Narzędzie do generowania alertów dla systemów komunikacyjnych

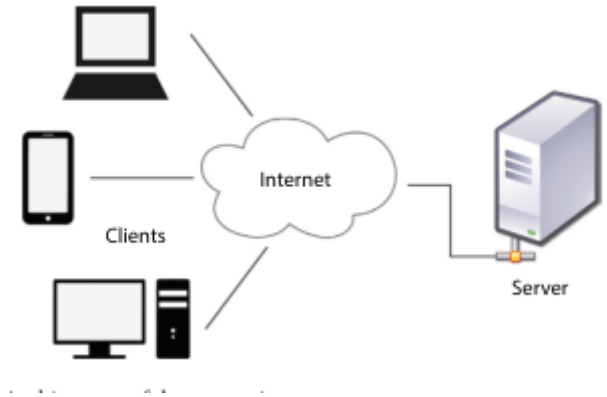
Streszczenie

W dzisiejszym świecie systemy przesyłania wiadomości odgrywają kluczową rolę w różnych branżach, umożliwiając skuteczną komunikację i wymianę informacji. Kluczowe jest jednak zapewnienie niezawodnego dostarczania komunikatów. Aby temu zaradzić, konieczne jest stworzenie narzędzia do generowania alertów dla systemów przesyłania wiadomości. Niniejszy raport przedstawia rozwiązanie oparte na języku Python, które wykorzystuje MySQL, Pandas, Twilio i SMTP do stworzenia niezawodnego systemu generowania alertów. Kod przedstawiony w raporcie demonstruje implementację narzędzia do generowania alertów, prezentując jego możliwości w zakresie monitorowania i analizowania wydajności systemu przesyłania wiadomości. Dzięki integracji standardowych technologii branżowych, rozwiązanie to oferuje skalowalne i elastyczne podejście do różnych scenariuszy z życia wziętych, umożliwiając organizacjom zapewnienie niezawodnego i wydajnego dostarczania wiadomości.

Słowa kluczowe: Systemy przesyłania wiadomości, terminowa komunikacja, alerty, optymalizacja wydajności, API

Wstęp

Systemy przesyłania wiadomości stały się niezbędnym narzędziem komunikacji zarówno dla osób prywatnych, jak i firm w dzisiejszym dynamicznym i połączonym świecie. System przesyłania wiadomości odnosi się do platformy lub infrastruktury, która umożliwia wymianę wiadomości między użytkownikami lub aplikacjami. Zapewnia niezawodny i wydajny sposób przesyłania informacji, umożliwiając użytkownikom wysyłanie i odbieranie wiadomości w różnych formatach, takich jak tekst, multimedia czy komunikacja w czasie rzeczywistym. Aby zapewnić bezbłędne dostarczanie wiadomości, architektura systemu przesyłania wiadomości zazwyczaj składa się z aplikacji klienckich, serwerów i infrastruktury zaplecza .



System przesyłania wiadomości opiera się na architekturze rozproszonej, w której wiadomości są wysyłane i odbierane między aplikacjami klienckimi a serwerami. Infrastruktura po stronie klienta, serwera i obsługi wiadomości to zazwyczaj trzy podstawowe części architektury. Interfejsy użytkownika lub programy, których użytkownicy używają do tworzenia i przeglądania wiadomości, znajdują się po stronie klienta. Różne komponenty po stronie serwera zarządzają dostarczaniem, przechowywaniem i routingiem wiadomości. Sieci, protokoły i systemy bazowe, które obsługują niezawodną transmisję wiadomości, tworzą infrastrukturę przesyłania wiadomości.

Znaczenie alertów w systemie przesyłania wiadomości

Generowanie alertów odgrywa kluczową rolę w zapewnieniu płynnego działania i niezawodności systemów przesyłania wiadomości. Oto kluczowe powody, dla których alerty są ważne:

Monitorowanie stanu systemu

Alerty zapewniają ciągłe monitorowanie stanu i wydajności systemu przesyłania wiadomości. Monitorując różne wskaźniki, takie jak czas reakcji serwera, kolejki wiadomości i wykorzystanie zasobów systemu, alerty mogą natychmiast powiadamiać administratorów lub operatorów o wszelkich potencjalnych problemach lub odchyleniach od normalnego działania.

Proaktywne rozwiązywanie problemów

Alerty umożliwiają proaktywne wykrywanie i rozwiązywanie problemów, zanim się zaostrzą. Dzięki ustaleniu progów i wyzwalaczy, administratorzy mogą otrzymywać alerty w przypadku spełnienia określonych warunków, takich jak wysoki wskaźnik awarii wiadomości, nadmierne opóźnienia lub osiągnięcie limitu pojemności pamięci masowej. Pozwala im to na natychmiastowe podjęcie działań, zapobiegając zakłóceniom w świadczeniu usług i minimalizując przestoje.

Optymalizacja wydajności

Alerty pomagają identyfikować wąskie gardła wydajności w systemie przesyłania wiadomości. Monitorując i analizując wskaźniki, takie jak przepustowość wiadomości, czas przetwarzania czy opóźnienia sieciowe, administratorzy mogą uzyskać wgląd w nieefektywne systemy i odpowiednio zoptymalizować infrastrukturę. Zapewnia to optymalną wydajność systemu i poprawia ogólne wrażenia użytkownika.

Planowanie wydajności

Alerty pomagają w planowaniu wydajności systemów przesyłania wiadomości. Monitorując kolejki wiadomości, obciążenie serwerów i wykorzystanie zasobów, administratorzy mogą przewidywać przyszłe zapotrzebowanie i odpowiednio planować rozbudowę lub optymalizację infrastruktury. Zapewnia to skalowalność i gwarantuje, że system może obsługiwać rosnącą liczbę wiadomości bez obniżania wydajności.

Bezpieczeństwo i zgodność

Alerty odgrywają kluczową rolę w utrzymaniu bezpieczeństwa i zgodności systemów przesyłania wiadomości. Monitorując pod kątem podejrzanych działań, prób nieautoryzowanego dostępu lub potencjalnych naruszeń bezpieczeństwa, alerty mogą powiadamiać administratorów o konieczności natychmiastowego podjęcia działań w celu ograniczenia ryzyka. Ponadto alerty pomagają zapewnić zgodność z przepisami branżowymi i zasadami ochrony danych, zapewniając wgląd w zdarzenia i anomalie systemowe. Podsumowując, alerty są niezbędne do efektywnego działania systemów przesyłania wiadomości. Poprzez ciągłe monitorowanie stanu systemu, proaktywne rozwiązywanie problemów, optymalizację wydajności, planowanie przepustowości oraz zapewnianie bezpieczeństwa i zgodności, alerty przyczyniają się do płynnego i niezawodnego funkcjonowania infrastruktury przesyłania wiadomości. Wdrożenie solidnego narzędzia do generowania alertów jest kluczowe dla administratorów, aby utrzymać stabilność, wydajność i bezpieczeństwo systemów przesyłania wiadomości, zapewniając użytkownikom płynną komunikację.

Monitorowanie wykorzystania procesora w czasie rzeczywistym

Znaczenie monitorowania wykorzystania procesora

Monitorowanie wykorzystania procesora ma kluczowe znaczenie dla utrzymania optymalnej wydajności systemu i zarządzania zasobami. Jednostka centralna (CPU) jest kluczowym elementem każdego systemu komputerowego, odpowiedzialnym za wykonywanie instrukcji i obliczeń. Monitorowanie wykorzystania procesora pozwala administratorom uzyskać wgląd w wydajność systemu, identyfikować potencjalne problemy i podejmować proaktywne działania w celu zapewnienia płynnego działania. Oto kilka kluczowych powodów podkreślających znaczenie monitorowania wykorzystania procesora:

Identyfikacja wąskich gardeł wydajności

Wysokie wykorzystanie procesora może wskazywać na wąskie gardła wydajności [2] w systemie. Monitorując wykorzystanie procesora, administratorzy mogą identyfikować procesy lub aplikacje, które zużywają nadmierne zasoby procesora, powodując spowolnienia lub opóźnienia. Informacje te pomagają w ustaleniu źródła spadku wydajności i podjęciu niezbędnych kroków w celu optymalizacji zasobów systemowych.

Diagnozowanie problemów z wydajnością

Monitorowanie wykorzystania procesora dostarcza cennych informacji na temat wydajności poszczególnych procesów lub aplikacji. Analizując wzorce wykorzystania procesora, administratorzy mogą identyfikować procesy, które monopolizują zasoby procesora, prowadząc do ogólnego spowolnienia systemu. Wiedza ta umożliwia im skuteczniejsze diagnozowanie i rozwiązywanie problemów z wydajnością.

Optymalizacja alokacji zasobów

Efektywna alokacja zasobów jest niezbędna do maksymalizacji wydajności systemu. Monitorując wykorzystanie procesora, administratorzy mogą identyfikować zasoby niedostatecznie lub nadmiernie wykorzystane. Niedostatecznie wykorzystane zasoby procesora wskazują na potencjalne możliwości równoważenia obciążenia lub redystrybucji obciążenia, zapewniając lepsze wykorzystanie dostępnej mocy obliczeniowej. Z drugiej strony, identyfikacja nadmiernie wykorzystanych zasobów pomaga zapobiegać konfliktom o zasoby i zapewnia sprawiedliwy przydział zasobów procesora między różne procesy lub aplikacje.

Proaktywne wykrywanie problemów

Monitorowanie wykorzystania procesora w czasie rzeczywistym umożliwia proaktywną identyfikację problemów, zanim spowodują one poważne zakłócenia. Ustawiając progi i alerty, administratorzy mogą otrzymywać powiadomienia, gdy wykorzystanie procesora przekroczy określone limity. To proaktywne podejście pozwala im podjąć natychmiastowe działania, zbadać przyczynę problemu i zapobiec niestabilności lub przestojom systemu.

Planowanie i skalowanie wydajności

Dane dotyczące wykorzystania procesora gromadzone w czasie dostarczają cennych informacji do planowania wydajności i podejmowania decyzji dotyczących skalowania. Analizując historyczne trendy wykorzystania procesora, administratorzy mogą prognozować przyszłe zapotrzebowanie na zasoby, identyfikować okresy szczytowego obciążenia i odpowiednio alokować zasoby. Takie podejście zapewnia, że system może efektywnie obsługiwać zwiększone obciążenia, zapobiegając spadkowi wydajności lub awariom systemu w okresach szczytowego zapotrzebowania.

Efektywność zasobów i optymalizacja kosztów

Monitorowanie wykorzystania procesora pomaga w optymalizacji wykorzystania zasobów, co prowadzi do oszczędności kosztów. Identyfikując procesy lub aplikacje, które zużywają nadmiernie dużo zasobów procesora, administratorzy mogą dostroić konfiguracje systemu, zoptymalizować kod lub rozważyć alternatywne strategie alokacji zasobów. Taka optymalizacja nie tylko zwiększa wydajność systemu, ale także zmniejsza zużycie energii i koszty infrastruktury. Wykorzystując dane o wykorzystaniu procesora, organizacje mogą poprawić stabilność, responsywność i opłacalność systemu, co ostatecznie poprawia ogólne wrażenia użytkownika.

Metodologia

Importowanie niezbędnych bibliotek

Kod rozpoczyna się od zaimportowania wymaganych bibliotek, w tym „psutil" do monitorowania wykorzystania procesora, „datetime" do pracy ze znacznikami czasu, „time" do funkcji związanych z czasem, „schedule" do planowania zadań oraz „openpyxl" do interakcji z plikami Excela.

Wprowadzanie identyfikatora procesu przez użytkownika

Kod prosi użytkownika o podanie identyfikatora procesu („pid"). Ten identyfikator procesu służy do monitorowania wykorzystania procesora i pamięci przez określony proces.

Definiowanie funkcji „warning()"

Funkcja „warning()" służy do sprawdzania, czy użycie procesora lub pamięci przekracza określone progi. Wykorzystuje ona bibliotekę „psutil" do pobierania informacji o użyciu procesora i pamięci. Jeśli użycie procesora lub pamięci przekroczy 50%, na konsoli wyświetlany jest komunikat ostrzegawczy.

Definiowanie funkcji „monitor()"

Funkcja „monitor()" odpowiada za monitorowanie użycia procesora i pamięci oraz zapisywanie danych w pliku Excel. Wykonuje ona następujące kroki:

a. Pobieranie bieżącego czasu: Bieżący czas jest pobierany za pomocą funkcji „datetime.datetime.now()" i formatowany jako ciąg znaków.

b. Pobieranie użycia procesora i pamięci: Funkcja „psutil.Process()" służy do tworzenia obiektu procesu na podstawie określonego identyfikatora procesu. Użycie procesora jest obliczane poprzez podzielenie procentowego użycia procesora przez całkowitą liczbę procesorów. Wykorzystanie pamięci jest uzyskiwane za pomocą funkcji „memory_full_info()".

c. Otwieranie pliku Excel: Kod ładuje istniejący plik Excel lub tworzy nowy za pomocą funkcji „openpyxl.load_workbook()". Ścieżka do pliku jest określona jako „.\memory_data.xlsx".

d. Zapisywanie danych do pliku Excel: Dane o wykorzystaniu procesora i pamięci, wraz z bieżącym czasem i identyfikatorem procesu, są zapisywane do pliku Excel. Funkcja „sheet.cell()" służy do określenia współrzędnych komórki, w której zostaną zapisane dane.

e. Stosowanie opcji formatowania: Obramowanie i style czcionek są stosowane do komórek za pomocą klas „Border" i „Font" z modułu „openpyxl.styles". Poprawia to wygląd danych w pliku Excel.

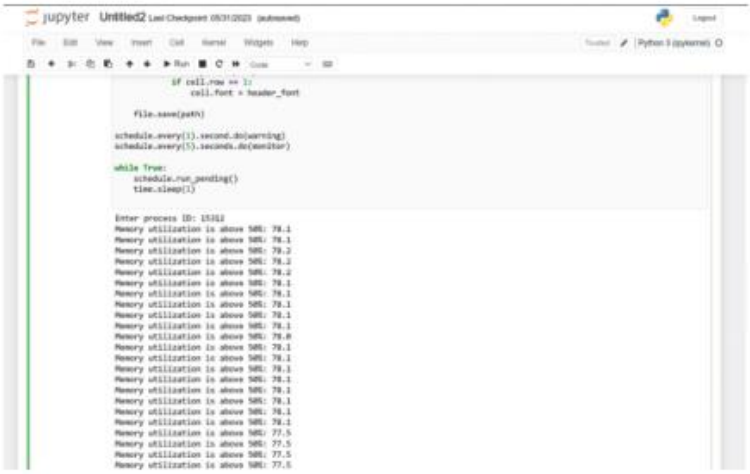
f. Zapisywanie pliku Excel: Zmodyfikowany plik Excel jest zapisywany za pomocą metody „save()" obiektu „Workbook".

Planowanie zadań monitorowania

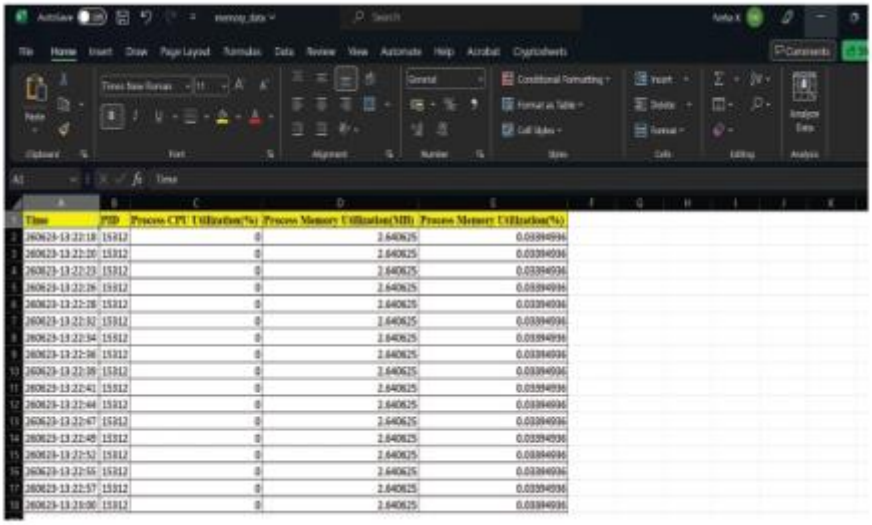
Kod wykorzystuje bibliotekę „schedule” do planowania uruchamiania funkcji „warning()” i „monitor()” w określonych odstępach czasu. Funkcja „schedule.every()” służy do określenia odstępu czasu, a funkcja „do()” służy do przypisywania funkcji do harmonogramu.

Uruchamianie pętli monitorowania

Kod uruchamia pętlę „while”, która działa w nieskończoność. Wewnątrz pętli wywoływana jest funkcja „schedule.run_pending()”, aby wykonać wszystkie oczekujące, zaplanowane zadania. Funkcja „time.sleep()” służy do wprowadzenia 1-sekundowego opóźnienia między iteracjami pętli. Łącząc te kroki, kod stale monitoruje użycie procesora i pamięci określonego procesu. Wyświetla komunikaty ostrzegawcze (rysunek 2)



, jeśli użycie przekroczy określone progi, i zapisuje dane dotyczące użycia w pliku Excel (rysunek 3) w celu dalszej analizy.



Zadania monitorowania są uruchamiane w określonych odstępach czasu, co umożliwia monitorowanie systemu w czasie rzeczywistym.

Kod Pythona

Aby utworzyć plik Excela

```
import openpyxl

path = r".\memory_data.xlsx"

# Create an empty workbook
workbook = openpyxl.Workbook()

# Access the active sheet
sheet = workbook.active

# Define the headers
headers = ["Time", "PID", "Process CPU Utilization (%)", "Process Memory
Utilization(MB)", "Process Memory Utilization(%)"]

# Assign headers to the first row
for col_num, header in enumerate (headers, 1):
    sheet.cell(row=1, column=col_num, value=header)

# Save the workbook to the specified path
workbook.save(path)
```

Aby monitorować procesor

```
import psutil
import datetime
import time
import schedule
import openpyxl
from openpyxl.styles import Font
from openpyxl.styles.borders import Border, Side

pid = int(input("Enter process ID: "))

def warning():
    cpu_usage = psutil.cpu_percent(interval=1)
    if cpu_usage > 50:
        print("CPU usage is above 50%:", cpu_usage)
    mem_usage = psutil.virtual_memory().percent
    if mem_usage > 50:
        print("Memory utilization is above 50%:", mem_usage)
def monitor():
    current_time = datetime.datetime.now().strftime("%d%m%y-%H:%M:%S")
    p = psutil.Process(pid)
    cpu = p.cpu_percent(interval=1) / psutil.cpu_count()
    memory_mb = p.memory_full_info().rss / (1024 * 1024)
    memory_percent = p.memory_percent()

    path = r".\memory_data.xlsx"
    file = openpyxl.load_workbook(path)
    sheet = file.active
    sheet.cell(column=1, row=sheet.max_row + 1, value=current_time)
    sheet.cell(column=2, row=sheet.max_row, value=pid)
    sheet.cell(column=3, row=sheet.max_row, value=cpu)
    sheet.cell(column=4, row=sheet.max_row, value=memory_mb)
    sheet.cell(column=5, row=sheet.max_row, value=memory_percent)

    # Apply borders to the table cells
    border_style = Border(left=Side(style='thin'),
                          right=Side(style='thin'),
                          top=Side(style='thin'),
                          bottom=Side(style='thin'))
# Set font style for header
    header_font = Font(name='Times New Roman', bold=True)
# Iterate through the cells in the sheet
    for row in sheet.iter_rows():
        for cell in row:
            # Apply border style
            cell.border = border_style

            # Set font style for header cells
            if cell.row == 1:
                cell.font = header_font
file.save(path)

schedule.every(1).second.do(warning)
schedule.every(5).seconds.do(monitor)
while True:
    schedule.run_pending()
    time.sleep(1)
```

Korzyści z monitorowania wykorzystania procesora w czasie rzeczywistym

Monitorowanie wykorzystania procesora w czasie rzeczywistym oferuje administratorom systemów i programistom szereg korzyści:

• Proaktywne wykrywanie problemów: Dzięki ciągłemu monitorowaniu wykorzystania procesora możliwe jest wykrywanie nieprawidłowych skoków obciążenia lub utrzymującego się wysokiego

poziomu wykorzystania w czasie rzeczywistym, co umożliwia wczesne wykrywanie problemów z wydajnością lub wąskich gardeł w zasobach.

• Optymalizacja zasobów: Monitorowanie w czasie rzeczywistym zapewnia wgląd w wzorce wykorzystania procesora, pomagając w identyfikacji procesów lub aplikacji zużywających nadmierne zasoby. Informacje te można wykorzystać do optymalizacji alokacji zasobów i poprawy ogólnej wydajności systemu.

• Analiza wydajności: Analizując historyczne dane dotyczące wykorzystania procesora, można zidentyfikować trendy i wzorce. Analiza ta pomaga zrozumieć zachowanie systemu, identyfikować okresy szczytowego obciążenia oraz podejmować świadome decyzje dotyczące planowania zasobów i zarządzania pojemnością.

• Rozwiązywanie problemów i diagnostyka: Monitorowanie wykorzystania procesora w czasie rzeczywistym może pomóc w rozwiązywaniu problemów związanych z wydajnością. Korelacja wysokiego wykorzystania procesora z określonymi zdarzeniami lub procesami ułatwia identyfikację pierwotnej przyczyny problemów z wydajnością i podejmowanie odpowiednich działań zaradczych.

Wdrażając solidny mechanizm monitorowania wykorzystania procesora, możemy zapewnić efektywne wykorzystanie zasobów, proaktywne wykrywanie problemów i optymalną wydajność systemu. Zebrane dane mogą być wykorzystane do dalszej analizy, podejmowania decyzji i poprawy ogólnej stabilności i wydajności systemu przesyłania wiadomości.

Śledzenie adresów URL

Twilio (rysunek 18.4) to platforma komunikacji w chmurze , która zapewnia programistom narzędzia i interfejsy API do tworzenia i wdrażania aplikacji komunikacyjnych.

Platforma umożliwia programistom integrację różnych kanałów komunikacji, w tym głosu, wiadomości, wideo i poczty e-mail, z ich aplikacjami za pomocą prostych i przyjaznych dla programistów interfejsów API.

Jedną z kluczowych ofert Twilio jest programowalny interfejs API SMS, który

umożliwia programistom wysyłanie i odbieranie wiadomości SMS z aplikacji do użytkowników na całym świecie. Usługa ta umożliwia wysyłanie automatycznych wiadomości, powiadomień lub alertów do odbiorcy. Numer telefonu Twilio to funkcja oferowana przez Twilio, która umożliwia firmom i programistom uzyskiwanie wirtualnych numerów telefonów do różnych celów. Dzięki numerowi telefonu Twilio użytkownicy mogą programowo wysyłać i odbierać wiadomości tekstowe oraz połączenia telefoniczne. Te wirtualne numery nie są powiązane z urządzeniem fizycznym ani lokalizacją, co zapewnia elastyczność i skalowalność potrzeb komunikacyjnych. Token uwierzytelniania i identyfikator bezpieczeństwa (SID) to ważne dane uwierzytelniające powiązane z kontem Twilio. Służą one jako mechanizm uwierzytelniania dostępu do interfejsów API i usług Twilio. Token uwierzytelniania to długi ciąg znaków, który działa jak hasło do konta Twilio. Służy do uwierzytelniania żądań API i zapewnia bezpieczny dostęp do zasobów konta. Z drugiej strony, Auth SID to unikalny identyfikator konta Twilio. Jest on używany w połączeniu z Auth Tokenem do uwierzytelniania i autoryzacji dostępu do funkcji konta. Zarówno Auth Token, jak i Auth SID są kluczowe dla zabezpieczenia dostępu do konta Twilio i powiązanych z nim usług. Ważne jest, aby zachować poufność tych danych uwierzytelniających

i unikać udostępniania ich osobom nieupoważnionym. Dane te są zazwyczaj wymagane podczas wysyłania żądań API do Twilio, niezależnie od tego, czy chodzi o wysyłanie wiadomości SMS, wykonywanie połączeń telefonicznych, czy zarządzanie różnymi ustawieniami konta.

Metodologia

Poniższy kod wykorzystuje bibliotekę „smtplib" do wysyłania powiadomień e-mail do odbiorcy. Rozpoczyna się od zdefiniowania konfiguracji poczty e-mail, w tym adresu e-mail nadawcy, danych serwera SMTP (w tym przypadku serwera SMTP Gmaila) oraz nazwy użytkownika i hasła nadawcy. Następnie tworzony jest obiekt „MIMEMultipart" w celu skonstruowania wiadomości e-mail. Ustawiane są nadawca, odbiorca i temat, a adres URL jest dodawany do treści wiadomości e-mail. Na koniec wiadomość łączy się z serwerem SMTP za pomocą szyfrowania TLS, loguje się, podając nazwę użytkownika i hasło, i wysyła wiadomość za pomocą metody „sendmail". Jeśli wiadomość zostanie wysłana pomyślnie, wyświetlany jest komunikat o pomyślnym wysłaniu (rysunek 5).

Kod Pythona

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from twilio.rest import Client

def send_email_with_notification(url, recipient_email):
    # Email Configuration
    sender_email = 'emailalertbot13@gmail.com'
    smtp_server = 'smtp.gmail.com'
    smtp_port = 587
    smtp_username = 'emailalertbot13@gmail.com'
    smtp_password = 'my password'

    # Create a multipart message
    message = MIMEMultipart()
    message['From'] = sender_email
    message['To'] = recipient_email
    message['Subject'] = 'URL Notification'

    # Add the URL to the email body
    body = f"Hi,\n\nHere is the URL: {url}"
    message.attach(MIMEText(body, 'plain'))

    # Connect to the SMTP server and send the email
    with smtplib.SMTP(smtp_server, smtp_port) as server:
        server.starttls()
        server.login(smtp_username, smtp_password)
        server.sendmail(sender_email, recipient_email, message.as_string())

    print("Email sent successfully!")
```

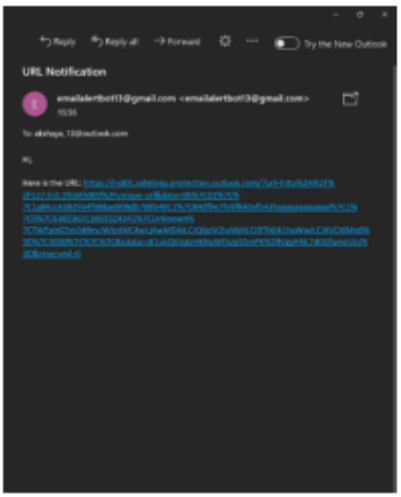
Dane wyjściowe

Poniższy kod serwera definiuje trasę dla unikalnego adresu URL („/unique-url”). Żądanie skierowane do tego adresu URL uruchamia funkcję „unique_url”. Wewnątrz tej funkcji wywoływana jest funkcja „send_sms_notification”, która wysyła powiadomienie SMS. Definiowane są identyfikator SID konta Twilio, token uwierzytelniający i numer telefonu Twilio. Następnie tworzona jest instancja klasy „Client” Twilio przy użyciu identyfikatora SID konta i tokena uwierzytelniającego. Metoda „create” służy do wysłania wiadomości SMS na numer telefonu wskazanego odbiorcy. Powiadomienie jest wyświetlane na konsoli, a zwracana jest prosta odpowiedź „Cześć, to Akshaya”.

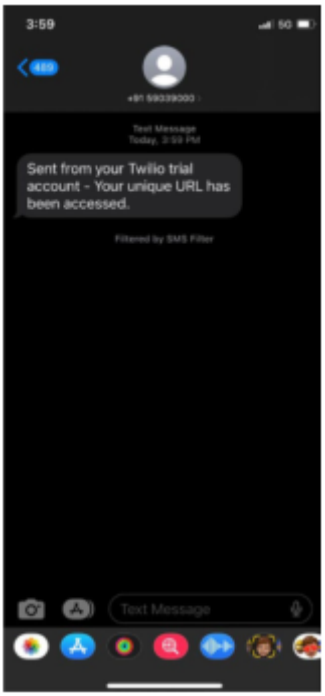
Kod Pythona

```
from flask import Flask, request
from twilio.rest import Client
app = Flask(__name__)
def send_sms_notification(to_number, body):
    account_sid = 'ACbd3ecf4a42c68168921b1f1346e28f04'
    auth_token = '2ddec7ea29d617d76197988ab73417a5'
    twilio_number = '+14175383274'

    client = Client(account_sid, auth_token)
    message = client.messages.create(
        body=body,
        from_=twilio_number,
        to=to_number
    )
 print("SMS notification sent successfully!")
@app.route('/unique-url')
def unique_url():
    # Trigger SMS notification when the URL is accessed
    to_number = 'receipent phone number'  # Replace with the recipient's
phone number
    notification_text = 'Your unique URL has been accessed.'
    send_sms_notification(to_number, notification_text)
 print(notification_text)  # Print the notification message to the console

    return 'Hi, this is Akshaya'

if __name__ == '__main__':
    app.run()
```

Podsumowując, po wykonaniu kodu e-mail na adres e-mail odbiorcy wysyłana jest wiadomość e-mail z podanym adresem URL [5] (rysunek 6).



Gdy odbiorca uzyska dostęp do unikalnego adresu URL, uruchamiany jest kod serwera, a na wskazany numer telefonu wysyłane jest powiadomienie SMS (rysunek 7) za pomocą interfejsu API Twilio.



Monitorowanie wydajności automatycznego dostarczania

Baza danych zawiera następujące dane:

- message_id: Unikalny identyfikator każdej wiadomości
- extra_attributes: Dodatkowe atrybuty powiązane z wiadomością, przechowywane jako JSON
- batch_id: Identyfikator partii wiadomości
- campaign_id: Identyfikator kampanii powiązanej z wiadomością
- is_unicode: Flaga wskazująca, czy wiadomość jest w formacie Unicode
- recipient: Numer telefonu odbiorcy
- route_id: Identyfikator trasy użytej do wysłania wiadomości
- source_id: Identyfikator źródła wiadomości
- user_code: Kod użytkownika powiązany z wiadomością
- sender_name: Nazwa nadawcy
- channel: Kanał użyty do wysłania wiadomości (w tym przypadku „http”)
- email: Adres e-mail powiązany z wiadomością
- operator: Informacje o operatorze
- country: Kraj odbiorcy
- processed_date: Data i godzina przetworzenia wiadomości
- description: Opis statusu wiadomości lub błędu
- code: Kod błędu powiązany z wiadomością

- submission_date: Data i godzina wysłania wiadomości wysłane

- message_pdu: PDU (jednostka danych protokołu) wiadomości

- message_state: stan wiadomości

- message_text: treść tekstowa wiadomości

- dlr_success_pdu: PDU pomyślnego potwierdzenia doręczenia

- dlr_failed_pdu: PDU nieudanego potwierdzenia doręczenia

- dlr_state: stan potwierdzenia doręczenia

- done_date: data i godzina zakończenia procesu doręczenia

- extra1 do extra7: dodatkowe pola ze szczegółowymi informacjami

- dlr_details: szczegóły potwierdzenia doręczenia

- level1_dlt_price: cena związana z technologią DLT (Distributed Ledger Technology)

- level1_price: cena wiadomości na poziomie 1

- level1_cost: koszt wiadomości na poziomie 1

- user_price: cena wiadomości dla użytkownika

- user_cost: koszt wiadomości dla użytkownika

- level1_refund: kwota zwrotu na poziomie 1

- user_refund: kwota zwrotu dla użytkownik

- final_error_code: Ostateczny kod błędu powiązany z komunikatem

- extra4 do extra6: Dodatkowe pola ze szczegółowymi informacjami

- extra7: Dodatkowe pole ze szczegółowymi informacjami

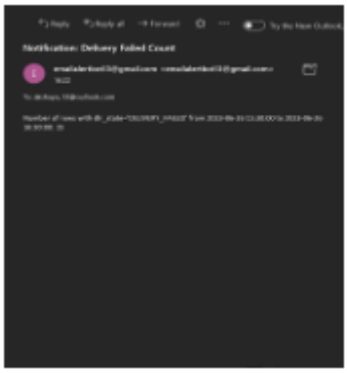
Zatem z podanej bazy danych nawiązaliśmy połączenie z bazą danych MySQL i pobraliśmy dane w oparciu o określony przedział czasowy. Obliczona została liczba wiadomości ze statusem „DELIVERY_FAILED”, wskazującym na problemy z dostarczeniem. Liczba ta służy do oceny wydajności dostarczania i identyfikacji potencjalnych problemów. Jeśli liczba wiadomości przekroczy określony próg, wysyłane jest powiadomienie SMS za pośrednictwem Twilio oraz powiadomienie e-mail za pośrednictwem SMTP. Kod zapewnia automatyczne monitorowanie i proaktywne alerty dotyczące problemów z dostarczeniem.

Metodologia

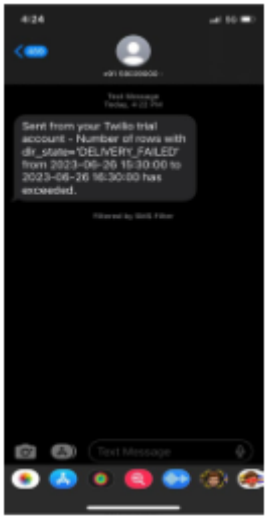
Importowane są niezbędne biblioteki, w tym „mysql.connector” do łączności z bazą danych MySQL, „pandas” do przetwarzania danych, „smtplib” do obsługi poczty e-mail oraz „twilio.rest.Client” do interakcji z API Twilio. Określone są dane uwierzytelniające konta Twilio i numery telefonów. Te dane uwierzytelniające służą do uwierzytelniania i autoryzacji dostępu do konta Twilio. Podano szczegóły konfiguracji poczty e-mail, w tym adresy e-mail nadawcy i odbiorcy, adres serwera SMTP, port, nazwę użytkownika i hasło. Dane te służą do nawiązania połączenia z serwerem SMTP w celu wysyłania powiadomień e-mail. Kod nawiązuje połączenie z bazą danych MySQL za pomocą funkcji „mysql.connector.connect()”. Wymaga podania hosta, nazwy użytkownika, hasła i nazwy bazy danych.

Należy zastąpić wartości zastępcze rzeczywistymi danymi uwierzytelniającymi bazy danych MySQL. Po nawiązaniu połączenia z bazą danych, za pomocą funkcji „connection.cursor()” tworzony jest obiekt kursora. Kursor umożliwia wykonywanie zapytań SQL i pobieranie danych z bazy danych. Kod wykonuje zapytanie SQL w celu pobrania wierszy z tabeli „progate_mdr_tbl” na podstawie określonego zakresu czasu zdefiniowanego w kolumnie „submission_date”. Zapytanie jest wykonywane za pomocą funkcji „cursor.execute(query)”. Pobrane wiersze są pobierane za pomocą metody „cursor.fetchall()”. Wiersze te zawierają dane pobrane z bazy danych. Nazwy kolumn są wyodrębniane z opisu kursora za pomocą wyrażenia listowego i zapisywane na liście „columns”. Pobrane dane są następnie wykorzystywane do utworzenia ramki danych pandas („df”) z wyodrębnionymi nazwami kolumn jako nagłówkami kolumn. Ramka danych jest eksportowana do pliku Excel określonego przez parametr „file_path” za pomocą metody „df.to_excel()”. Powoduje to utworzenie pliku Excel zawierającego dane z tabeli bazy danych. Kod zlicza liczbę wierszy, w których wartość w kolumnie „dlr_state” jest równa „DELIVERY_FAILED”. Robi to poprzez iterację po pobranych wierszach i sprawdzanie wartości kolumny „dlr_state”. Liczba jest zapisywana w zmiennej „failed_count”. Kursor i połączenie z bazą danych są zamykane za pomocą „cursor.close()” i „connection.close()”, aby zwolnić zasoby i zapewnić prawidłowe czyszczenie. Jeśli „failed_count” jest większe niż 10, kod kontynuuje wysyłanie powiadomień. Na koniec kod wyświetla komunikat na konsoli (rysunek 8)

informujący o wyeksportowaniu danych do programu Excel oraz wysłaniu powiadomień SMS i e-mail. Tworzone jest powiadomienie e-mail (rysunek 9)



z określonym tematem, treścią, nadawcą i odbiorcą. Wiadomość e-mail jest wysyłana za pomocą skonfigurowanego serwera SMTP. Menedżer kontekstu „smtplib.SMTP()” służy do nawiązania bezpiecznego połączenia, zalogowania się przy użyciu podanych danych uwierzytelniających i wysłania wiadomości e-mail. Za pomocą klienta Twilio wysyłane jest powiadomienie SMS (rysunek 10)



na wskazany numer odbiorcy. Treść wiadomości zawiera informacje o liczbie wierszy ze statusem „DELIVERY_FAILED” podczas zdefiniowanej ramki powiązania.

Kod

```python
import mysql.connector
import pandas as pd
import smtplib
from email.mime.text import MIMEText
from twilio.rest import Client

# Twilio account credentials
twilio_account_sid = 'accnt id'
twilio_auth_token = 'accnt pswd'
twilio_phone_number = '+14175383274'
your_phone_number = 'my phone no'

# Email configuration
email_from = 'emailalertbot13@gmail.com'
email_to = 'akshaya_13@outlook.com'
smtp_server = 'smtp.gmail.com'
smtp_port = 587
smtp_username = 'emailalertbot13@gmail.com'
smtp_password = 'mypassword'

# Establish connection
connection = mysql.connector.connect(
    host="localhost",  # Replace with your MySQL host
    user="root",  # Replace with your MySQL username
    password="mypassword",  # Replace with your MySQL password
    database="progate"  # Replace with your MySQL database name
)

# Create a cursor object
cursor = connection.cursor()

# Execute a query to get the rows that match the condition
query = """
    SELECT *
    FROM progate_mdr_tbl
    WHERE submission_date >= '2023-06-26 15:30:00'
    AND submission_date <= '2023-06-26 16:30:00'
"""
cursor.execute(query)

# Fetch all rows
rows = cursor.fetchall()

# Get the column names from the cursor
columns = [column[0] for column in cursor.description]

# Create a pandas DataFrame
df = pd.DataFrame(rows, columns=columns)

# Specify the file path and name for the Excel file
file_path = 'C:\\Users\\Akshaya\\OneDrive\\Pantech\\processed_data.xlsx'

# Export the DataFrame to Excel
df.to_excel(file_path, index=False)

# Count the number of rows where dlr_state is 'DELIVERY_FAILED'
dlr_state_column_index = columns.index('dlr_state')
failed_count = sum(1 for row in rows if row[dlr_state_column_index] ==
'DELIVERY_FAILED')
print("Number of rows with dlr_state='DELIVERY_FAILED':", failed_count)

# Close the cursor and connection
cursor.close()
connection.close()

# Send an SMS and email notification only if failed_count is more than 10
if failed_count > 10:
    # Send an SMS using Twilio
    client = Client(twilio_account_sid, twilio_auth_token)
    message = client.messages.create(
        body=f"Number of rows with dlr_state='DELIVERY_FAILED' from 2023-
06-26 15:30:00 to 2023-06-26 16:30:00 has exceeded.",
        from_=twilio_phone_number,
        to=your_phone_number
    )

    # Send an email notification
    email_subject = 'Notification: Delivery Failed Count'
    email_body = f"Number of rows with dlr_state='DELIVERY_FAILED' from
2023-06-26 15:30:00 to 2023-06-26 16:30:00: {failed_count}"

    msg = MIMEText(email_body)
    msg['From'] = email_from
    msg['To'] = email_to
    msg['Subject'] = email_subject

    with smtplib.SMTP(smtp_server, smtp_port) as server:
        server.starttls()
        server.login(smtp_username, smtp_password)
        server.send_message(msg)

print("Data exported to Excel.")
print("SMS and email has sent.")
```

Wyjście

Zastosowania

Kampanie marketingowe

Firmy prowadzące kampanie marketingowe SMS lub e-mail mogą używać tego kodu do monitorowania skuteczności dostarczania wiadomości. Pomaga on identyfikować wszelkie problemy lub niepowodzenia w dostarczaniu wiadomości, umożliwiając szybkie podejmowanie działań w celu zapewnienia skuteczności kampanii.

Powiadomienia transakcyjne

Organizacje wysyłające ważne wiadomości transakcyjne [5], takie jak potwierdzenia zamówień lub powiadomienia o kontach, mogą wykorzystywać ten kod do śledzenia statusu dostarczania. Umożliwia im to identyfikację i rozwiązanie wszelkich problemów z dostarczaniem klientom informacji wrażliwych na czas.

Systemy obsługi klienta

Firmy posiadające systemy obsługi klienta mogą używać tego kodu do monitorowania statusu dostarczania wiadomości związanych z pomocą techniczną. Pomaga on zapewnić, że krytyczne komunikaty dla klientów, takie jak aktualizacje zgłoszeń lub powiadomienia serwisowe, są dostarczane pomyślnie, umożliwiając sprawną obsługę klienta.

Alerty systemowe

Firmy mogą zintegrować ten kod ze swoimi narzędziami do monitorowania systemu, aby śledzić dostarczanie wiadomości alertowych. W przypadku jakichkolwiek problemów z dostarczaniem wiadomości, administratorzy systemu lub zespoły wsparcia mogą zostać natychmiast powiadomieni, co pozwoli im na podjęcie szybkich działań w celu rozwiązania problemów systemowych.

Ocena wydajności

Ten kod może być używany przez dostawców usług do oceny wydajności infrastruktury dostarczania wiadomości. Analizując liczbę nieudanych dostaw w określonych ramach czasowych, mogą oni ocenić niezawodność i wydajność swoich systemów przesyłania wiadomości oraz wprowadzić niezbędne usprawnienia. Ogólnie rzecz biorąc, kod ten stanowi cenne narzędzie do monitorowania dostarczania wiadomości i zapewnienia efektywnej komunikacji w różnych rzeczywistych sytuacjach, pomagając firmom utrzymać zadowolenie klientów i wydajność operacyjną.

Alert o dużej liczbie wiadomości testowych

Otrzymaliśmy od naszego branżowego przewodnika plik bazy danych zawierający różne rekordy wiadomości. Celem była analiza tej bazy danych i identyfikacja w danym momencie przypadków, w których znaczna liczba wiadomości testowych została wysłana do konkretnego odbiorcy. W przypadku testowym próg wynosił 5. Przeszukaliśmy bazę danych w oparciu o określone kryteria, koncentrując się na wiadomościach wysłanych do odbiorcy z numerem telefonu +919812455454, pochodzących ze

źródła o identyfikatorze „http” i mieszczących się w przedziale czasowym od 26.06.2023 godz. 15:00:00 do 26.06.2023 godz. 15:40:00. Wyodrębnienie odpowiednich wierszy z bazy danych pozwoliło nam wyizolować interesujące nas wiadomości do dalszej analizy. Aby określić dokładną liczbę, zaimplementowaliśmy funkcję zliczania w wyodrębnionych danych, która ujawniła, że liczba wiadomości testowych przekroczyła próg 5. Aby poradzić sobie z tak dużą liczbą wiadomości testowych, skonfigurowaliśmy system powiadomień za pomocą usługi przesyłania wiadomości Twilio . Powiadomienia wysyłano zarówno e-mailem, jak i SMS-em, aby wskazać odbiorców, jeśli liczba wiadomości przekroczyła próg.

Metodologia

Import niezbędnych bibliotek

Kod importuje wymagane biblioteki, w tym „mysql.connector”, „pandas”, „smtplib”, „MIMEText” z „email.mime.text” oraz „Client”

z „twilio.rest”.

Konfiguracja poświadczeń Twilio i poczty e-mail

Kod konfiguruje poświadczenia konta Twilio (SID konta, token uwierzytelniający i numery telefonów) oraz konfigurację poczty e-mail (adres nadawcy, adres odbiorcy i dane serwera SMTP).

Nawiązywanie połączenia z bazą danych MySQL

Kod nawiązuje połączenie z bazą danych MySQL , podając hosta, nazwę użytkownika, hasło i nazwę bazy danych.

Tworzenie obiektu kursora i wykonywanie zapytania

Kod tworzy obiekt kursora do interakcji z bazą danych i wykonuje zapytanie SQL w celu wybrania wierszy z tabeli „progate_mdr_tbl” na podstawie określonych warunków, takich jak identyfikator źródła, odbiorca, tekst wiadomości i data przetworzenia.

Pobieranie danych i tworzenie ramki danych Pandas

Kod pobiera wszystkie wiersze zwrócone przez zapytanie i nazwy kolumn. Następnie tworzy ramkę danych Pandas, używając pobranych danych i nazw kolumn.

Eksport danych do programu Excel

Kod określa ścieżkę i nazwę pliku Excel oraz eksportuje ramkę danych do pliku Excel za pomocą funkcji „to_excel”.

Zliczanie wiadomości testowych

Kod zlicza liczbę wierszy, w których tekst wiadomości to „wiadomość testowa”, iterując po wierszach i sprawdzając wartość kolumny „message_text”.

Zamykanie kursora i połączenia

Kod zamyka kursor i połączenie z bazą danych MySQL .

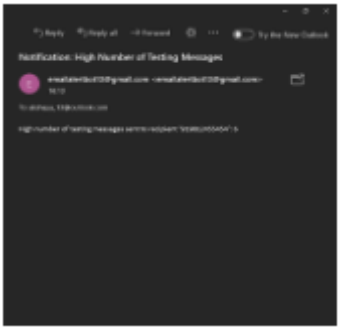
Wyświetlanie komunikatów o stanie

Kod wyświetla na konsoli komunikat (rysunek 11)

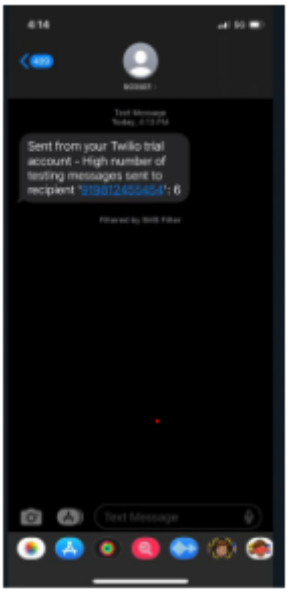
informujący o pomyślnym eksporcie danych do programu Excel oraz o tym, czy na podstawie liczby wiadomości testowych wysłano powiadomienia SMS i e-mail.

Wysyłanie powiadomień SMS i e-mail

Jeśli liczba wiadomości testowych przekracza 5, kod wysyła powiadomienie e-mail (rysunek 12)



za pomocą dostarczonej konfiguracji poczty e-mail, a także wysyła powiadomienie SMS (rysunek 13)



za pomocą Twilio. Wiadomość SMS zawiera liczbę wiadomości testowych, a temat i treść wiadomości są również generowane na podstawie tej liczby.

Kod Pythona

```python
import mysql.connector
import pandas as pd
import smtplib
from email.mime.text import MIMEText
from twilio.rest import Client

# Twilio account credentials
twilio_account_sid = 'my acount id'
twilio_auth_token = 'my token'
twilio_phone_number = '+14175383274'
your_phone_number = 'my number'

# Email configuration
email_from = 'emailalertbot13@gmail.com'
email_to = 'akshaya_13@outlook.com'
smtp_server = 'smtp.gmail.com'
smtp_port = 587
smtp_username = 'emailalertbot13@gmail.com'
smtp_password = 'my password'

# Establish connection
connection = mysql.connector.connect(
    host="localhost",  # Replace with your MySQL host
    user="root",  # Replace with your MySQL username
    password="my password",  # Replace with your MySQL password
    database="progate"  # Replace with your MySQL database name
)

# Create a cursor object
cursor = connection.cursor()

# Execute a query to get the rows that match the condition
query = """
    SELECT *
    FROM progate_mdr_tbl
    WHERE source_id = 'http_f1'
    AND recpient = '919812455454'
    AND message_text = 'testing message'
    AND processed_date >= '2023-06-26 15:00:00'
    AND processed_date <= '2023-06-26 15:40:00'
"""
cursor.execute(query)

# Fetch all rows
rows = cursor.fetchall()

# Get the column names from the cursor
columns = [column[0] for column in cursor.description]

# Create a pandas DataFrame
df = pd.DataFrame(rows, columns=columns)

# Specify the file path and name for the Excel file
file_path = 'C:\\Users\\Akshaya\\OneDrive\\Pantech\\processed_data.xlsx'

# Export the DataFrame to Excel
df.to_excel(file_path, index=False)
```

```
# Count the number of rows where message_txt is 'testing message'
message_txt_column_index = columns.index('message_text')
testing_count = sum(1 for row in rows if row[message_txt_column_index] ==
'testing message')
print("Number of testing messages sent to recipient '919812455454':",
testing_count)

# Close the cursor and connection
cursor.close()
connection.close()

# Send an SMS and email notification only if testing_count is more than 5
if testing_count > 5:
    # Send an SMS using Twilio
    client = Client(twilio_account_sid, twilio_auth_token)
    message = client.messages.create(
        body=f"High number of testing messages sent to recipient
'919812455454': {testing_count}",
        from_=twilio_phone_number,
        to=your_phone_number
    )

    # Send an email notification
    email_subject = 'Notification: High Number of Testing Messages'
    email_body = f"High number of testing messages sent to recipient
'919812455454': {testing_count}"

    msg = MIMEText(email_body)
    msg['From'] = email_from
    msg['To'] = email_to
    msg['Subject'] = email_subject

    with smtplib.SMTP(smtp_server, smtp_port) as server:
        server.starttls()
        server.login(smtp_username, smtp_password)
        server.send_message(msg)

    print("Data exported to Excel.")
    print("SMS and Email notifications sent successfully if the
testing_count > 5.")
else:
    print("Data exported to Excel.")
    print("No SMS or Email notifications sent as the testing_count <= 5."
```

Wyjście

```
Microsoft Windows [Version 10.0.22621.1848]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Akshaya\OneDrive\Desktop\pythonrc:/Users/Akshaya/OneDrive/Desktop/python/venv/Scripts/activate.bat

(venv) C:\Users\Akshaya\OneDrive\Desktop\pythonrc:/Users/Akshaya/OneDrive/Desktop/python/venv/Scripts/python.exe c:/Users/Aksh
ya/OneDrive/Desktop/python/testmg.py
Number of testing messages sent to recipient '919812455454': 6
Data exported to Excel.
SMS and Email notifications sent successfully if the testing_count > 5.

(venv) C:\Users\Akshaya\OneDrive\Desktop\pythonrc
```

Wnioski

Podsumowując, opracowanie narzędzia do generowania alertów dla systemów przesyłania wiadomości stanowi cenne rozwiązanie do proaktywnego monitorowania i zarządzania dostarczaniem wiadomości. Korzystając z tego narzędzia, organizacje mogą skutecznie śledzić status swoich wiadomości i szybko reagować na wszelkie problemy z ich dostarczaniem. Zdolność narzędzia do generowania alertów, takich jak powiadomienia e-mail lub wiadomości SMS, umożliwia aktualizacje w czasie rzeczywistym i natychmiastowe działania w celu rozwiązania wszelkich problemów. Dzięki temu narzędziu organizacje mogą zwiększyć niezawodność swojego systemu przesyłania wiadomości, zapewnić terminową komunikację z odbiorcami i utrzymać wysokie standardy obsługi. Ostatecznie narzędzie do generowania alertów przyczynia się do poprawy efektywności operacyjnej, zadowolenia klientów i ogólnej wydajności systemu przesyłania wiadomości.