LABORATORY REPORT

Application Development Lab (CS33002)

B. Tech Program in ECSc

Submitted By

Name:- Akriti Patro

Roll No: 2230232

Kalinga Institute of Industrial Technology
(Deemed to be University)
Bhubaneswar, India

Spring 2024-2025

Table of Content

Experiment Number	4
Experiment Title	Conversational Chatbot with Any Files
Date of Experiment	28/01/25
Date of Submission	09/02/25

1. **Objective:-** To build a chatbot capable of answering queries from an uploaded PDF/Word/Excel document.

2. **Procedure:-**

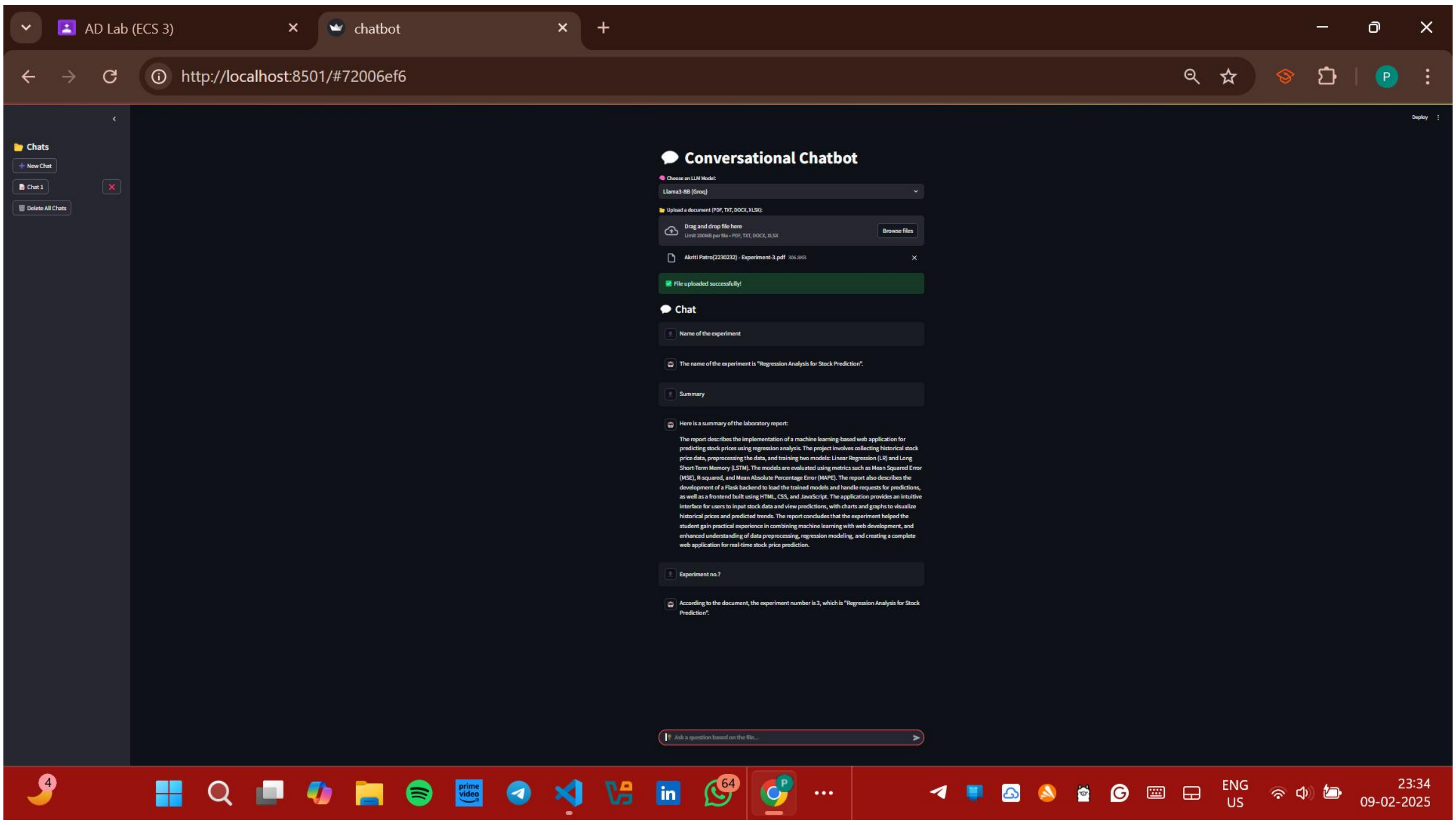
1. I integrated open-sources LLMs such as LLama and Gemini .
2. I developed a Streamlit backend to process the PDF/word/excel content.
3. I implemented Natural Language Processing (NLP) to allow queries. I used Langchain.
4. I created a frontend to upload document files and interact with the chatbot, just like OpenAI interface.
5. I provided an option to choose the LLM model from a dropdown list.
6. I displayed the chatbot responses on the webpage.

3. **Code:-**

```
import streamlit as st
import os
from langchain_groq import ChatGroq
import google.generativeai as genai
from PyPDF2 import PdfReader
from docx import Document
import pandas as pd
from dotenv import load_dotenv

# Load environment variables
load_dotenv()

# API Keys
groq_api_key = os.getenv("GROQ_API_KEY")
google_api_key = os.getenv("GOOGLE_API_KEY")

if not groq_api_key or not google_api_key:
    st.error("❌ ERROR: Missing API keys! Please check your .env file.")
    st.stop()

# LLM Model Options
model_options = {
    "Llama3-8B (Groq)": "llama3-8b-8192",
    "Mixtral (Groq)": "mixtral-8x7b-32768",
    "Gemini Pro (Google)": "gemini-pro"
}

# Sidebar for chat management
st.sidebar.title("    Chats")
if "chats" not in st.session_state:
    st.session_state.chats = {}
```

```
# Initialize selected model in session state if it doesn't exist
if "selected_model" not in st.session_state:
    st.session_state.selected_model = None

# Create new chat or reset if all chats are deleted
if st.sidebar.button("➕ New Chat"):
    # Default naming for chats should be Chat 1, Chat 2, Chat 3, etc.
    chat_name = f"Chat {len(st.session_state.chats) + 1}"
    st.session_state.chats[chat_name] = []
    st.session_state.current_chat = chat_name
    st.session_state.file_uploaded = None  # Reset file when new chat is created
    st.session_state.selected_model = None  # Reset model when new chat is created
    st.rerun()

# Delete a specific chat
for chat_name in list(st.session_state.chats.keys()):
    cols = st.sidebar.columns([0.8, 0.2])
    if cols[0].button(f"    {chat_name}", key=chat_name):
        st.session_state.current_chat = chat_name
        st.rerun()
    if cols[1].button("❌", key=f"delete_{chat_name}") :
        del st.session_state.chats[chat_name]
        st.session_state.current_chat = list(st.session_state.chats.keys())[0] if st.session_state.chats else None
        st.session_state.file_uploaded = None  # Reset file when chat is deleted
        st.session_state.selected_model = None  # Reset model when chat is deleted
        st.rerun()

# Option to delete all chats
if st.sidebar.button("    Delete All Chats"):
    st.session_state.chats = {}
    st.session_state.current_chat = None
    st.session_state.file_uploaded = None  # Reset file when all chats are deleted
    st.session_state.selected_model = None  # Reset model when all chats are deleted
    st.rerun()

# Ensure there's at least one active chat
if "current_chat" not in st.session_state or not st.session_state.current_chat:
    if st.session_state.chats:
        st.session_state.current_chat = list(st.session_state.chats.keys())[0]
    else:
        chat_name = f"Chat 1"
        st.session_state.chats[chat_name] = []
        st.session_state.current_chat = chat_name

# Main UI
st.title("    Conversational Chatbot")
selected_model = st.selectbox("    Choose an LLM Model:", list(model_options.keys()))

uploaded_file = st.file_uploader("    Upload a document (PDF, TXT, DOCX, XLSX):", type=["pdf", "txt",
"docx", "xlsx"])

# Store file content only for the current chat
if uploaded_file:
    st.session_state.file_uploaded = uploaded_file
    st.success("✅ File uploaded successfully!")
elif "file_uploaded" not in st.session_state or not st.session_state.file_uploaded:
    file_content = ""
else:
    uploaded_file = st.session_state.file_uploaded

# Function to extract text from files
def extract_text(file):
    if file.type == "application/pdf":
        pdf_reader = PdfReader(file)
        return "\n".join([page.extract_text() for page in pdf_reader.pages if page.extract_text()])
    elif file.type == "text/plain":
        return file.getvalue().decode("utf-8")
    elif file.type == "application/vnd.openxmlformats-officedocument.wordprocessingml.document":
        doc = Document(file)
        return "\n".join([para.text for para in doc.paragraphs])
    elif file.type == "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet":
        df = pd.read_excel(file, sheet_name=None)
        text = []
        for sheet, data in df.items():
            text.append(f"    Sheet: {sheet}\n")
            text.append(data.to_string(index=False))
        return "\n\n".join(text)
    return ""

file_content = extract_text(uploaded_file) if uploaded_file else ""

# Set up the LLM model
if selected_model != st.session_state.selected_model:
    st.session_state.selected_model = selected_model

if selected_model.startswith("Gemini"):
    genai.configure(api_key=google_api_key)
    model = genai.GenerativeModel("gemini-pro")
else:
    model = ChatGroq(groq_api_key=groq_api_key, model_name=model_options[selected_model])

st.subheader("    Chat")

# Display chat history
if st.session_state.current_chat in st.session_state.chats:
    for chat in st.session_state.chats[st.session_state.current_chat]:
        if chat["role"] == "user":
            with st.chat_message("user", avatar="    "):
                st.write(chat["text"])
        else:
            with st.chat_message("assistant", avatar="    "):
                st.write(chat["text"])

# User input
user_input = st.chat_input("    Ask a question based on the file...")

# Handle the user input and generate the response
if user_input and file_content:
    st.session_state.chats[st.session_state.current_chat].append({"role": "user", "text": user_input})

    prompt = f"Here's the document content:\n\n{file_content}\n\nBased on this, answer the question:
{user_input}"

    if selected_model.startswith("Gemini"):
        response = model.generate_content(prompt)
        answer = response.text
    else:
        response = model.invoke(prompt)
        answer = response.content if hasattr(response, "content") else "No answer found."

    st.session_state.chats[st.session_state.current_chat].append({"role": "bot", "text": answer})
    st.rerun()

elif user_input:
    st.warning("   Please upload a file before asking a question.")
```

4. Results/Output:-



Conversational Chatbot

Choose an LLM Model: Llama3-8B (Groq)

Upload a document (PDF, TXT, DOCX, XLSX):

Drag and drop file here
Limit 200MB per file • PDF, TXT, DOCX, XLSX
Browse files

Akriti Patro(2230232) - Experiment-3.pdf 306.8KB

File uploaded successfully!

Chat

Name of the experiment

The name of the experiment is "Regression Analysis for Stock Prediction".

Summary

Here is a summary of the laboratory report:

The report describes the implementation of a machine learning-based web application for predicting stock prices using regression analysis. The project involves collecting historical stock price data, preprocessing the data, and training two models: Linear Regression (LR) and Long Short-Term Memory (LSTM). The models are evaluated using metrics such as Mean Squared Error (MSE), R-squared, and Mean Absolute Percentage Error (MAPE). The report also describes the development of a Flask backend to load the trained models and handle requests for predictions, as well as a frontend built using HTML, CSS, and JavaScript. The application provides an intuitive interface for users to input stock data and view predictions, with charts and graphs to visualize historical prices and predicted trends. The report concludes that the experiment helped the student gain practical experience in combining machine learning with web development, and enhanced understanding of data preprocessing, regression modeling, and creating a complete web application for real-time stock price prediction.

Experiment no.?

According to the document, the experiment number is 3, which is "Regression Analysis for Stock Prediction".

Ask a question based on the file...

5. Remarks:-

In this experiment, I developed a chatbot that answers queries based on the content of uploaded PDF, DOCX, and XLSX files. I integrated open-source LLMs like LLama and Gemini, used LangChain for NLP, and extracted document text using libraries like PyPDF2, python-docx, and pandas. The frontend was built with Streamlit, allowing users to interact with the chatbot by uploading documents and selecting an LLM model.

Signature of the Student	Signature of the Lab Coordinator
Akriti Patro	Prof. Bhargav Appasani