Project document:

13 November 2024 10:50

Project title: Automate Invoice and Payment Notifications to Slack via Xero and Zapier

Objective:

Automate notifications to a Slack channel whenever a new invoice is created or a payment is received in Xero. This will streamline communication and ensure timely updates on financial transactions directly in Slack.

Tools Required:

- **Xero Account**
- **Slack Account**
- **Zapier Account**
- **Python**
- **Requests library** (pip install requests)

Steps:

1. Set Up Your Accounts

Ensure you have active accounts with Xero, Slack, and Zapier.

2. Obtain API Credentials

- **Xero:** Obtain your API credentials from the Xero developer portal.
- **Slack:** Create a Slack app and obtain the Webhook URL for sending messages to a Slack channel.
- **Zapier:** Create a Zapier account and set up a new Zap to connect Xero and Slack.

3. Install Required Libraries

Install the requests library in Python:

Use below mentioned command for installing requests library:

Command:pip install requests

4. Set Up Zapier

1. **Create a Zap:**
    - Log in to your Zapier account and create a new Zap.
2. **Set Up the Trigger:**
    - **App:** Xero
    - **Event:** New Sales Invoice or New Payment

- **Account:** Connect your Xero account
- **Trigger:** Set up the trigger to capture new invoices or payments.

3. **Set Up the Action:**

   - **App:** Webhooks by Zapier
   - **Event:** Custom Request
   - **Method:** POST
   - **URL:** Use the Slack Webhook URL (https://hooks.slack.com/services/YOUR_SLACK_WEBHOOK_URL)
   - **Data:** Construct the payload with the invoice or payment details.

5. Python Script

Example .env File

Make sure this file is named .env and located in the same directory as your Python script:

YOUR_XERO_CLIENT_ID=CF15468A353C46588071B43CC1636C67
YOUR_XERO_CLIENT_SECRET=UK2LT80ZGozBy7ZHEf-oRP5hD6vgdPo6kQa-5QvOoIpjWEPf
YOUR_XERO_TENANT_ID=2820b5d5-fc3b-4f8c-9ecc-213482005224
YOUR_XERO_REFRESH_TOKEN=u9fFPDzhxr_vL4rCdO5BRcosP6I97B7zYwxVEnzbs9M
YOUR_SLACK_WEBHOOK_URL=https://hooks.slack.com/services/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

Complete Python Script

```
import requests
import json
import time
import os
from requests.auth import HTTPBasicAuth
from dotenv import load_dotenv

# Load environment variables from a .env file
load_dotenv()

# Xero and Slack credentials from environment variables
XERO_CLIENT_ID = os.getenv("CF15468A353C46588071B43CC1636C67")
XERO_CLIENT_SECRET = os.getenv("UK2LT80ZGozBy7ZHEf-oRP5hD6vgdPo6kQa-5QvOoIpjWEPf")
XERO_TENANT_ID = os.getenv("2820b5d5-fc3b-4f8c-9ecc-213482005224")
XERO_REFRESH_TOKEN = os.getenv("u9fFPDzhxr_vL4rCdO5BRcosP6I97B7zYwxVEnzbs9M")
SLACK_WEBHOOK_URL =
os.getenv("https://hooks.slack.com/services/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")

print(f"Client ID: {XERO_CLIENT_ID}")
print(f"Client Secret: {XERO_CLIENT_SECRET}")
print(f"Tenant ID: {XERO_TENANT_ID}")
print(f"Refresh Token: {XERO_REFRESH_TOKEN}")
print(f"Slack Webhook URL: {SLACK_WEBHOOK_URL}")

# Temporary storage for pending invoices (in memory for simplicity)
pending_invoices = {}
```

```
def get_xero_access_token():
    # Function to refresh and get a new access token from Xero
    token_url = "https://identity.xero.com/connect/token"
    token_data = {
        "grant_type": "refresh_token",
        "refresh_token": XERO_REFRESH_TOKEN,
    }
    response = requests.post(token_url, data=token_data, auth=HTTPBasicAuth(XERO_CLIENT_ID,
XERO_CLIENT_SECRET))
    if response.status_code == 200:
        tokens = response.json()
        return tokens['access_token']
    else:
        raise Exception(f"Failed to get access token: {response.text}")

def fetch_invoices(access_token):
    # Function to fetch invoices from Xero
    invoice_url = f"https://api.xero.com/api.xro/2.0/Invoices"
    headers = {
        "Authorization": f"Bearer {access_token}",
        "Xero-tenant-id": XERO_TENANT_ID,
    }
    response = requests.get(invoice_url, headers=headers)
    if response.status_code == 200:
        return response.json()["Invoices"]
    else:
        raise Exception(f"Failed to fetch invoices: {response.text}")

def fetch_payments(access_token):
    # Function to fetch payments from Xero
    payment_url = f"https://api.xero.com/api.xro/2.0/Payments"
    headers = {
        "Authorization": f"Bearer {access_token}",
        "Xero-tenant-id": XERO_TENANT_ID,
    }
    response = requests.get(payment_url, headers=headers)
    if response.status_code == 200:
        return response.json()["Payments"]
    else:
        raise Exception(f"Failed to fetch payments: {response.text}")

def send_slack_notification(message):
    payload = {'text': message}
    response = requests.post(SLACK_WEBHOOK_URL, data=json.dumps(payload), headers={'Content-
Type': 'application/json'})
    if response.status_code != 200:
        raise Exception(f"Request to Slack returned an error {response.status_code}, the response is:
\n{response.text}")

def main():
    while True:
        try:
            # Refresh access token
            access_token = get_xero_access_token()

            # Fetch and send notifications for new invoices
            invoices = fetch_invoices(access_token)
```

```
        for invoice in invoices:
            if invoice['Status'] == 'AUTHORISED':  # Check if the invoice is authorised
                pending_invoices[invoice['InvoiceNumber']] = invoice
                send_slack_notification(f"New Invoice Created: {invoice['InvoiceNumber']}")

        # Fetch payments and update pending invoices status
        payments = fetch_payments(access_token)
        for payment in payments:
            invoice_number = payment['Invoice']['InvoiceNumber']
            if invoice_number in pending_invoices:
                send_slack_notification(f"Payment Received: {payment['PaymentID']} for Invoice:
{invoice_number}")
                del pending_invoices[invoice_number]

        # Sleep for a specified time before checking again
        time.sleep(300)  # Check every 5 minutes
    except Exception as e:
        print(f"An error occurred: {e}")
        time.sleep(60)  # Wait for a minute before retrying

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

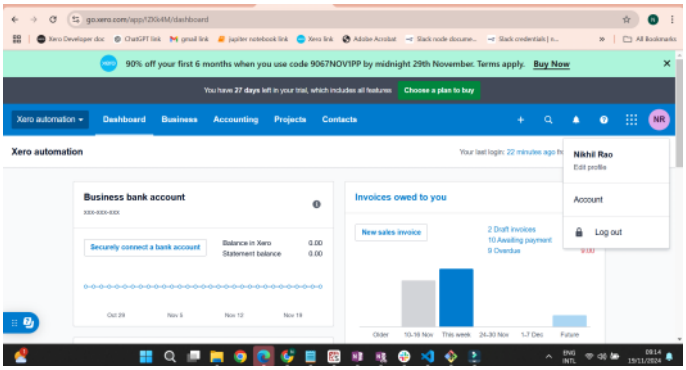
Final Steps:

1. **Replace Placeholder Values:**
    - Replace YOUR_XERO_CLIENT_ID, YOUR_XERO_CLIENT_SECRET, YOUR_XERO_TENANT_ID, YOUR_XERO_REFRESH_TOKEN, and YOUR_SLACK_WEBHOOK_URL with your actual credentials.
2. **Run the Script:**
    - Ensure the .env file is in the same directory as your script.
    - Run the script to start monitoring for new invoices and payments and send notifications to Slack.

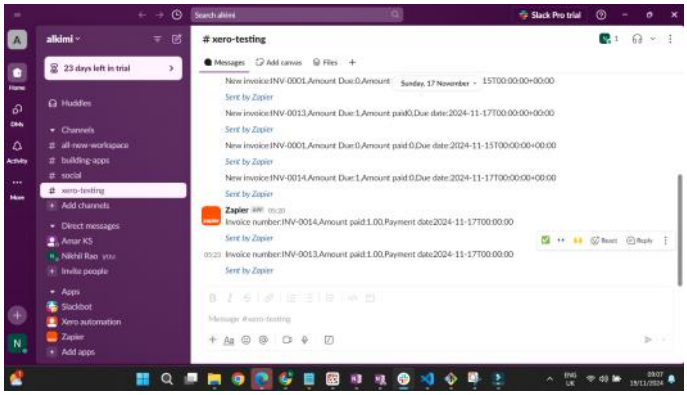
Xero Project screenshots:

15 November 2024 11:25

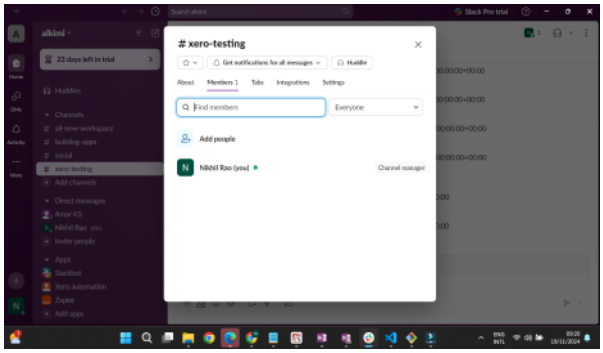
Screenshot of Xero account:



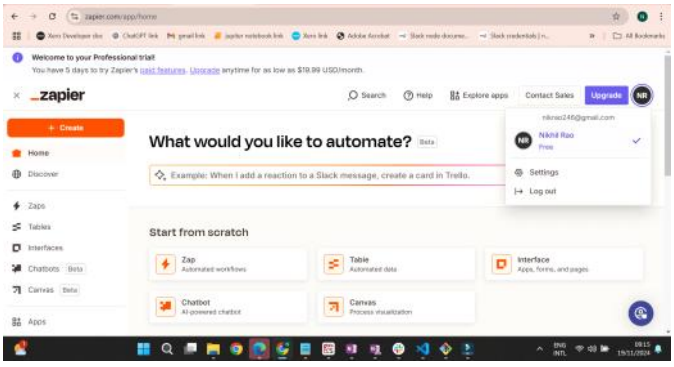
Screenshot of invoice
And payment
Notifications:



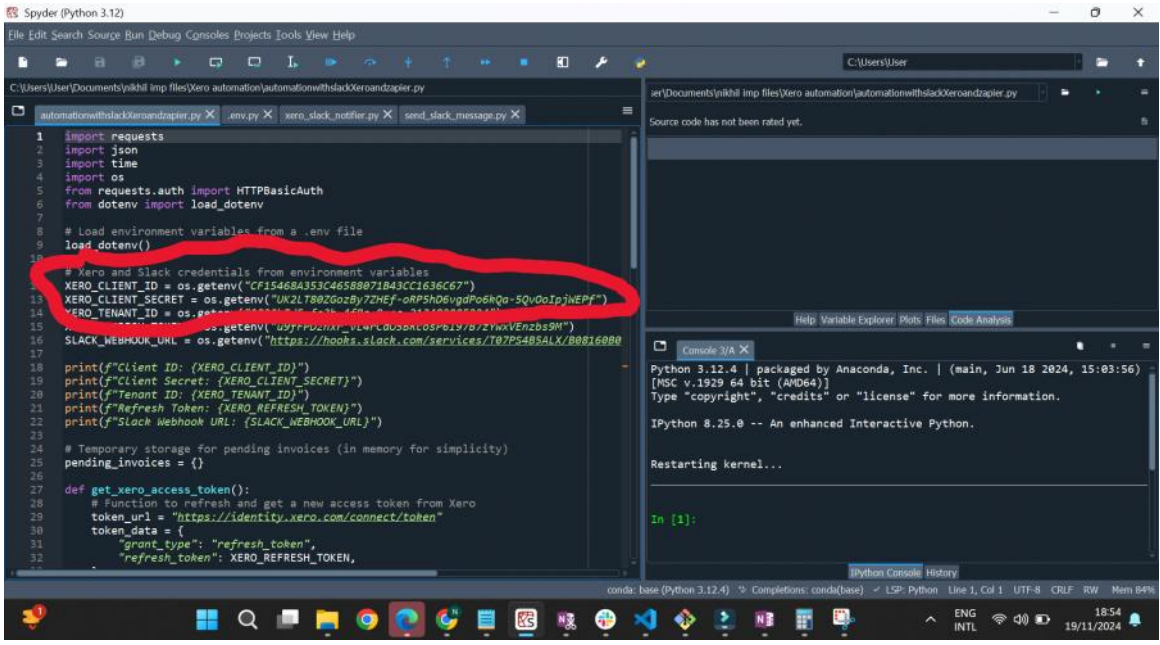
Screenshot of Slack
account:



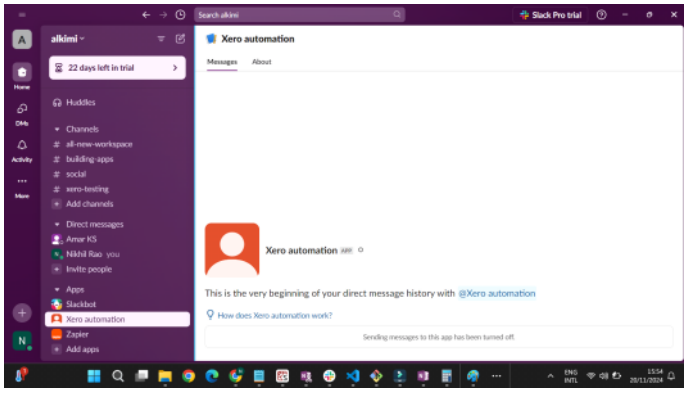
Screenshot of Zapier account:



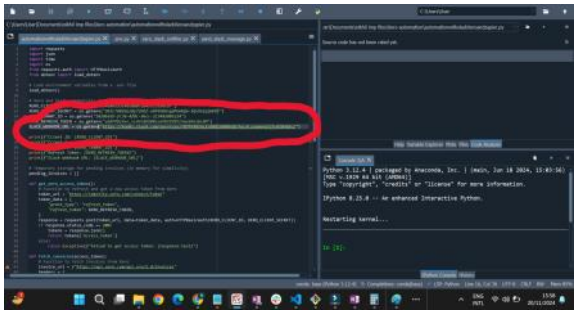
Screenshot of Xero API credentials:



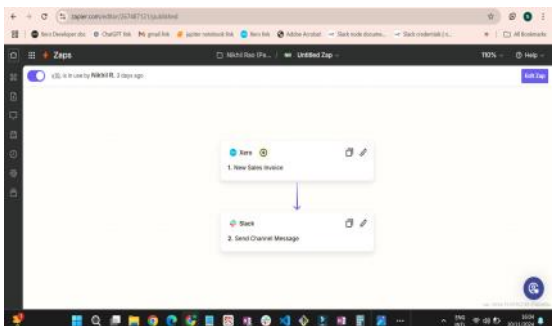
Screenshot of app details in slack



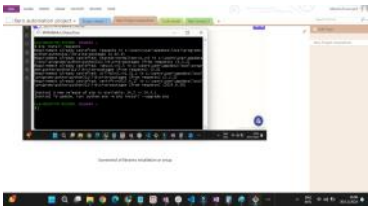
Screenshot of slack webhook URL



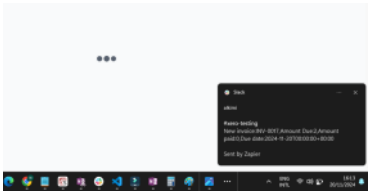
Screenshot of Zap details



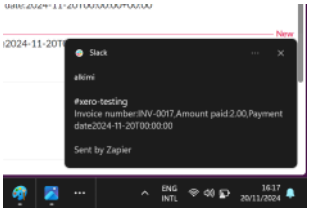
Screenshot of libraries installation or setup



Screenshot of invoice notifications received on slack channel



Screenshot of payment notifications received on slack channel



Code:

```
import requests
import json
import time
import os
from requests.auth import HTTPBasicAuth
from dotenv import load_dotenv

# Load environment variables from a .env file
load_dotenv()

# Xero and Slack credentials from environment variables
XERO_CLIENT_ID = os.getenv("CF15468A353C46588071B43CC1636C67")
XERO_CLIENT_SECRET = os.getenv("UK2LT80ZGozBy7ZHEf-oRP5hD6vgdPo6kQa-5QvOoIpjWEPf")
XERO_TENANT_ID = os.getenv("2820b5d5-fc3b-4f8c-9ecc-213482005224")
XERO_REFRESH_TOKEN = os.getenv("u9fFPDzhxr_vL4rCdO5BRcosP6I97B7zYwxVEnzbs9M")
SLACK_WEBHOOK_URL =
os.getenv("https://hooks.slack.com/services/T07PS4B5ALX/B08160B0VGB/haL4lzeaWwVpOjfyRQBH
BXuI")

print(f"Client ID: {XERO_CLIENT_ID}")
print(f"Client Secret: {XERO_CLIENT_SECRET}")
print(f"Tenant ID: {XERO_TENANT_ID}")
print(f"Refresh Token: {XERO_REFRESH_TOKEN}")
print(f"Slack Webhook URL: {SLACK_WEBHOOK_URL}")

# Temporary storage for pending invoices (in memory for simplicity)
pending_invoices = {}

def get_xero_access_token():
    # Function to refresh and get a new access token from Xero
    token_url = "https://identity.xero.com/connect/token"
    token_data = {
        "grant_type": "refresh_token",
        "refresh_token": XERO_REFRESH_TOKEN,
    }
    response = requests.post(token_url, data=token_data, auth=HTTPBasicAuth(XERO_CLIENT_ID,
XERO_CLIENT_SECRET))
    if response.status_code == 200:
        tokens = response.json()
        return tokens['access_token']
    else:
        raise Exception(f"Failed to get access token: {response.text}")

def fetch_invoices(access_token):
    # Function to fetch invoices from Xero
    invoice_url = f"https://api.xero.com/api.xro/2.0/Invoices"
    headers = {
        "Authorization": f"Bearer {access_token}",
        "Xero-tenant-id": XERO_TENANT_ID,
```

```
    }
    response = requests.get(invoice_url, headers=headers)
    if response.status_code == 200:
        return response.json()["Invoices"]
    else:
        raise Exception(f"Failed to fetch invoices: {response.text}")

def fetch_payments(access_token):
    # Function to fetch payments from Xero
    payment_url = f"https://api.xero.com/api.xro/2.0/Payments"
    headers = {
        "Authorization": f"Bearer {access_token}",
        "Xero-tenant-id": XERO_TENANT_ID,
    }
    response = requests.get(payment_url, headers=headers)
    if response.status_code == 200:
        return response.json()["Payments"]
    else:
        raise Exception(f"Failed to fetch payments: {response.text}")

def send_slack_notification(message):
    payload = {'text': message}
    response = requests.post(SLACK_WEBHOOK_URL, data=json.dumps(payload), headers={'Content-
Type': 'application/json'})
    if response.status_code != 200:
        raise Exception(f"Request to Slack returned an error {response.status_code}, the response is:
\n{response.text}")

def main():
    while True:
        try:
            # Refresh access token
            access_token = get_xero_access_token()

            # Fetch and send notifications for new invoices
            invoices = fetch_invoices(access_token)
            for invoice in invoices:
                if invoice['Status'] == 'AUTHORISED':  # Check if the invoice is authorised
                    pending_invoices[invoice['InvoiceNumber']] = invoice
                    send_slack_notification(f"New Invoice Created: {invoice['InvoiceNumber']}")

            # Fetch payments and update pending invoices status
            payments = fetch_payments(access_token)
            for payment in payments:
                invoice_number = payment['Invoice']['InvoiceNumber']
                if invoice_number in pending_invoices:
                    send_slack_notification(f"Payment Received: {payment['PaymentID']} for Invoice:
{invoice_number}")
                    del pending_invoices[invoice_number]

            # Sleep for a specified time before checking again
            time.sleep(300)  # Check every 5 minutes
        except Exception as e:
            print(f"An error occurred: {e}")
            time.sleep(60)  # Wait for a minute before retrying

if __name__ == "__main__":
```

```
main()
```