**I created an AI email marketing agent using Langchain and AI21 jurrasic J2 model in python.**

[[Sangeeth Joseph - The AI dev](https://medium.com/@sangeeth123sj?source=post_page-----bd5a02862aa8--------------------------------)](https://medium.com/@sangeeth123sj?source=post_page-----bd5a02862aa8--------------------------------)

[Sangeeth Joseph - The AI dev](https://medium.com/@sangeeth123sj?source=post_page-----bd5a02862aa8--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F34f2ddc7dbd6&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40sangeeth123sj%2Fi-created-an-ai-email-marketing-agent-using-langchain-and-ai21-jurrasic-j2-model-in-python-bd5a02862aa8&user=Sangeeth+Joseph+-+The+AI+dev&userId=34f2ddc7dbd6&source=post_page-34f2ddc7dbd6----bd5a02862aa8---------------------post_header-----------)

6 min read

·

Oct 7

8



It all begins with a thought..

Recently I was starting to use langchain, which is a good framework for creating AI agents. To put it simply, langchain is tool which you can use to link AI LLM APIs like openai gpt or ai21 and so on with url reading functionality or document reading functionality or reading plain text and so on. It enables us to chain such functionalities together one by one to form an workflow creating AI chatbots, AI search engines, AI agents or anything AI that you wish to.

**The Idea**

But then I came to a thought, why now create a python script to read a csv file of contacts lists containing the name email and description of the lead/contact/recipient and send customised AI generated emails to each of them based on an intention we input to the script. My intention was to automate outreaches like cold emailing, email marketing customised for each individual contact, email updates etc.

The possibilities that I saw in using AI like this was to increase conversions, increase email engagement due to customisation for individual customers.

**The implementation**

Like the great creator of linux said:  
**“Talk is cheap. Show me the code.” — Linus Torvalds**

So let’s dive into this python script.

**Installation and Configuration:**

Create a new project folder and setup a virtual environment.

mkdir email\_agent #creating the folder/directory  
cd email\_agent #entering the directory  
python3 -m venv env #create the environment  
source env/bin/activate #activate the environment

**Installing the dependencies into the environment:**

pip install langchain  
pip install openpyxl  
pip install python-dotenv

Here **Openpyxl** is the library for reading,writing and handling excel files (here I’m using .xlsx file)

Dotenv is used for keeping api keys on a seperate .env file and importing them in the script file.

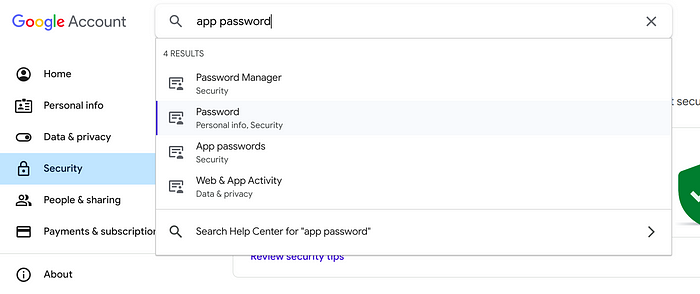
now let’s create the .env file

ai21\_token=<your ai21 key>  
sender\_email=sangeeth123sj@gmail.com  
email\_password=<your google account app password>

You can create an account on AI21 and get their api key to access their Jurassic model.

And you can create an app password by going into google account > search ‘app password’

and select the app password option that comes in the drop down.  
And create the app password and copy it to the .env file



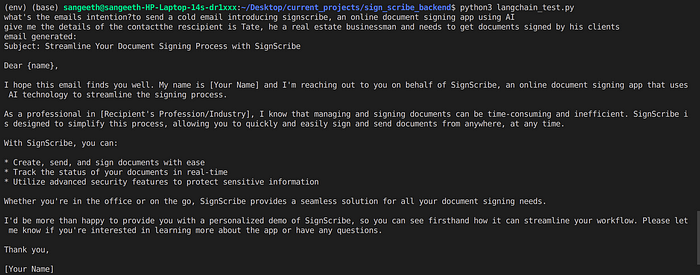
google app password search

**The Code**

Now let’s create a new file in the directory main.py.

**Step 1: Implementing the langchain AI21 email generation**

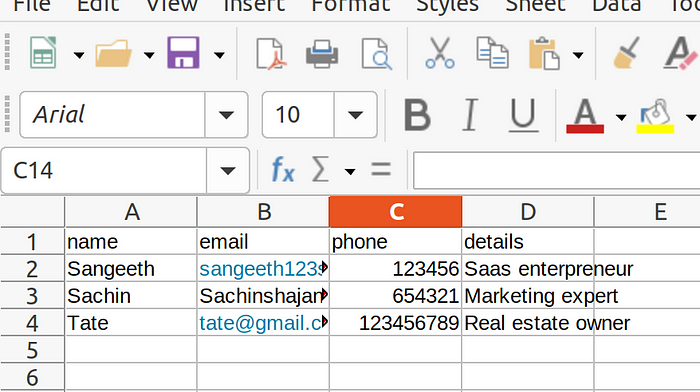
import os  
from dotenv import load\_dotenv  
from langchain.llms import AI21  
from langchain.prompts import PromptTemplate  
from langchain.chains import LLMChain  
# Load environment variables from .env  
load\_dotenv()  
#Taking the intention input from the user of the script  
intention = input("what's the emails intention?")  
details = input("give me the details of the contact")  
# Set up the LLM  
template = """Create me an email with the Intention: {intention}  
 the details of the recipient for the context of the mail is {details}  
 Create me an email for this recipient and mention in the email how the intention I mentioned matches the details of the recipient like his professions or business that I mentioned in the details"""  
#creating the PromptTemplate of langchain using the input intention and details  
prompt = PromptTemplate(template=template, input\_variables=["intention", "details"])  
  
# Initialize LLM using ai21 api key  
llm = AI21(ai21\_api\_key=os.getenv("ai21\_token"))  
# adding the prompt (PromptTemplate) and llm (AI21) to the llm chain  
llm\_chain = LLMChain(prompt=prompt, llm=llm)  
  
#getting the ai llm generated response  
message = llm\_chain.run(intention=intention, details="recipient email is {email} and name of recipient: {name} other details of recipient to use in the mail content:{details}")  
print(f"email generated: {message}")



AI email generation agent output

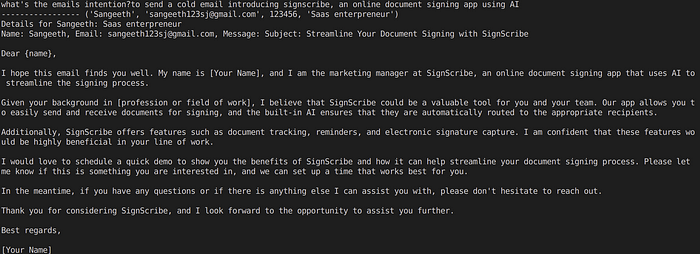
**Step 2: Adding in the excel file reading functionality for reading a list of contacts.**

Now we need to create an excel file in the same directory with .xlsx extension  
with the first row having headings as : name, email, phone, details and the corresponding values of each contact in the subsequent rows.



And then let’s modify the script adding the openpyxl library to read each contact and create messages for the same using langchain and AI21 model.

import os  
from dotenv import load\_dotenv  
from langchain.llms import AI21  
from langchain.prompts import PromptTemplate  
from langchain.chains import LLMChain  
from openpyxl import load\_workbook  
  
# Load environment variables from .env  
load\_dotenv()  
  
intention = input("what's the emails intention?")  
# Set up the LLM  
template = """Create me an email with the Intention: {intention}  
 the details of the recipient for the context of the mail is {details}  
 Create me an email for this recipient and mention in the email how the intention I mentioned matches the details of the recipient like his professions or business that I mentioned in the details"""  
prompt = PromptTemplate(template=template, input\_variables=["intention", "details"])  
  
# Initialize LLM  
llm = AI21(ai21\_api\_key=os.getenv("ai21\_token"))  
llm\_chain = LLMChain(prompt=prompt, llm=llm)  
  
# Load Excel file  
excel\_file\_path = "./contacts.xlsx"  
  
#loading the contact.  
wb = load\_workbook(excel\_file\_path)  
sheet = wb.active  
row\_counter = 2 # Start from row 2, assuming headers in row 1  
  
# Get user inputs for email intention  
for row in sheet.iter\_rows(min\_row=row\_counter, values\_only=True):  
 print("-----------------",row)  
 name, email, phone, details = row  
 print(f"Details for {name}: {details}")  
 message = llm\_chain.run(intention=intention, details="recipient email is {email} and name of recipient: {name} other details of recipient to use in the mail content:{details}")  
 message = message.strip() # Remove any leading/trailing whitespaces  
 print(f"Name: {name}, Email: {email}, Message: {message}")

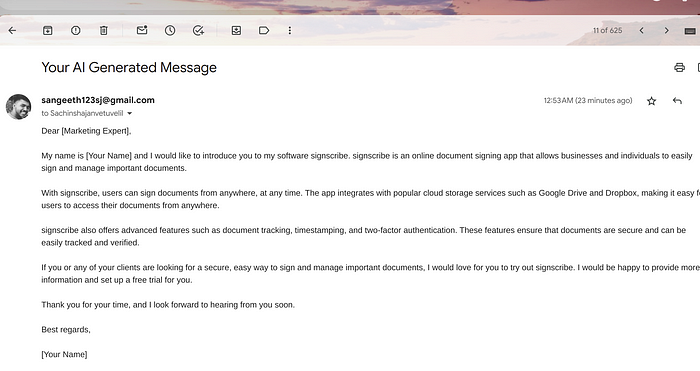


output for python script reading contact details from excel sheet and generating AI email content for them

**Step3: Adding automatic email sending functionality using smtp(Simple mail transfer Protocol) in python**

In this last step we are automating the sending of emails too with the AI generated email content. Here we are using our gmail to automatically send the emails. Remember the email and google account app password of that email we saved in the .env file. We access them in the code to send the mail.

import os  
from dotenv import load\_dotenv  
from langchain.llms import AI21  
from langchain.prompts import PromptTemplate  
from langchain.chains import LLMChain  
from openpyxl import load\_workbook  
import smtplib  
from email.mime.multipart import MIMEMultipart  
from email.mime.text import MIMEText  
  
# Load environment variables from .env  
load\_dotenv()  
  
intention = input("what's the emails intention?")  
# Set up the LLM  
template = """Create me an email with the Intention: {intention}  
 the details of the recipient for the context of the mail is {details}  
 Create me an email for this recipient and mention in the email how the intention I mentioned matches the details of the recipient like his professions or business that I mentioned in the details"""  
prompt = PromptTemplate(template=template, input\_variables=["intention", "details"])  
  
# Initialize LLM  
llm = AI21(ai21\_api\_key=os.getenv("ai21\_token"))  
llm\_chain = LLMChain(prompt=prompt, llm=llm)  
  
# Load Excel file  
excel\_file\_path = "./contacts.xlsx"  
  
wb = load\_workbook(excel\_file\_path)  
sheet = wb.active  
row\_counter = 2 # Start from row 2, assuming headers in row 1  
  
# Get user inputs for email intention  
for row in sheet.iter\_rows(min\_row=row\_counter, values\_only=True):  
 print("-----------------",row)  
 name, email, phone, details = row  
 print(f"Details for {name}: {details}")  
 message = llm\_chain.run(intention=intention, details="recipient email is {email} and name of recipient: {name} other details of recipient to use in the mail content:{details}")  
 message = message.strip() # Remove any leading/trailing whitespaces  
 print(f"Name: {name}, Email: {email}, Message: {message}")  
   
 # Send the email  
 # initializing the email server  
 msg = MIMEMultipart()  
 msg['From'] = os.getenv("sender\_email")  
 msg['To'] = email  
 msg['Subject'] = "Your AI Generated Message"  
  
 msg.attach(MIMEText(message, 'plain'))  
  
 try:  
 server = smtplib.SMTP('smtp.gmail.com', 587)  
 server.starttls()  
 server.login(os.getenv("sender\_email"), os.getenv("email\_password"))  
 server.sendmail(os.getenv("sender\_email"), email, msg.as\_string())  
 server.close()  
 print(f"Email sent to {email}")  
 except Exception as e:  
 print(f"Failed to send email to {email}. Error: {str(e)}")  
  
# Save the modified Excel file  
wb.save(excel\_file\_path)  
print("Emails sent.")



The screenshot of an automated AI email sent by using AI automation with langchain in python from a contact list excel file

**That’s it. This is a proof of concept for the idea!**

AI21 models don’t perform as well as openai models in this use case for using the contact’s details as the context for the email.It is easily possible to switch to OpenAI model in the langchain, which will improve the outputs.

Thanks for reading till now. I’m Sangeeth Joseph, a fullstack freelance developer on upwork with focus on AI integrations. Follow me for more AI ideas, updates and content.