import os  
import sys  
  
current\_path = os.path.dirname(os.path.abspath(\_\_file\_\_))  
print(current\_path)  
root\_path = os.path.abspath(os.path.join(current\_path,'..','..','..'))  
print(root\_path)  
sys.path.append(root\_path)  
  
from Xcro.Utils.Escrow.PropertyRead import Property  
import time  
from collections import defaultdict  
import xml.dom.minidom as md2  
import warnings  
  
warnings.filterwarnings("ignore")  
import pandas as pd  
import requests  
import random  
import xml.etree.ElementTree as ET  
  
class CreateAccounts:  
 # Endpoint for API-1  
 endpoint = "https://cbsintuatws.idfcfirstbank.com:7009"  
 # endpoint = "https://10.5.162.78:7113/"  
 resource = "/CreateCustomer/CreateCustomerInterfaceHttpService"  
 url = endpoint + resource  
  
 # Number of requests  
 # num\_requests = 4  
 # print(f"Have selected for {num\_requests} number of requests")  
  
 def extract\_and\_clean\_acc\_num(self,xml\_response):  
 namespaces = {  
 'soapenv': 'http://schemas.xmlsoap.org/soap/envelope/',  
 'v1': 'http://BaNCS.TCS.com/webservice/CreateCustomerInterface/v1',  
 'ban': 'http://TCS.BANCS.Adapter/BANCSSchema',  
 'ns1': 'http://TCS.BANCS.Adapter/BANCSSchema'  
 }  
  
 root = ET.fromstring(xml\_response)  
 acc\_num = root.find(".//ns1:AcctNum", namespaces)  
  
 if acc\_num is not None:  
 acc\_num = acc\_num.text.replace('-', '')  
 return acc\_num  
 else:  
 return None  
  
 def extract\_and\_clean\_cust\_num(self,xml\_response):  
 namespaces = {  
 'soapenv': 'http://schemas.xmlsoap.org/soap/envelope/',  
 'v1': 'http://BaNCS.TCS.com/webservice/CreateCustomerInterface/v1',  
 'ban': 'http://TCS.BANCS.Adapter/BANCSSchema',  
 'ns1': 'http://TCS.BANCS.Adapter/BANCSSchema'  
 }  
  
 root = ET.fromstring(xml\_response)  
 cust\_num\_element = root.find(".//ns1:CustNum", namespaces)  
  
 if cust\_num\_element is not None:  
 cust\_num = cust\_num\_element.text.replace('-', '')  
 return cust\_num  
 else:  
 return None  
  
 def generate\_random\_10\_digit\_number(self):  
 random\_number = ''.join([str(random.randint(0, 9)) for \_ in range(10)])  
 return random\_number  
  
 customers\_list = []  
 accounts\_list = []  
 final\_dict = defaultdict(list)  
  
 # Customer Creation API  
 def getAccounts(self,num\_customers = 1,num\_accounts = 2,balance=[]):  
 custlist = []  
 for i in range(num\_customers):  
 id\_num = self.generate\_random\_10\_digit\_number()  
 customer\_xml = Property.CUSTOMER\_XML  
 file = md2.parse(customer\_xml)  
 file.getElementsByTagName("ban:IdNum")[0].childNodes[0].nodeValue = id\_num  
 with open(customer\_xml, "w") as fs:  
 fs.write(file.toxml())  
 fs.close()  
 contents = open(customer\_xml).read()  
 # print(contents)  
  
 headers = {  
 'Content-Type': 'text/xml',  
 }  
 response = requests.post(self.url, data=contents, headers=headers, verify=False)  
 time.sleep(1)  
 if response.status\_code == 200:  
 result = self.extract\_and\_clean\_cust\_num(response.text)  
 if result is not None:  
 print(f"{result}")  
 self.customers\_list.append(result)  
 custlist.append(result)  
 else:  
 print(response.text)  
  
 # Accounts and Balance Update API  
 print(self.customers\_list)  
 myacc\_list = []  
 for customer in custlist:  
 for i in range(num\_accounts):  
 new\_request\_body = f"""  
 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v1="http://BaNCS.TCS.com/webservice/CreateDepositAccountInterface/v1" xmlns:ban="http://TCS.BANCS.Adapter/BANCSSchema">  
 <soapenv:Header/>  
 <soapenv:Body>  
 <v1:createDepositAccount>  
 <!--Optional:-->  
 <DepAcctCreateRq>  
 <ban:RqHeader>  
 <ban:InstNum>3</ban:InstNum>  
 <ban:BrchNum>10204</ban:BrchNum>  
 <ban:TellerNum>1101</ban:TellerNum>  
 <ban:Flag4>W</ban:Flag4>  
 <ban:Flag5>Y</ban:Flag5>  
 <ban:UUIDSource></ban:UUIDSource>  
 <ban:UUIDNUM></ban:UUIDNUM>  
 <ban:UUIDSeqNo></ban:UUIDSeqNo>  
 </ban:RqHeader>  
 <ban:Data>  
 <ban:CustNum>{customer}</ban:CustNum>  
 <ban:IdNum></ban:IdNum>  
 <ban:IdTyp1></ban:IdTyp1>  
 <ban:CustName></ban:CustName>  
 <ban:Addr1></ban:Addr1>  
 <ban:Addr2></ban:Addr2>  
 <ban:FileInd></ban:FileInd>  
 <ban:NomReq1>1</ban:NomReq1>  
 <ban:ResdntStat></ban:ResdntStat>  
 <ban:Natlty>IN</ban:Natlty>  
 <ban:NomCIF1></ban:NomCIF1>  
 <ban:JDCCNum1></ban:JDCCNum1>  
 <ban:PostCode></ban:PostCode>  
 <ban:Prod>1000</ban:Prod>  
 <ban:Typ>3001</ban:Typ>  
 <ban:PmtMthd>R</ban:PmtMthd>  
 <ban:LangCode></ban:LangCode>  
 <ban:DtInd></ban:DtInd>  
 <ban:MailInd></ban:MailInd>  
 <ban:NoticeInd></ban:NoticeInd>  
 <ban:VarIntRate></ban:VarIntRate>  
 <ban:Freq2></ban:Freq2>  
 <ban:TermLgth></ban:TermLgth>  
 <ban:TermBasis></ban:TermBasis>  
 <ban:Amt2></ban:Amt2>  
 <ban:TierGrpId></ban:TierGrpId>  
 <ban:CustRisk></ban:CustRisk>  
 <ban:CrossBrdrRisk></ban:CrossBrdrRisk>  
 <ban:AcctCrossBrdrRisk>ZZ</ban:AcctCrossBrdrRisk>  
 <ban:SecuInd>Z</ban:SecuInd>  
 <ban:TimeBand>ZZ</ban:TimeBand>  
 <ban:NoticeProc>0</ban:NoticeProc>  
 <ban:AcctSegCode>189</ban:AcctSegCode>  
 <ban:SMSFlag>Y</ban:SMSFlag>  
 <ban:VendorCode></ban:VendorCode>  
 <ban:TermDays></ban:TermDays>  
 <ban:TermMnths></ban:TermMnths>  
 <ban:TermYrs></ban:TermYrs>  
 <ban:AcctOpenDt></ban:AcctOpenDt>  
 <ban:BkdtReCalcInd></ban:BkdtReCalcInd>  
 <ban:TrnDt></ban:TrnDt>  
 <ban:MatDt></ban:MatDt>  
 <ban:BkdtIntRecalcReq></ban:BkdtIntRecalcReq>  
 <ban:RDInstlmntAmt>0</ban:RDInstlmntAmt>  
 <ban:Freqn1>0</ban:Freqn1>  
 <ban:MnthlyRDInstlmntDueDay>0</ban:MnthlyRDInstlmntDueDay>  
 <ban:MOD></ban:MOD>  
 <ban:LinkAcctTyp></ban:LinkAcctTyp>  
 <ban:TagNum></ban:TagNum>  
 <ban:DdoCode></ban:DdoCode>  
 <ban:PrimAcctNum></ban:PrimAcctNum>  
 <ban:SubCat></ban:SubCat>  
 <ban:LnkdBrchCode></ban:LnkdBrchCode>  
 <ban:InstlmntCnt></ban:InstlmntCnt>  
 <ban:PrdBal></ban:PrdBal>  
 <ban:IrreglrCnt></ban:IrreglrCnt>  
 <ban:ExtensionCnt></ban:ExtensionCnt>  
 <ban:AcctStat></ban:AcctStat>  
 <ban:Prmpt>0</ban:Prmpt>  
 <ban:OnOffId1></ban:OnOffId1>  
 <ban:OnOffId3></ban:OnOffId3>  
 <ban:OnOffId4></ban:OnOffId4>  
 <ban:OnOffId5>1</ban:OnOffId5>  
 <ban:OnOffId6></ban:OnOffId6>  
 <ban:OnOffId7></ban:OnOffId7>  
 <ban:OnOffId8></ban:OnOffId8>  
 <ban:OnOffId9></ban:OnOffId9>  
 <ban:OnOffId10></ban:OnOffId10>  
 <ban:OnOffId11></ban:OnOffId11>  
 <ban:OnOffId12></ban:OnOffId12>  
 <ban:OnOffId13></ban:OnOffId13>  
 <ban:OnOffId15></ban:OnOffId15>  
 <ban:OnOffId16></ban:OnOffId16>  
 <ban:OnOffId17></ban:OnOffId17>  
 <ban:OnOffId18></ban:OnOffId18>  
 <ban:OnOffId19></ban:OnOffId19>  
 <ban:OnOffId20></ban:OnOffId20>  
 <ban:OnOffId21></ban:OnOffId21>  
 <ban:OnOffId22></ban:OnOffId22>  
 <ban:OnOffId23></ban:OnOffId23>  
 <ban:OnOffId24></ban:OnOffId24>  
 <ban:OnOffId25></ban:OnOffId25>  
 <ban:OnOffId26></ban:OnOffId26>  
 <ban:OnOffId27></ban:OnOffId27>  
 <ban:OnOffId28></ban:OnOffId28>  
 <ban:OnOffId29></ban:OnOffId29>  
 <ban:OnOffId30></ban:OnOffId30>  
 <ban:OnOffId31></ban:OnOffId31>  
 <ban:OnOffId32></ban:OnOffId32>  
 <ban:Filler1></ban:Filler1>  
 <ban:Filler2></ban:Filler2>  
 <ban:Filler3></ban:Filler3>  
 <ban:Filler4></ban:Filler4>  
 <ban:Filler5></ban:Filler5>  
 <ban:Filler6></ban:Filler6>  
 <ban:Filler7></ban:Filler7>  
 <ban:Filler8></ban:Filler8>  
 <ban:Filler9></ban:Filler9>  
 <ban:Filler10></ban:Filler10>  
 <ban:Filler11></ban:Filler11>  
 <ban:Filler12></ban:Filler12>  
 <ban:Filler13></ban:Filler13>  
 <ban:Filler14></ban:Filler14>  
 <ban:Filler15></ban:Filler15>  
 <ban:Filler16></ban:Filler16>  
 <ban:Filler17></ban:Filler17>  
 <ban:Filler18></ban:Filler18>  
 <ban:Filler19></ban:Filler19>  
 <ban:Filler20></ban:Filler20>  
 <ban:Filler21></ban:Filler21>  
 <ban:Filler22></ban:Filler22>  
 <ban:Filler23></ban:Filler23>  
 <ban:Filler24></ban:Filler24>  
 <ban:Filler25></ban:Filler25>  
 <ban:Filler26></ban:Filler26>  
 <ban:Filler27></ban:Filler27>  
 <ban:Filler28></ban:Filler28>  
 <ban:Filler29></ban:Filler29>  
 <ban:Filler30></ban:Filler30>  
 <ban:Filler31></ban:Filler31>  
 <ban:Filler32></ban:Filler32>  
 <ban:NomName></ban:NomName>  
 <ban:Minor></ban:Minor>  
 <ban:GuardnName></ban:GuardnName>  
 <ban:NomTyp></ban:NomTyp>  
 <ban:AahGRefNum></ban:AahGRefNum>  
 <ban:Amt8></ban:Amt8>  
 <ban:Amt7></ban:Amt7>  
 <ban:FundSrc></ban:FundSrc>  
 <ban:AcctNum10></ban:AcctNum10>  
 <ban:InstruNum></ban:InstruNum>  
 <ban:CityCodeSof></ban:CityCodeSof>  
 <ban:BankCodeSof></ban:BankCodeSof>  
 <ban:BranchCodeSof></ban:BranchCodeSof>  
 <ban:InstruTypeSof></ban:InstruTypeSof>  
 <ban:InstruDateSof></ban:InstruDateSof>  
 <ban:ClearingDateSof></ban:ClearingDateSof>  
 <ban:ClearingTypeSof></ban:ClearingTypeSof>  
 <ban:MatAmt></ban:MatAmt>  
 <ban:TdrPayAmt></ban:TdrPayAmt>  
 <ban:IntRate></ban:IntRate>  
 <ban:IntrnNomNum1></ban:IntrnNomNum1>  
 <ban:AutoClosureFlag></ban:AutoClosureFlag>  
 <ban:PmtMthd1></ban:PmtMthd1>  
 <ban:CrDrAcctFlag></ban:CrDrAcctFlag>  
 <ban:XferAcctNum></ban:XferAcctNum>  
 <ban:TDSExmptnFlag></ban:TDSExmptnFlag>  
 <ban:NoSig></ban:NoSig>  
 <ban:SignVldNum></ban:SignVldNum>  
 <ban:ChqDt></ban:ChqDt>  
 <ban:CMPNum></ban:CMPNum>  
 <ban:InstruTyp></ban:InstruTyp>  
 <ban:ChqAmt1></ban:ChqAmt1>  
 <ban:ChqAmt2></ban:ChqAmt2>  
 <ban:ChqAmt3></ban:ChqAmt3>  
 <ban:ChqAmt4></ban:ChqAmt4>  
 <ban:ChqAmt5></ban:ChqAmt5>  
 <ban:ChqAmt6></ban:ChqAmt6>  
 <ban:Comsn></ban:Comsn>  
 <ban:ChqAcctCurrCode1></ban:ChqAcctCurrCode1>  
 <ban:TtlLmt></ban:TtlLmt>  
 <ban:RenewalPrinCode></ban:RenewalPrinCode>  
 <ban:ThrsldBal></ban:ThrsldBal>  
 <ban:RsltntBal></ban:RsltntBal>  
 <ban:MaxAccumSweep></ban:MaxAccumSweep>  
 <ban:SweepMul></ban:SweepMul>  
 <ban:RbiClassCatCr>180</ban:RbiClassCatCr>  
 <ban:RbiClassCatDr>350</ban:RbiClassCatDr>  
 <ban:RbiClassCodeDr>383</ban:RbiClassCodeDr>  
 <ban:Form60Flag>1</ban:Form60Flag>  
 <ban:LgCode></ban:LgCode>  
 <ban:LcCode></ban:LcCode>  
 </ban:Data>  
 </DepAcctCreateRq>  
 </v1:createDepositAccount>  
 </soapenv:Body>  
 </soapenv:Envelope>"""  
  
 # End point for API-2  
 second\_api\_url = 'https://cbsintuatws.idfcfirstbank.com:7009/'  
 # second\_api\_url = 'https://10.5.162.78:7113'  
 second\_api\_resource = '/CreateDepositAccount/CreateDepositAccountInterfaceHttpService'  
  
 response2 = requests.post(second\_api\_url + second\_api\_resource, data=new\_request\_body,  
 headers={'Content-Type': 'text/xml'}, verify=False)  
  
 if response2.status\_code == 200:  
 accountnum = self.extract\_and\_clean\_cust\_num(response2.text)  
 print(response2.text)  
 self.accounts\_list.append(accountnum)  
 myacc\_list.append(accountnum)  
 self.final\_dict[customer].append(accountnum)  
 print(accountnum)  
 print(i)  
  
 curr\_bal = 0.0  
 req\_balance = balance[i]  
  
 while(curr\_bal+1000000< float(req\_balance)):  
  
 curr\_bal = curr\_bal + 1000000  
 # End point for API-3  
 try:  
 url = 'https://cbsintuatws.idfcfirstbank.com:7009/DepositCashAccount/DepositCashAccountInterfaceHttpService'  
 # url = 'https://10.5.162.78:7113/DepositCashAccount/DepositCashAccountInterfaceHttpService'  
 myxml = Property.BALANCE\_UPDATE\_XML  
 file = md2.parse(myxml)  
 file.getElementsByTagName("ban:AcctNum")[0].childNodes[0].nodeValue = accountnum  
 file.getElementsByTagName("ban:Amt")[0].childNodes[0].nodeValue = 1000000  
 file.getElementsByTagName("ban:UUIDNUM")[0].childNodes[0].nodeValue = random.randint(1000, 9999)  
 with open(myxml, "w") as fs:  
 fs.write(file.toxml())  
 fs.close()  
 contents = open(myxml).read()  
 headers = {'Content-Type': 'text/xml'}  
 res = requests.post(url=url, data=contents, headers=headers, verify=False)  
 print(f"Balance status Code: {res.status\_code}")  
 print(res.content)  
 except requests.exceptions.HTTPError as errh:  
 print("Http Error:", errh)  
  
 # Remaining amount  
 try:  
 url = 'https://cbsintuatws.idfcfirstbank.com:7009/DepositCashAccount/DepositCashAccountInterfaceHttpService'  
 # url = 'https://10.5.162.78:7113/DepositCashAccount/DepositCashAccountInterfaceHttpService'  
 myxml = Property.BALANCE\_UPDATE\_XML  
 file = md2.parse(myxml)  
 file.getElementsByTagName("ban:AcctNum")[0].childNodes[0].nodeValue = accountnum  
 file.getElementsByTagName("ban:Amt")[0].childNodes[0].nodeValue = float(req\_balance)-curr\_bal  
 file.getElementsByTagName("ban:UUIDNUM")[0].childNodes[0].nodeValue = random.randint(1000, 9999)  
 with open(myxml, "w") as fs:  
 fs.write(file.toxml())  
 fs.close()  
 contents = open(myxml).read()  
 headers = {'Content-Type': 'text/xml'}  
 res = requests.post(url=url, data=contents, headers=headers, verify=False)  
 print(f"Balance status Code: {res.status\_code}")  
 print(res.content)  
 except requests.exceptions.HTTPError as errh:  
 print("Http Error:", errh)  
  
 else:  
 print("Account Creation API request failed.")  
 print(response2.text)  
  
 print(self.final\_dict)  
 return myacc\_list