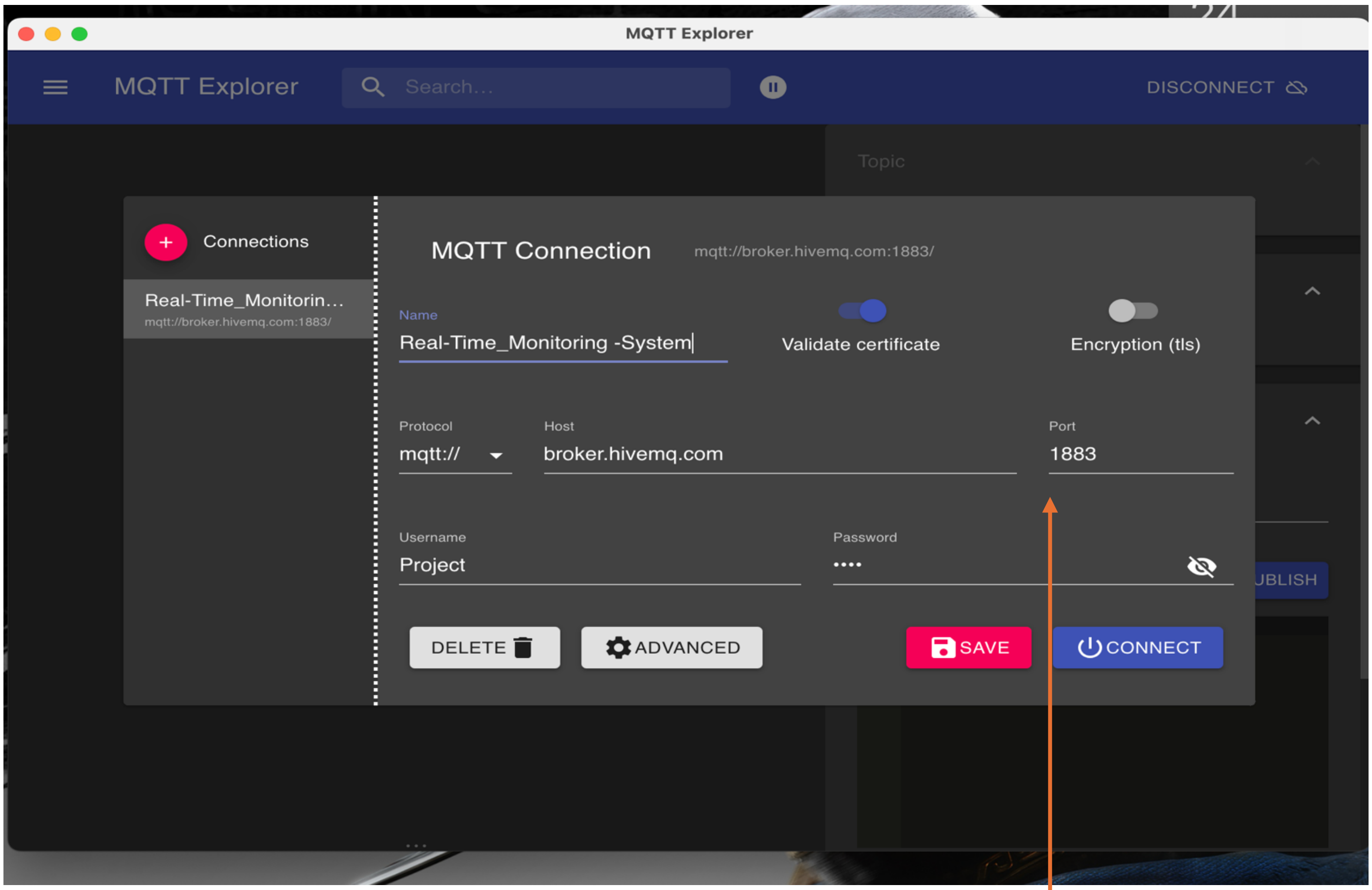
Future Improvements

- **Data Persistence**: Implement a database to store historical health data for long-term analysis.

- **Advanced Anomaly Detection**: Improve the machine learning model by incorporating more sophisticated techniques or domain-specific rules.

- **User Interface**: Enhance the dashboard with a GUI, making it more user-friendly.

- **Connection to Real-Time Monitor**


MQTT Explorer

MQTT Connection mqtt://broker.hivemq.com:1883/

Name: Real-Time_Monitoring -System

Validate certificate

Encryption (tls)

Protocol: mqtt://

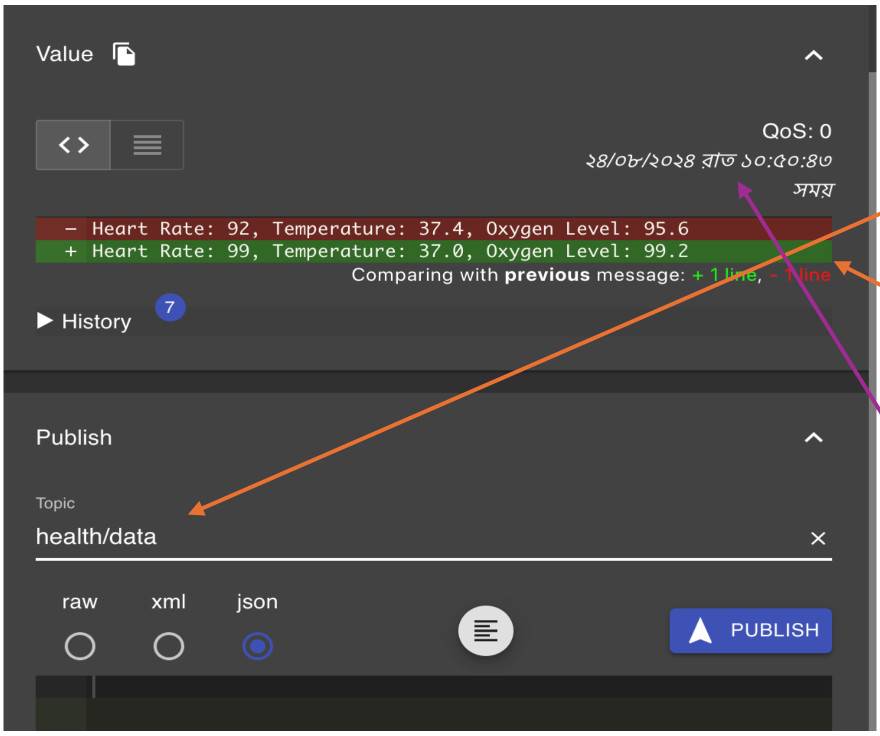
Host: broker.hivemq.com

Port: 1883

Username: Project

Password: ••••

1. First, connect the PyCharm to the MQTT explorer set with the **Port number and Host Name.**
2. After connection set a Data Monitoring Name **(health/data**) and set a Time for view.


Value
QoS: 0
২৪/০৮/২০২৪ রাত ১০:৫০:৪৩ সময়
History
7
Publish
Topic
raw
xml
json
PUBLISH
Publish a real-time View data
Real-time Monitoring
Set a time for viewing data All reload 5sec

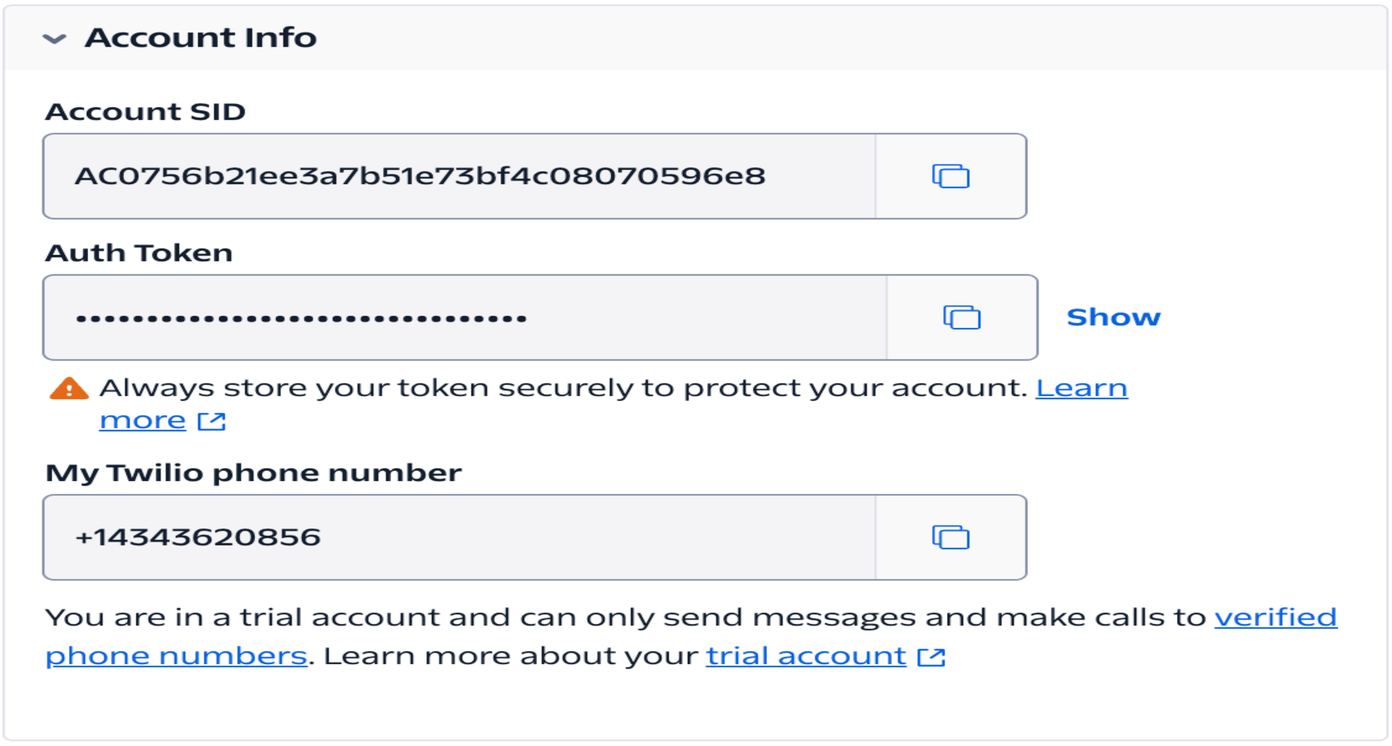
```
- Heart Rate: 92, Temperature: 37.4, Oxygen Level: 95.6
+ Heart Rate: 99, Temperature: 37.0, Oxygen Level: 99.2
```

Comparing with **previous** message: + 1 line, - 1 line

Topic: health/data

3. Set the Real-time alert – SMS System –

1. Create a Twilio A/c Set All details and fill in your Alert SMS PH. No [ex- (+91 974952037*)]



Account SID

AC0756b21ee3a7b51e73bf4c08070596e8

Auth Token

••••••••••••••••••••••••••••••••

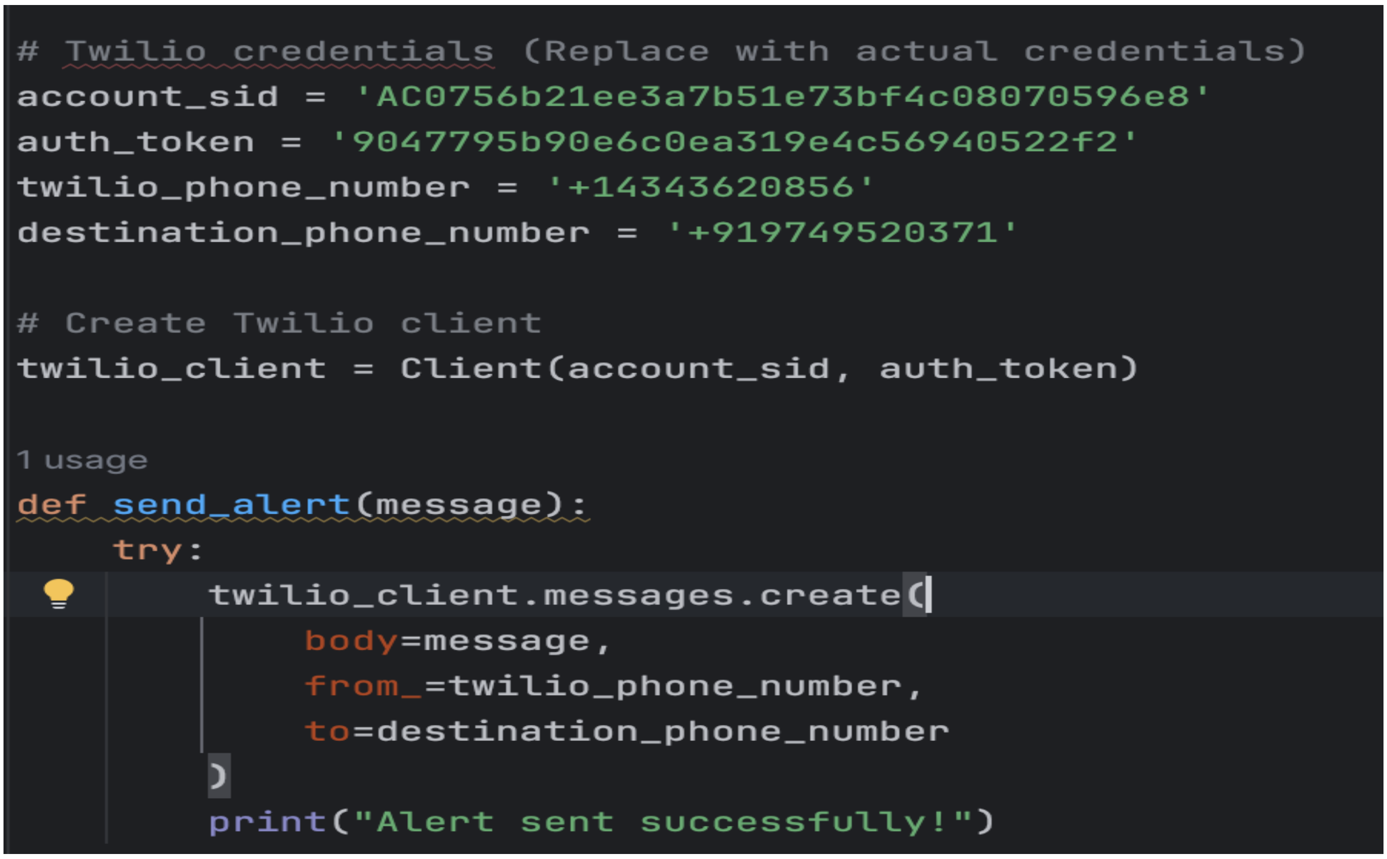
Always store your token securely to protect your account. Learn more

My Twilio phone number

+14343620856

You are in a trial account and can only send messages and make calls to verified phone numbers. Learn more about your trial account

NOTE- A/C SID / Auth Token / SMS send Twilio Number to fill the your code

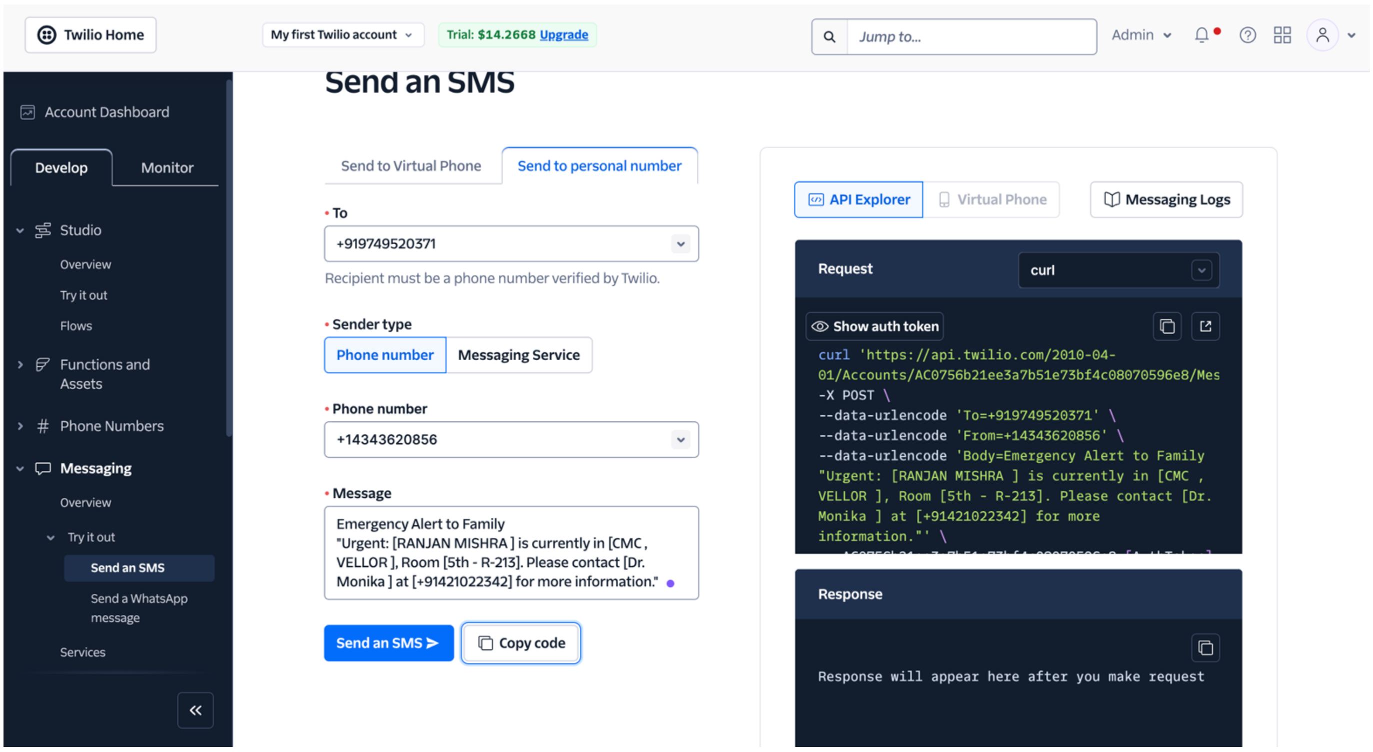


```
# Twilio credentials (Replace with actual credentials)
account_sid = 'AC0756b21ee3a7b51e73bf4c08070596e8'
auth_token = '9047795b90e6c0ea319e4c56940522f2'
twilio_phone_number = '+14343620856'
destination_phone_number = '+919749520371'

# Create Twilio client
twilio_client = Client(account_sid, auth_token)

1 usage
def send_alert(message):
    try:
        twilio_client.messages.create(
            body=message,
            from_=twilio_phone_number,
            to=destination_phone_number
        )
        print("Alert sent successfully!")
```

4. Create an SMS Alert CODE AND Paste the **anomaly.py** path:-



Send an SMS

Send to Virtual Phone | Send to personal number

To: +919749520371

Recipient must be a phone number verified by Twilio.

Sender type: [✓ Phone number] Messaging Service

Phone number: +14343620856

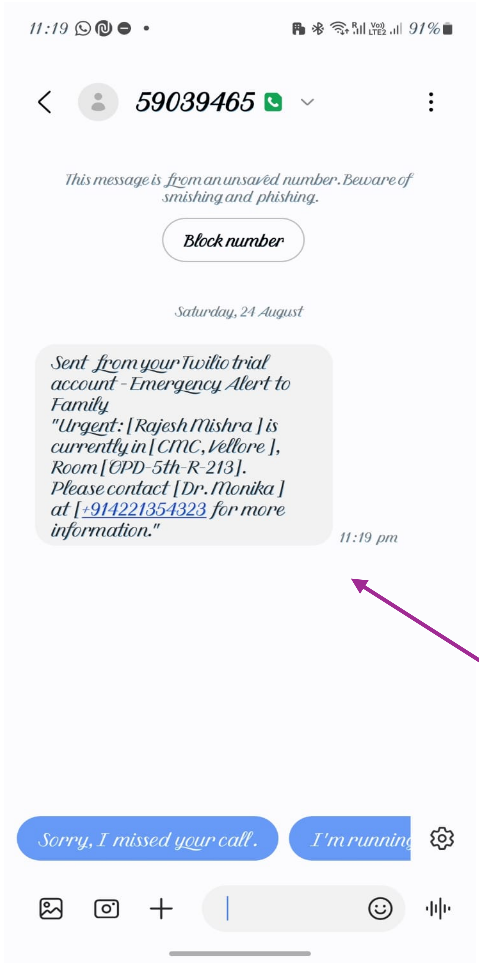
Message:
Emergency Alert to Family
"Urgent: [RANJAN MISHRA] is currently in [CMC , VELLOR], Room [5th - R-213]. Please contact [Dr. Monika] at [+91421022342] for more information."

```
curl 'https://api.twilio.com/2010-04-
01/Accounts/AC0756b21ee3a7b51e73bf4c08070596e8/Mes
-X POST \
--data-urlencode 'To=+919749520371' \
--data-urlencode 'From=+14343620856' \
--data-urlencode 'Body=Emergency Alert to Family
"Urgent: [RANJAN MISHRA ] is currently in [CMC ,
VELLOR ], Room [5th - R-213]. Please contact [Dr.
Monika ] at [+91421022342] for more
information."' \
```

Response

```
Response will appear here after you make request
```

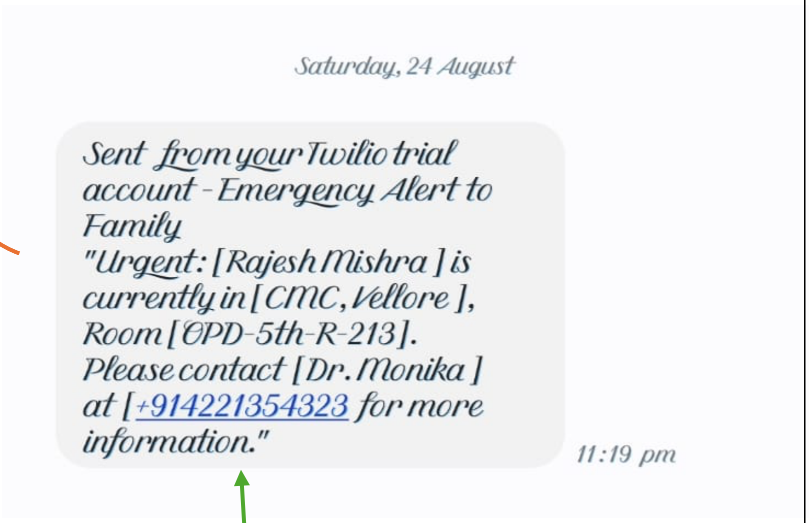
5. **Alert message Example for Patient condition:-**


11:19
91%
59039465
This message is from an unsaved number. Beware of smishing and phishing.
Block number
Saturday, 24 August
Sorry, I missed your call.
I'm running

Sent from your Twilio trial account - Emergency Alert to Family
"Urgent: [Rajesh Mishra] is currently in [CMC, Vellore], Room [OPD-5th-R-213]. Please contact [Dr. Monika] at [+914221354323 for more information."

11:19 pm

- *Depending on the Message of the patient's Condition*



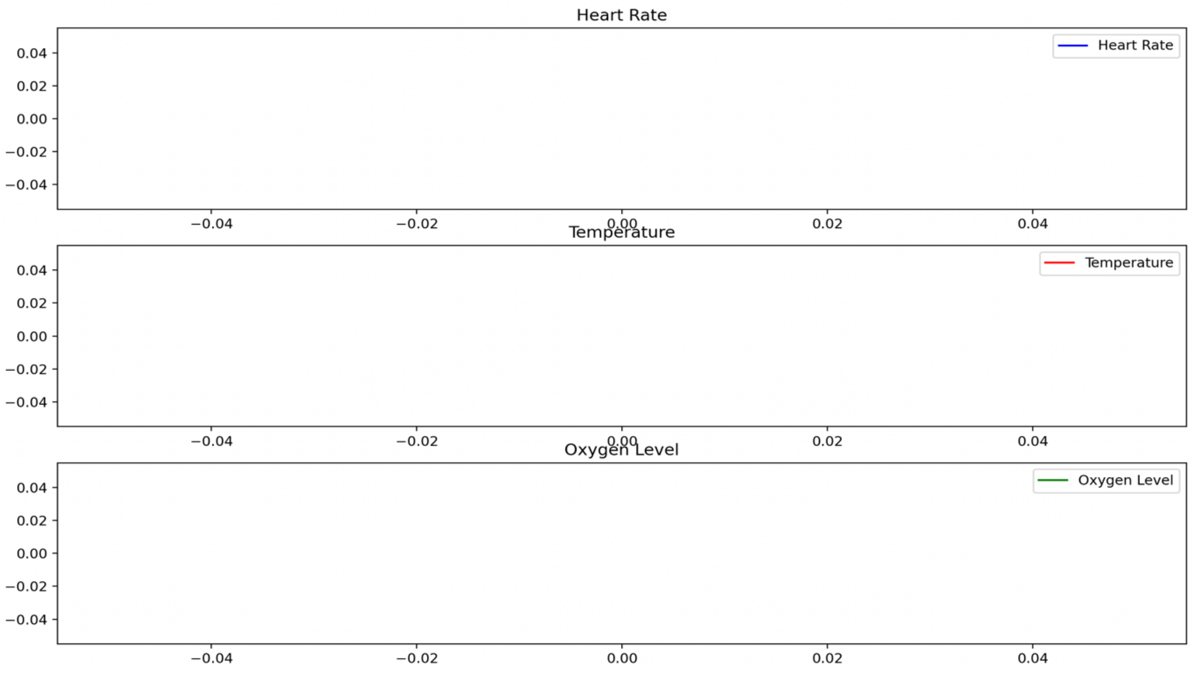
Sent from your Twilio trial account - Emergency Alert to Family
"Urgent: [Rajesh Mishra] is currently in [CMC, Vellore], Room [OPD-5th-R-213]. Please contact [Dr. Monika] at [+914221354323 for more information."

11:19 pm

Mobile SMS alert From Hospital

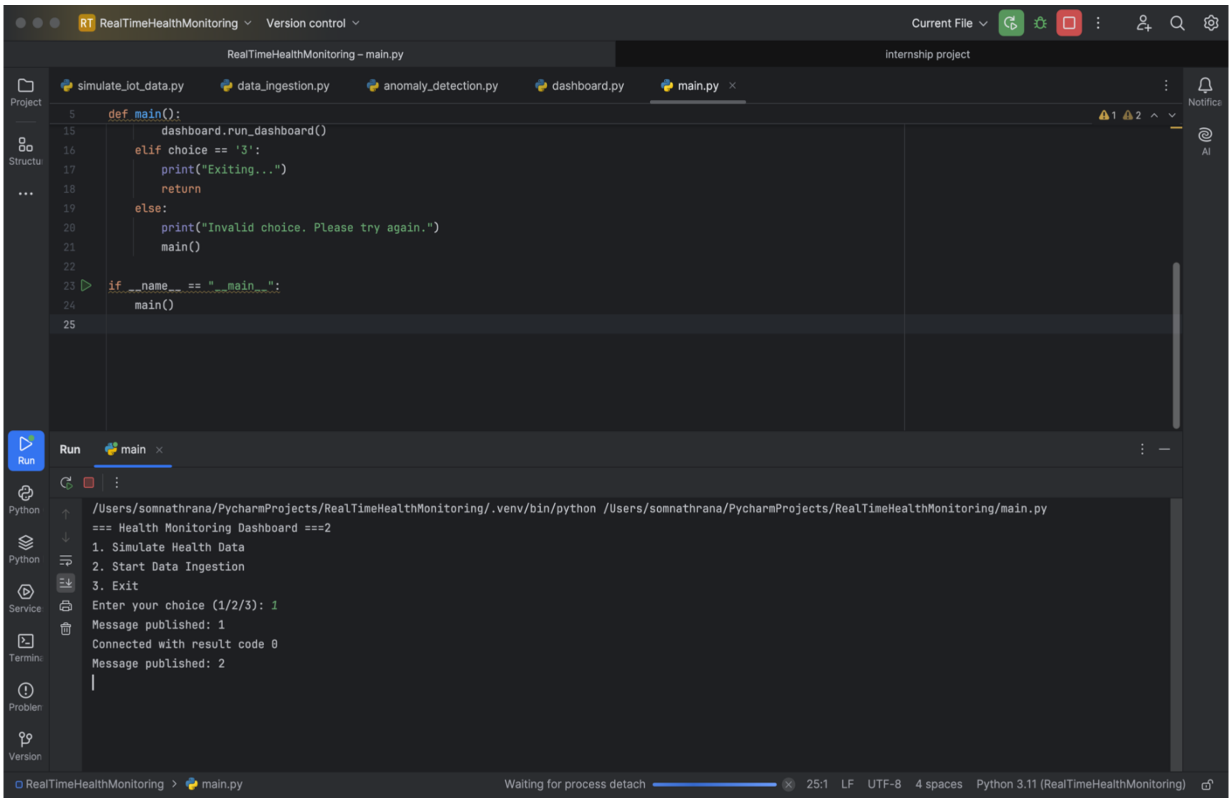
- **IoT Connection-based Screen Out-Put:-**

Only showing the live plots/graphs of the Human body connect



- **Real-time Monitoring Screen-**

1st Start a code and press 1 to connect the MQTT Explorer for Real-Time Monitoring Message.

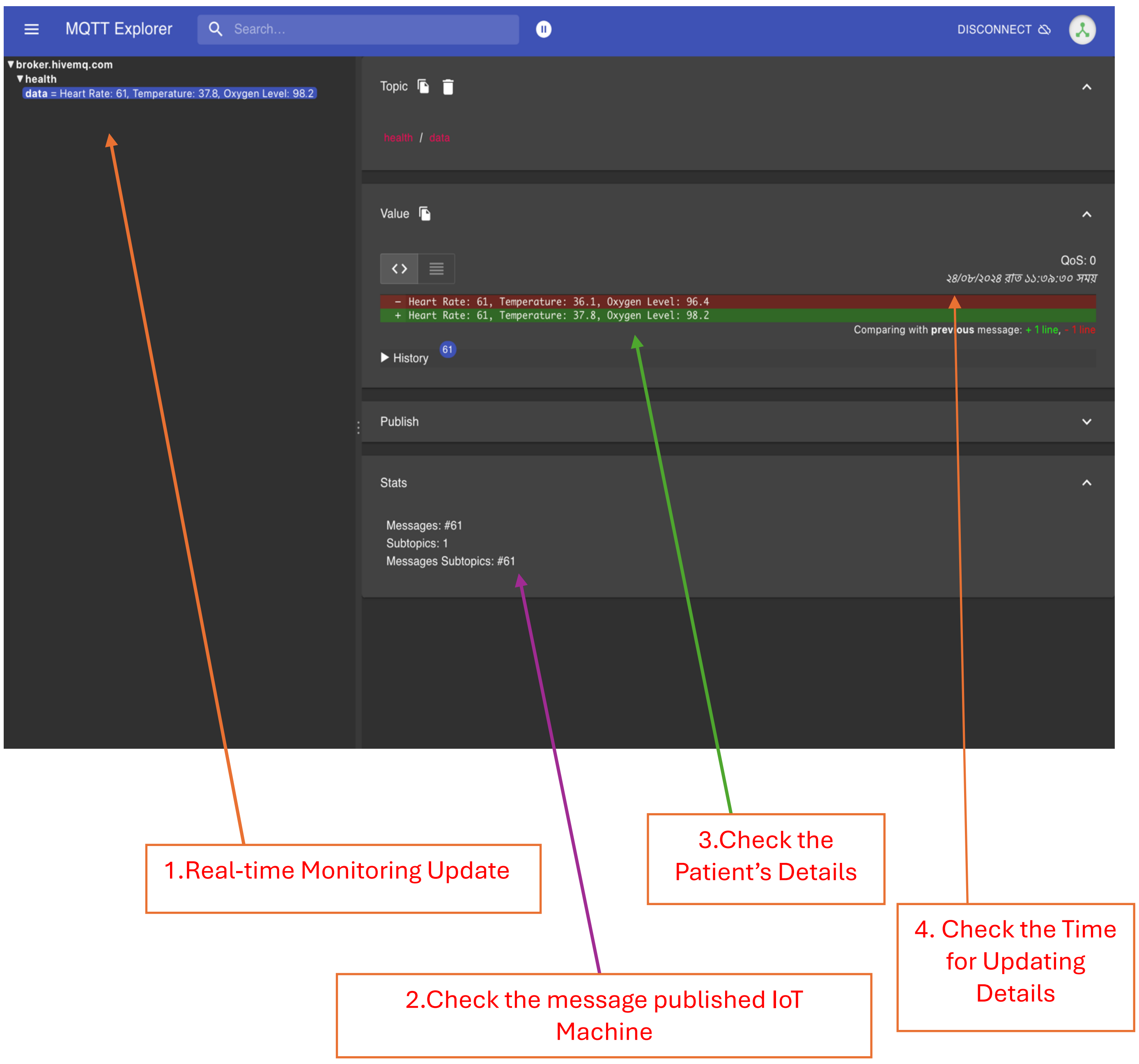


```python
def main():
        dashboard.run_dashboard()
    elif choice == '3':
        print("Exiting...")
        return
    else:
        print("Invalid choice. Please try again.")
        main()

if __name__ == "__main__":
    main()
```

```
/Users/somnathrana/PycharmProjects/RealTimeHealthMonitoring/.venv/bin/python /Users/somnathrana/PycharmProjects/RealTimeHealthMonitoring/main.py
=== Health Monitoring Dashboard ===2
1. Simulate Health Data
2. Start Data Ingestion
3. Exit
Enter your choice (1/2/3): 1
Message published: 1
Connected with result code 0
Message published: 2
```

2nd Open and Post and Hostname connection The System for Real-Time Monitoring.


MQTT Explorer
Search...
DISCONNECT
Topic
Value
QoS: 0
History
61
Publish
Stats
1.Real-time Monitoring Update
3.Check the Patient's Details
4. Check the Time for Updating Details
2.Check the message published IoT Machine

▾ broker.hivemq.com
▾ health
data = Heart Rate: 61, Temperature: 37.8, Oxygen Level: 98.2

health / data

২৪/০৮/২০২৪ রাত ১১:৩৯:৩০ সময়

```
 − Heart Rate: 61, Temperature: 36.1, Oxygen Level: 96.4
 + Heart Rate: 61, Temperature: 37.8, Oxygen Level: 98.2
```

Comparing with previous message: + 1 line, - 1 line

Messages: #61
Subtopics: 1
Messages Subtopics: #61

Code Snippet –

▪ *Simulate_iot_data.py*

```
import paho.mqtt.client as mqtt
import time
import random

BROKER = "broker.hivemq.com"
PORT = 1883
TOPIC = "health/data"

def on_connect(client, userdata, flags, rc):
  print(f"Connected with result code {rc}")

def on_publish(client, userdata, mid):
  print(f"Message published: {mid}")

def simulate_health_data():
  client = mqtt.Client("HealthSimulator")
  client.on_connect = on_connect
  client.on_publish = on_publish

  client.connect(BROKER, PORT, 60)
  client.loop_start()

  try:
    while True:
      heart_rate = random.randint(60, 100)
      temperature = round(random.uniform(36.0, 38.5), 1)
      oxygen_level = round(random.uniform(95.0, 100.0), 1)

      message = f"Heart Rate: {heart_rate}, Temperature: {temperature}, Oxygen
Level: {oxygen_level}"
      client.publish(TOPIC, message)

      time.sleep(5)  # Publish every 5 seconds

  except KeyboardInterrupt:
    print("Simulation interrupted by user.")
  finally:
    client.loop_stop()
    client.disconnect()
    print("Disconnected from broker.")

if __name__ == "__main__":
  simulate_health_data()
```

- ***data_ingestion.py***

```
import paho.mqtt.client as mqtt
import matplotlib.pyplot as plt
import matplotlib.animation as animation

BROKER = "broker.hivemq.com"
TOPIC = "health/data"

# Global variables for storing data
data = {'heart_rate': [], 'temperature': [], 'oxygen_level': []}

def on_connect(client, userdata, flags, rc):
  print(f"Connected with result code {rc}")
  client.subscribe(TOPIC)

def on_message(client, userdata, message):
  try:
    data_str = message.payload.decode("utf-8")
    parts = data_str.split(', ')
    heart_rate = float(parts[0].split(': ')[1])
    temperature = float(parts[1].split(': ')[1])
    oxygen_level = float(parts[2].split(': ')[1])

    # Append data to global lists
    data['heart_rate'].append(heart_rate)
    data['temperature'].append(temperature)
    data['oxygen_level'].append(oxygen_level)

    # Limit the length of data lists to keep the plot responsive
    max_length = 100
    if len(data['heart_rate']) > max_length:
      data['heart_rate'].pop(0)
      data['temperature'].pop(0)
      data['oxygen_level'].pop(0)

  except Exception as e:
    print(f"Error processing message: {e}")

def update_plot(frame):
  plt.clf()
  plt.subplot(3, 1, 1)
  plt.plot(data['heart_rate'], label='Heart Rate', color='blue')
  plt.legend(loc='upper right')
  plt.title("Heart Rate")

  plt.subplot(3, 1, 2)
  plt.plot(data['temperature'], label='Temperature', color='red')
  plt.legend(loc='upper right')
  plt.title("Temperature")

  plt.subplot(3, 1, 3)
  plt.plot(data['oxygen_level'], label='Oxygen Level', color='green')
  plt.legend(loc='upper right')
```

```
    plt.title("Oxygen Level")

def start_ingestion():
    client = mqtt.Client("DataIngestion")
    client.on_connect = on_connect
    client.on_message = on_message

    client.connect(BROKER)
    client.loop_start()

    # Set up matplotlib for real-time plotting
    fig = plt.figure()
    ani = animation.FuncAnimation(fig, update_plot, interval=1000)
    plt.tight_layout()
    plt.show()

    try:
        while True:
            pass  # Keep the script running to process incoming messages

    except KeyboardInterrupt:
        print("Data ingestion interrupted by user.")
    finally:
        client.loop_stop()
        client.disconnect()
        print("Disconnected from broker.")

if __name__ == "__main__":
    start_ingestion()
```

- ***anomaly_detection.py***

```
  from sklearn.ensemble import IsolationForest
import numpy as np
from twilio.rest import Client

# Twilio credentials (Replace with actual credentials)
account_sid = 'AC0756b21ee3a7b51e73bf4c08070596e8'
auth_token = '9047795b90e6c0ea319e4c56940522f2'
twilio_phone_number = '+14343620856'
destination_phone_number = '+919749520371'

# Create Twilio client
twilio_client = Client(account_sid, auth_token)

def send_alert(message):
   try:
      twilio_client.messages.create(
         body=message,
         from_=twilio_phone_number,
         to=destination_phone_number
      )
      print("Alert sent successfully!")
   except Exception as e:
      print(f"Error sending alert: {e}")

# Train the model (dummy training data)
X_train = np.array([[70, 36.5, 98], [72, 36.7, 97], [75, 36.4, 99], [80, 36.8, 96]])
model = IsolationForest(contamination=0.1)
model.fit(X_train)

def detect_anomaly(new_data):
   pred = model.predict([new_data])
   if pred == -1:
      anomaly_message = f"Anomaly detected! Data: {new_data}"
      print(anomaly_message)
      send_alert(anomaly_message)
      return "Anomaly Detected"
   else:
      return "Normal"
```

- ***dashboard.py***

```
import os

def run_dashboard():
    os.system('cls' if os.name == 'nt' else 'clear')  # Clear the console

    print("=== Health Monitoring Dashboard ===")
    print("1. Simulate Health Data")
    print("2. Start Data Ingestion")
    print("3. Exit")

    while True:
        choice = input("Enter your choice (1/2/3): ")

        if choice == '1':
            import simulate_iot_data
            simulate_iot_data.simulate_health_data()
        elif choice == '2':
            import data_ingestion
            data_ingestion.start_ingestion()
        elif choice == '3':
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please try again.")
```

- ***Main.py***

```
import data_ingestion
import simulate_iot_data  # Ensure this module is correctly implemented
import dashboard  # Ensure this module is correctly implemented

def main():
    print("=== Health Monitoring Dashboard ===2")
    print("1. Simulate Health Data")
    print("2. Start Data Ingestion")
    print("3. Exit")
    choice = input("Enter your choice (1/2/3): ")

    if choice == '1':
        simulate_iot_data.simulate_health_data()
    elif choice == '2':
        dashboard.run_dashboard()
    elif choice == '3':
        print("Exiting...")
        return
    else:
        print("Invalid choice. Please try again.")
        main()

if __name__ == "__main__":
    main()
```

Grade sheet of assignments/ marks card from the MOOC –

Course Introduction: -Data Science and Machine Learning or Data Visualization & Cleaning
Platform: Coursera
Google Mooc Duration:- **8week completion**
Date: **July-28-2024**
Instructor(s):**Amanda Brophy**

Assignment mark Obtained for 8 Course:-

Assignment/Module	Max Mark	Obtained Mark %	Remark
Foundations: Data, Data, Everywhere	100	91.37%	Excellent
Ask Questions to Make Data-Driven Decisions	100	94.37%	Outstanding
Prepare Data for Exploration	100	86.03%	Very Good
Process Data from Dirty to Clean	100	88.62%	Very Good
Analyze Data to Answer Questions	100	92.62%	Excellent
Share Data Through the Art of Visualization	100	89.76%	Very Good
Data Analysis with R Programming	100	92.75%	Excellent
Google Data Analytics Capstone: Complete a Case Study	100	100%	Outstanding

References-

1. **McKinney, W. (2017).** *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython* (2nd ed.). O'Reilly Media.
o A practical guide to data analysis in Python, focusing on the Pandas library.
2. **Geron, A. (2019).** *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*
(2nd ed.). O'Reilly Media.
o This book provides a hands-on approach to machine learning, with practical examples using Python libraries.
3. **Bishop, C. M. (2006).** *Pattern Recognition and Machine Learning*. Springer.
o A widely used textbook in the field of machine learning, covering various algorithms and their applications.
4. **Aggarwal, C. C. (2015).** *Data Mining: The Textbook*. Springer.
o An in-depth resource on data mining techniques, including clustering, classification, and anomaly detection.
5. **Provost, F., & Fawcett, T. (2013).** *Data Science for Business: What You Need to Know About*
Data Mining and Data-Analytic Thinking. O'Reilly Media.
o A book that bridges the gap between business strategies and data science, emphasizing
the practical applications of data mining.
6. **VanderPlas, J. (2016).** *Python Data Science Handbook: Essential Tools for Working with*
Data. O'Reilly Media.
o A comprehensive guide to the core tools used in the Python data science ecosystem, including NumPy, Pandas, Matplotlib, and Scikit-Learn. **7. Zhou, Z.-H. (2021).** *Machine Learning* (2nd ed.). Springer.
o An accessible textbook that covers both fundamental and advanced topics in machine learning, suitable for beginners and experts alike.
8. Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.
o A thorough exploration of statistical learning techniques, often used as a reference in both academic and professional settings.