

## Lesson 3 Demo 10

### AutoGen for Automated Contract Review and Compliance Check

**Objective:** To develop an AutoGen-powered contract review system

Legal teams spend significant time reviewing contracts to ensure compliance, mitigate risks, and suggest revisions. This manual process is time-consuming, error-prone, and inconsistent. An automated contract review system can streamline the process, improving efficiency, accuracy, and compliance.

The goal is to develop an AutoGen-powered contract review system that:

1. Extracts key clauses from contracts for structured analysis
2. Checks compliance with regulations like GDPR, corporate policies, and industry standards
3. Identifies potential risks, missing clauses, and ambiguous terms
4. Suggests revision to improve clarity and ensure compliance

This system aims to reduce review time, improve accuracy, and ensure legal consistency across contracts.

#### Prerequisites:

1. Create a virtual environment
2. Activate the virtual environment
3. Install dependencies

#### Steps to be followed:

1. Setup the environment
2. Extract text using OCR
3. Define AutoGen agents
4. Create the Streamlit UI
5. Run the webapp

## Step 1: Set up the environment

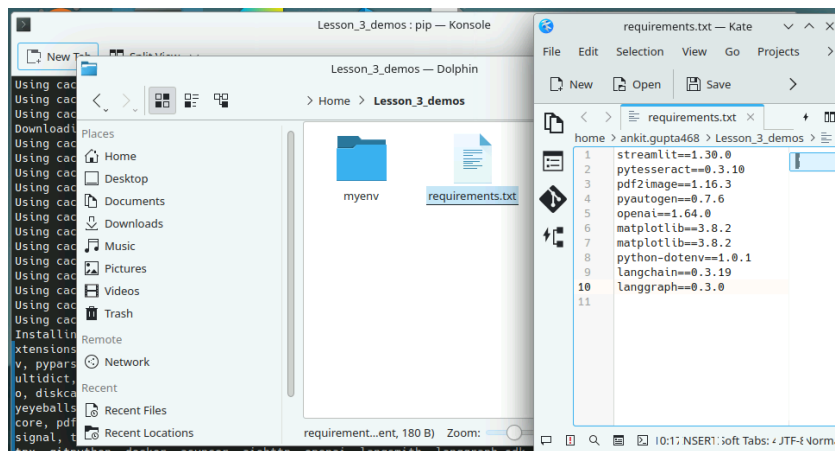
- 1.1 Open command prompt and go to the “**Lesson\_3\_demos**” folder (which we created in Demo\_1) using the command given below:

```
mkdir Lesson_3_demos (not needed if the folder is already created in Demo1)
cd Lesson_3_demos
```

- 1.2 After this, activate the virtual environment using the command below:

```
python3 -m venv venv (not needed if the virtual env. is already created in Demo1)
source venv/bin/activate
```

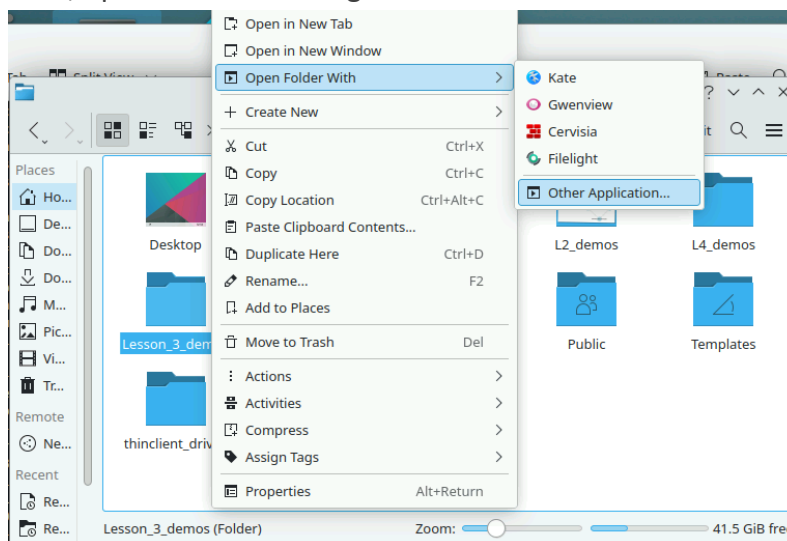
- 1.3 Now, create a `requirements.txt` file inside the folder with required libraries (not needed if already done in Demo1):



- 1.4 Install all the required libraries using the command below:

```
pip install -r requirements.txt (not needed if already done in Demo1)
```

- 1.5 Now, open the folder using VS Code editor:



- 1.6 After this, open a new Python file using the “**New File**” option and name it as “**Demo10**”.

## Step 2: Extract text using OCR

- 2.1 Create a temporary File: Save the uploaded PDF to a temporary file
- 2.2 Get temporary file path: Store the file path for further processing
- 2.3 Convert PDF to images: Use `convert_from_path` to convert PDF pages into images
- 2.4 Initialize text storage: Create an empty string to store extracted text
- 2.5 Extract text using OCR: Iterate through each image and apply `pytesseract.image_to_string` to extract text
- 2.6 Return extracted text: Concatenate and return the extracted text from all pages

```
import streamlit as st
import pytesseract
from pdf2image import convert_from_path
from autogen import ConversableAgent, UserProxyAgent
import tempfile
import os

# Ensure Tesseract is correctly set up (update path if necessary)
# pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe" # Windows users
pytesseract.pytesseract.tesseract_cmd = '/usr/bin/tesseract'

# Step 1: Function to extract text from a PDF using OCR
def extract_text_from_pdf(uploaded_file):
    with tempfile.NamedTemporaryFile(delete=False, suffix=".pdf") as temp_file:
        temp_file.write(uploaded_file.read()) # Save uploaded PDF to a temp file
        temp_file_path = temp_file.name # Get temp file path

    try:
        images = convert_from_path(temp_file_path) # Convert PDF to images
        text = "\n".join(pytesseract.image_to_string(img, config="--psm 6") for img in images)
    # Perform OCR
    except Exception as e:
        st.error(f"Error processing PDF: {e}")
        text = ""

    os.remove(temp_file_path) # Cleanup temp file
    return text
```

## Step 3: Define AutoGen agents

3.1 Create a UserProxyAgent to act as the sender of the message

3.2 Create multiple ConversableAgent instances, each specializing in a different aspect of contract analysis

3.3 The agents include:

- a) Document ingestor: Extracts key clauses from the contract
- b) Compliance checker: Verifies compliance with GDPR, corporate policies, and industry standards
- c) Risk assessor: Identifies potential risks, ambiguities, and missing clauses
- d) Revision recommender: Suggests modifications to improve clarity, reduce risks, and ensure compliance

```
# Define AutoGen Agents
user_agent = UserProxyAgent(name="User", code_execution_config={"use_docker": False})

doc_ingestor = ConversableAgent(
    name="Document_Ingestor",
    system_message="Extract key clauses from a contract document."
)

compliance_checker = ConversableAgent(
    name="Compliance_Checker",
    system_message="Review the contract for compliance with GDPR, corporate policies, and industry standards."
)

risk_assessor = ConversableAgent(
    name="Risk_Assessor",
    system_message="Analyze the contract and highlight potential risks, missing clauses, and ambiguities."
)

revision_recommender = ConversableAgent(
    name="Revision_Recommender",
    system_message="Suggest contract modifications to improve clarity, reduce risks, and ensure compliance."
)
```

## Step 4: Create Streamlit UI

- 4.1 Create a user-friendly web interface using Streamlit
- 4.2 Provide a file uploader for users to upload a contract PDF
- 4.3 Display extracted text in UI
- 4.4 Once uploaded, extract and display the contract text in a scrollable text area. If the **Analyze Contract** button is clicked:
  - a) The system sends the extracted text to different AutoGen agents for analysis.
  - b) Each agent processes the contract based on its role (for example, compliance check and risk assessment).
- 4.5 Display AI-generated insights in a structured format:
  - Extracted clauses: Display in an info box
  - Compliance issues: Display in a warning box
  - Risk analysis: Display in an error box
  - Suggested revisions: Display in a success box

(code on next page)

```

# Step 3: Create Streamlit UI
st.title("AutoGen-Powered Contract Review System")
st.write("Upload a contract PDF to analyze compliance, risks, and suggested revisions.")

# Step 3.1: File Upload Mechanism
uploaded_file = st.file_uploader("Upload Contract PDF", type=["pdf"])

if uploaded_file:
    st.write("Extracting text from PDF...")
    contract_text = extract_text_from_pdf(uploaded_file) # Extract text from PDF

    # 3.2 Display extracted text in UI
    st.subheader("Extracted Contract Text")
    st.text_area("Contract Content", contract_text, height=200)

    if st.button("Analyze Contract"):
        with st.spinner("Processing..."):
            extracted_clauses = user_agent.initiate_chat(
                recipient=doc_ingestor,
                message=f"Extract key clauses from the following contract:\n{contract_text}"
            )
            compliance_issues = user_agent.initiate_chat(
                recipient=compliance_checker,
                message=f"Review this contract for compliance with GDPR and corporate legal policies:\n{contract_text}"
            )
            risk_analysis = user_agent.initiate_chat(
                recipient=risk_assessor,
                message=f"Analyze the contract and highlight risks or missing clauses:\n{contract_text}"
            )
            revisions = user_agent.initiate_chat(
                recipient=revision_recommender,
                message=f"Suggest modifications to improve contract clarity and compliance:\n{contract_text}"
            )

        # 3.3 : Display Analysis Results
        st.subheader("Contract Analysis Results")

        with st.expander("**Extracted Clauses**"):
            st.info(extracted_clauses)

        with st.expander("**Compliance Issues**"):
            st.warning(compliance_issues)

        with st.expander("**Risk Analysis**"):
            st.error(risk_analysis)

        with st.expander("**Suggested Revisions**"):
            st.success(revisions)

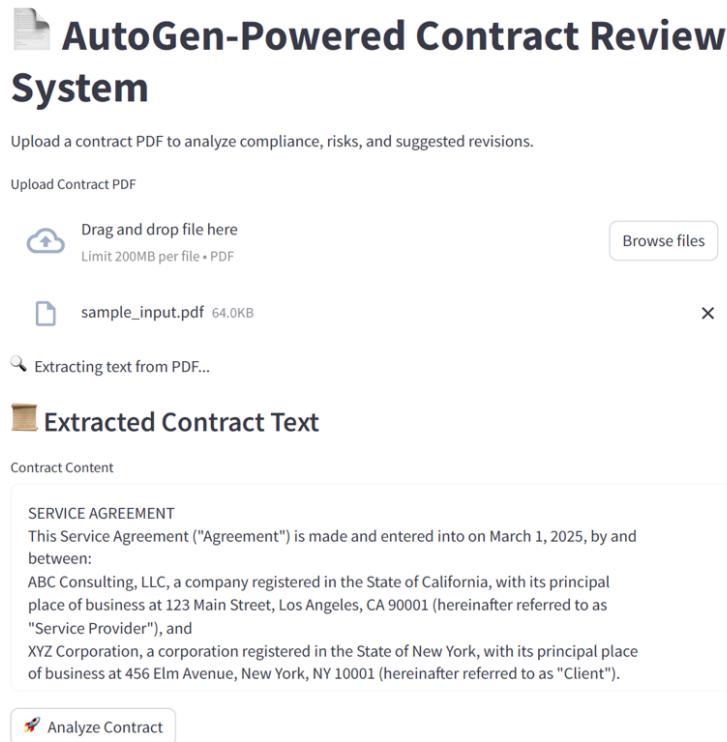
```

## Step 5: Run the webapp

- 5.1 Save the file and then run the streamlit webapp from command prompt using the command given below:

**streamlit run Demo10.py**

Output:



The screenshot shows a web application titled "AutoGen-Powered Contract Review System". Below the title, it says "Upload a contract PDF to analyze compliance, risks, and suggested revisions." and "Upload Contract PDF". There is a file upload area with a cloud icon and text "Drag and drop file here" and "Limit 200MB per file • PDF". A "Browse files" button is to the right. Below this, a file named "sample\_input.pdf" (64.0KB) is shown with a close button (X). A status message "Extracting text from PDF..." is displayed. Below that, a section titled "Extracted Contract Text" shows the following content:

Contract Content

SERVICE AGREEMENT  
This Service Agreement ("Agreement") is made and entered into on March 1, 2025, by and between:  
ABC Consulting, LLC, a company registered in the State of California, with its principal place of business at 123 Main Street, Los Angeles, CA 90001 (hereinafter referred to as "Service Provider"), and  
XYZ Corporation, a corporation registered in the State of New York, with its principal place of business at 456 Elm Avenue, New York, NY 10001 (hereinafter referred to as "Client").

An "Analyze Contract" button is at the bottom.

By following the above-mentioned steps, you have successfully showcased how AutoGen-powered automation can streamline contract analysis by integrating OCR-based text extraction, AI-driven document processing, and an interactive web interface using Streamlit. The system automatically extracts key contract clauses, checks compliance, assesses risks, and suggests revisions—all without manual intervention.

By leveraging ConversableAgents, this automation ensures efficiency by eliminating the need for manual contract reviews, improves accuracy by identifying risks and compliance gaps, and offers scalability to handle large volumes of documents.

This implementation highlights how AutoGen can be used to build intelligent, automated workflows, reducing effort while improving decision-making in contract management.

## Appendix:

### 1. Required python libraries - Install the necessary dependencies using pip:

```
pip install -r requirements.txt
```

### 2. Tesseract OCR

**Windows:** Download and install Tesseract OCR.

**Linux (Ubuntu/Debian):** Install Tesseract using:

```
sudo apt install tesseract-ocr
```

**Mac: Install via Homebrew:**

```
brew install tesseract
```

### 3. Poppler for PDF processing

**Windows:** Download poppler from this [link](#) and add its bin folder to the system PATH.

**Linux (Ubuntu/Debian):** Install using:

```
sudo apt install poppler-utils
```

**Mac: Install via Homebrew:**

```
brew install poppler
```