# Revolutionizing PDF Reading and Customer Support with AI Technology

Submitted in fulfillment of the requirements of the degree of

Bachelor of Engineering

By

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | |  | | **ID** | |
| Jash Patel | |  | | VU4F2021010 | |
| Ayush Singh | |  | | VU4F2021011 | |
| Durvesh Teke  Vaibhav Ghutukade | |  | | VU4F2021021 VU4F2021026 | |
|  | | Supervisor: | |  | |

Prof. Gayatri Bachhav



Information Technology

Vasantdada Patil Pratishthan’s College of Engineering

2023-24

**A PROJECT REPORT ON**

# Revolutionizing PDF Reading and Customer Support with AI Technology

Submitted in fulfillment of the requirements of the degree of

Bachelor of Engineering

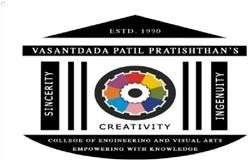
|  |  |  |
| --- | --- | --- |
|  | By |  |
| **Name** |  | **ID** |
| Jash Patel |  | VU4F2021010 |
| Ayush Singh |  | VU4F2021011 |
| Durvesh Teke  Vaibhav Ghutukade |  | VU4F2021021  VU4F2021026 |

Under the Guidance Of

**Prof. Gayatri Bachhav**

**Department of Information Technology**

**Vasantdada Patil Pratishthan's College of Engineering & Visual Arts**



Vasantdada Patil Educational Complex, Eastern Express Highway, Near Everard Nagar,Sion- Chunabhatti, Mumbai – 400022.

**UNIVERSITY OF MUMBAI**

**2023 to 2024**

## CERTIFICATE

This is to certify that the project entitled **“****Revolutionizing PDF Reading and Customer Support with AI Technology”** is a bonafide work of “**Jash Patel (VU4F2021010)”, “Ayush Singh (VU4F2021011), “Durvesh Teke (VU4F2021021)”, “Vaibhav Ghutukade (VU4F2021026)”** submitted to the University of Mumbai in fulfillment of the requirement for the award of the degree of **“Bachelor of Engineering”** in **“Information Technology”**.

Prof. Gayatri Bachhav (Project Guide)

Dr.Pradip Mane HOD(IT)

Dr.Alam N. Shaikh (Principal)

Project Report Approval for B. E.

This thesis / dissertation/project report entitled **Revolutionizing PDF Reading and Customer Support with AI Technology** by **Jash Patel, Ayush Singh, Durvesh Teke, Vaibhav Ghutukade** is approved for the degree of Bachelor of Engineering (Information Technology).

Examiners

1.

2.

Date:

Place: Mumbai

# Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Jash Patel (VU4F2021010)

Ayush Singh (VU4F2021011)

Durvesh Teke (VU4F2021021)

Vaibhav Ghutukade (VU4F2021026)

Date:

## Abstract

The rapid growth of digital documents has necessitated efficient tools for document management and reading. This paper introduces a novel PDF reader chatbot application designed to improve the user experience and interaction with PDF documents. The chatbot leverages natural language processing (NLP) and machine learning techniques to enable users to interact with PDF documents through conversational interfaces. Users can ask questions, request summaries, extract specific information, and navigate through PDFs using intuitive and natural language commands. The chatbot employs text extraction and comprehension mechanisms to provide accurate and relevant responses, enhancing document understanding and accessibility. The proposed PDF reader chatbot aims to streamline document retrieval, comprehension, and navigation, ultimately improving productivity and accessibility for users in various domains. The primary objective of this study is to develop an innovative PDF reader chatbot application that leverages natural language processing (NLP) to enhance user interaction with PDF documents. The goal is to enable more intuitive and conversational interaction, allowing users to efficiently navigate, extract relevant information, and obtain summaries from PDFs using natural language commands. By achieving this objective, we aim to significantly improve document accessibility, streamline document retrieval and comprehension, and ultimately enhance productivity for users across diverse domains

Submitted by:

Jash Patel (VU4F2021010)

Ayush Singh (VU4F2021011)

Durvesh Teke (VU4F2021021)

Vaibhav Ghutukade (VU4F2021026)

### Table of Contents

**Chapters**

|  |  |
| --- | --- |
| **1. Introduction** | **1** |
| 1.1 Aim of Project | 2 |
| 1.2 Objectives | 2 |
| 1.3 Scope of Project | 3 |
| 1.4 Organization of Report | 4 |
| **2. Review of Literature** | **5** |
| 2.1 Existing System | 5 |
| 2.2 Literature Survey | 6 |
| 2.3 Proposed System | 9 |
| **3. Requirement Analysis** | **10** |
| 3.1 Software Requirements | 10 |
| 3.2 System Requirements | 11 |
| **4. System Design** | **12** |
| 4.1 Working & Architecture | 12 |
| 4.1.1 Algorithm Used | 14 |
| 4.2 Flow Chart | 15 |
| **5. Implementation** | **16** |
| 5.1 Code | 16 |
| **6. Results** | **45** |
| **7. Conclusion** | **48** |
| 7.1 Conclusion | 48 |
| 7.2 Future Scope | 48 |
| **8. References** | **49** |
| **Acknowledgement** | **50** |
| **Publication and Certificates** | **51** |

**List of Figures**

|  |  |
| --- | --- |
| 4.1.1 System Architecture | 12 |
| 4.1.2 Working | 13 |
| 4.2.1 Flow Chart | 15 |
| 6.1 Result Screenshots | 45 |

## Chapter 1 Introduction

The introduction sets the stage by explaining the ubiquity of PDF files in today's digital world and their importance in sharing information. It also highlights the drawbacks of current PDF reader applications, such as limited interactivity and lack of intelligent features. The introduction introduces the proposed solution: a sophisticated PDF reader chatbot web application. This application aims to revolutionize the way users interact with PDFs by leveraging conversational AI to simplify information extraction, provide quick summaries, and assist in effective PDF navigation.The introduction introduces the proposed solution: a sophisticated PDF reader chatbot web application. This application aims to revolutionize the way users interact with PDFs by leveraging conversational AI to simplify information extraction, provide quick summaries, and assist in effective PDF navigation. The introduction seeks to illuminate the significance of Portable Document Format (PDF) in contemporary communication and information sharing. It elucidates how PDFs have evolved to become a standard for document exchange and archiving due to their platform-independent nature and consistent formatting. Despite their prevalence, traditional PDF readers have not kept pace with the evolving needs of users. The conventional approach to PDF reading lacks dynamism and falls short in providing an enriching user experience. The proposed solution, a sophisticated PDF reader chatbot web application, aims to amalgamate the efficiency of traditional PDF reading with the power of natural language processing (NLP) and conversational AI. This fusion intends to create a seamless interaction platform that simplifies the extraction of valuable information, offers summarization capabilities, and enhances navigation within PDF documents. By leveraging the potential of AI-driven chatbot technology, this application is envisioned to reshape how users engage with PDFs, making the process more intuitive, interactive, and productive.

### Aim of Project

With the exponential growth of textual information, PDF documents, in particular, have become static repositories that often overwhelm users with information overload. Traditional methods of document interaction, such as keyword searches, are inefficient and fail to provide an intuitive way to access and extract valuable knowledge. The aim of this project is to design, develop, and implement a PDF Reader AI Chatbot that serves as a versatile and intelligent customer support tool. This chatbot will be tailored to address the specific challenges associated with handling PDF documents in various customer support scenarios.

### Objectives

1. To Enhance PDF Document Accessibility:

Develop a chatbot capable of processing and extracting information from PDF documents, making their content more accessible to users.

2. To Provide Seamless Customer Support:

Create a user-friendly interface that allows customers to interact with the chatbot effortlessly, seeking information and assistance related to PDF documents.

3. To Improve Response Times:

Implement advanced natural language processing (NLP) algorithms to enable the chatbot to understand and respond to user queries with accuracy and speed, reducing response times in customer support interactions.

4. To Support Various Use Cases:

Design the chatbot to accommodate a wide range of customer support scenarios, including technical troubleshooting, product inquiries, and general information retrieval, all within the context of PDF documents.

5. To Enhance User Experience:

Focus on user-centric design to ensure that users, even those without technical expertise, can benefit from the chatbot's capabilities, improving overall customer satisfaction.

### Scope of Project

The scope of this project encompasses the development of a PDF Reader AI Chatbot tailored for customer support, with a primary focus on enhancing the user experience when dealing with PDF documents. The chatbot's capabilities will include efficient PDF file parsing, text extraction, search functionalities, and context-aware responses to user queries related to the content within PDF files .The project enables users to upload PDF documents, which are then processed and split into pages for further analysis. Each page's content is embedded into a vector space using Hugging Face embeddings and stored using Chroma for efficient retrieval. Users can then engage with a chat interface to ask questions about the content of the uploaded PDFs. The chatbot leverages a combination of language model chains, including Ollama and RetrievalQA, to provide answers based on contextual prompts derived from the document content. Overall, the project aims to create an interactive platform for extracting and querying information from PDF documents through natural language interactions.

### Organization of Report

The rest of the report is organized as follows:

* + - Chapter 2 describes the literature survey.
    - Chapter 3 describes our requirement Analysis.
    - Chapter 4 demonstrate the system design with the help of architecture and diagrams.
    - Chapter 5 depicts the implementation part of the system.
    - Chapter 6 discusses the results.
    - Chapter 7 concludes the report with conclusion and future Scope with the published paper.

## Chapter 2 Review of Literature

### Existing System

By understanding the limitations of the current systems and the evolving user expectations, your project seeks to bridge the gap between PDF readers and customer support chatbots, offering a more advanced and integrated solution. This will improve user experiences and enhance the efficiency of handling customer inquiries related to PDF content.

* PDF Readers:
* Conventional PDF readers primarily focus on rendering and displaying PDF documents.
* They provide basic functionalities such as zooming, searching, and text selection.
* Many existing PDF readers lack advanced features, like text extraction, natural language processing (NLP), and the ability to provide context-aware information.
* Customer Support Chatbots:
* These chatbots are text-based or voice-based, and they can answer frequently asked questions and provide basic assistance.
* Many customer support chatbots do not seamlessly integrate with PDF readers, making it challenging to provide immediate and accurate responses to PDF-related queries.
* Challenges in the Existing System:
* Lack of context: Conventional PDF readers and chatbots do not have a deep understanding of the content within PDF documents. They do not consider the context in which the user is reading the document.
* Manual retrieval: Users typically need to manually search for information in PDF documents, which can be time-consuming and frustrating.
* Limited interactivity: Existing customer support chatbots may not be able to interpret and respond to PDF-specific questions effectively.
* Scalability and adaptability issues: Many chatbots are rigid and require substantial manual maintenance to adapt to changing support needs.

### Literature Survey

Each study provides valuable insights into the methodology, results, and scope of the respective research. This literature survey provides a comprehensive overview of relevant studies, their methodologies, results, and scope. It highlights the diverse areas of research related to PDF Reader AI Chatbots for customer support, ranging from general customer support applications to industry-specific implementations. The findings from these studies collectively contribute to the understanding and potential applications of PDF Reader AI Chatbots in enhancing customer support services.

[1] A new method of information extraction from PDF files by Fang Yuan, & Bo Lu. (2005)

|  |  |  |
| --- | --- | --- |
| **METHODOLOGY** | **RESULT** | **SCOPE** |
| The methodology presented in this paper involves several steps. First, the PDF files are parsed to extract text information. Then, tags are injected into the text information to identify the structure of the information. | The experiment showed that with the increase of the amount of training data, the accuracy of information extraction increased correspondingly. The result of the experiment proved that the new method of information extraction from PDF files is effective and makes a foundation for managing and searching a large amount of PDF files. | The scope of this paper is to present a new method of information extraction from PDF files that can help manage and search a large amount of PDF files efficiently and quickly. The paper focuses on the methodology used to extract information from PDF files, which involves parsing PDF files, injecting tags into text information, and applying a pattern match algorithm based on tree model to obtain the solution. |

[2] Chatbot: An automated conversation system on Artificial Intelligence and Natural Language Processing by Mondal, A., Dey, M., Das, D., Nagpal, S., & Garda, K .

|  |  |  |
| --- | --- | --- |
| **METHODOLOGY** | **RESULT** | **SCOPE** |
| The methodology section of the PDF file discusses the approach used to design the chatbot system. It includes the data preparation, feature extraction, and model building in detail . | The PDF file presents the results of the chatbot system in terms of precision, recall, Fmeasure, and accuracy score for various combinations of the dataset . The paper also provides an example of a question asked by a visitor and the corresponding response generated by the chatbot . | The concluding remarks and future scope of the research are discussed in Section V of the PDF file . The paper suggests that the proposed chatbot system can be further improved by incorporating more advanced natural language processing techniques and by expanding the dataset to cover a wider range of topics. |

[3] Leveraging GPT-4 for PDF Data Extraction: A Comprehensive Guide by Manish Sharma.

|  |  |  |
| --- | --- | --- |
| **METHODOLOGY** | **RESULT** | **SCOPE** |
| Machine learning techniques involve training models to recognize and extract information from PDF files, regardless of the file structure  . GPT-4 is a powerful language model that can be used to perform question answering tasks for PDF extraction. | The limitations of current PDF data extraction methods include the inability to extract information from unstructured documents and the potential for inaccurate extractions using machine learning techniques. | The PDF covers the current methods of PDF data extraction, their limitations, and how GPT-4 can be used to perform question-answering tasks for PDF extraction. The PDF also provides a step-bystep guide for implementing  GPT-4 for PDF data extraction. |

[4] Florez-Choque, O., & Cuadros-Vargas, E. (2007). Improving Human Computer Interaction through Spoken Natural Language. 2007 IEEE

|  |  |  |
| --- | --- | --- |
| **METHODOLOGY** | **RESULT** | **SCOPE** |
| The methodology used in the PDF file involves the development of a model that uses spoken natural language queries and voice communication to recover information from databases. | Overall, the study concludes that spoken natural language has the potential to significantly improve humancomputer interaction, and the use of voice and natural language allows for a better experience with the computer. | The scope of the PDF file is to explore the potential of spoken natural language to improve human-computer interaction and to present a model that can recognize phonemes in spoken words and support the syntactic and semantic structure of the Spanish language. |

[5] Dr. M .John Basha , Dr. S. Vijayakumar , J. Jayashankari ,Ahmed Hussein ,Alawadi Durdona (2019). "Advancements in Natural Language Processing for Document Analysis." Proceedings of the Annual Conference on Neural Information Processing Systems.

|  |  |  |
| --- | --- | --- |
| **METHODOLOGY** | **RESULT** | **SCOPE** |
| Methods and techniques related to Natural Language Processing, including deep learning, information retrieval, knowledge graphs, text summarization, machine reading comprehension, aspect-based sentiment analysis, and cross-lingual natural language processing. | Overall, the result of this PDF file is to provide a comprehensive overview of the latest developments, challenges, and strategies in the field of Natural Language  Processing for Text  Understanding. | Additionally, the paper also covers the challenges and ongoing research in the field of NLP, including coreference and anaphora resolution tasks, bias and fairness problems in NLP models and datasets, and cross-lingual natural language processing. |

### Proposed System

The proposed system entails the development of an advanced PDF chatbot leveraging cutting-edge technologies. It involves creating a web application using FastAPI, allowing users to upload PDF documents for analysis. Upon upload, the system utilizes text splitting techniques to segment the document into manageable chunks. Each chunk undergoes embedding into a vector space via Hugging Face embeddings and is indexed for efficient retrieval using Chroma. The chatbot interface enables users to interact with the system by posing questions about the uploaded PDFs. Through a combination of language model chains, including Ollama and RetrievalQA, the chatbot generates responses based on the context extracted from the documents, providing accurate and relevant information to the user's queries. This system aims to offer an intuitive and effective platform for extracting insights from PDF documents through natural language interactions, catering to diverse user needs across various domains.

Conversational Interface: The chatbot will provide a user-friendly conversational interface where users can interact with their PDF documents using natural language.

Document Retrieval: Users can upload or link their PDF documents to the chatbot, and the chatbot will store and organize them for easy access.

Natural Language Commands: Users can ask questions, request summaries, extract specific information, and navigate through PDFs using intuitive and natural language commands. For example, they can ask, "Summarize the second paragraph of the document" or "Find all instances of 'key terms' in the PDF."

* Advantages of the proposed system:
* Enhanced user experience with natural language interaction.
* 24/7 availability for customer support.
* Efficient handling of complex PDF documents.
* Reduced human resource costs.

## Chapter 3 Requirement Analysis

### Software Requirements:

|  |  |  |
| --- | --- | --- |
| **SR No.** | **Software** | **Purpose** |
| 1 | Python 3.x | Core programming language for development |
| 2 | FastAPI | Web framework for building APIs |
| 3 | PyTorch | Deep learning framework for language models |
| 4 | Hugging Face Transformers | Library for natural language processing models |
| 5 | PyPDF2 | Library for extracting text from PDF documents |
| 6 | ChromaDB | Library for indexing and querying document embeddings |
| 7 | Gunicorn | WSGI HTTP server for deploying FastAPI applications |
| 8 | Jinja2 | Template engine for rendering HTML templates |

### System Requirements:

The minimum PC requirements for running the PDF chatbot project:

* Processor: Intel Core i3 or equivalent AMD processor
* RAM: 4 GB
* Storage: At least 10 GB of free disk space
* Operating System: Windows 10, macOS, or Linux
* Python Version: Python 3.6 or higher
* Internet Connection: Required for downloading dependencies and model files
* Graphics: Integrated graphics card is sufficient for basic usage
* Display: Monitor with a resolution of 1280x720 or higher
* Web Browser: Any modern web browser such as Chrome, Firefox, or Safari for accessing the chatbot interface

These specifications should be adequate for running the PDF chatbot project smoothly on a standard desktop or laptop computer.

## Chapter 4 System Design

### Working & Architecture

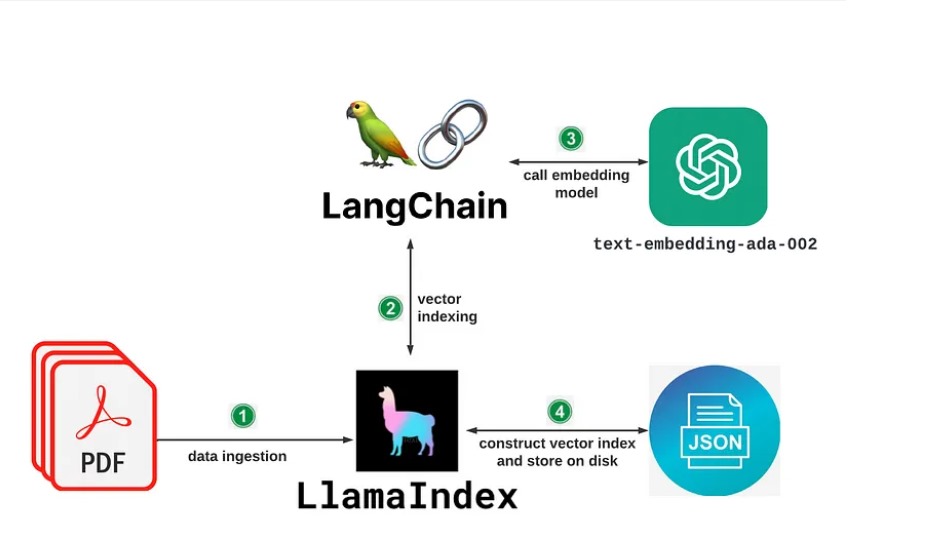


Fig.4.1.1: System Architecture

* PDF data ingestion: In this step, the system ingests data from a PDF file.
* Text-embedding model: The text extracted from the PDF is then processed by a text-embedding model, which generates a numerical representation of the text data. This step allows the system to understand the semantic meaning of the text.
* Call embedding model: This step calls another embedding model, possibly to further enrich the data or create a specific kind of embedding.
* Vector indexing: The generated vector representations are then indexed using LlamaIndex, which might be a specific software library for indexing and searching vector data.
* Construct vector index and store on disk: Finally, the indexed data is stored on disk for later use.

Overall, this system architecture seems to describe a process for converting textual data from a PDF file into a searchable format using text embedding and indexing techniques.

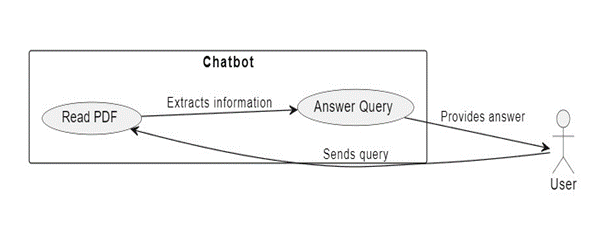


Fig.4.1.2: Working

* Chatbot: The process starts with the chatbot, which reads information from a text file.
* Extracts information: The chatbot then extracts information from the text file. What kind of information is extracted depends on the purpose of the chatbot and the content of the text file.
* Formulates a query: Based on the extracted information, the chatbot formulates a query.
* Sends query to user: The formulated query is then sent to the user.
* User: The user receives the query from the chatbot.
* Provides answer: The user reads the query and provides an answer to the chatbot.
* Chatbot: Finally, the chatbot receives the user's answer.
  + 1. **Algorithm Used:**
* Text Splitting Algorithm: The project utilizes a text splitting algorithm, specifically the Recursive Character Text Splitter, to segment PDF documents into smaller, manageable chunks. This algorithm helps in breaking down large documents into more digestible parts for analysis.
* Embedding Algorithm: The project employs embedding algorithms to convert text data into numerical representations. Specifically, it utilizes Hugging Face Embeddings, which likely incorporates transformer-based architectures such as BERT or RoBERTa, to generate dense vector representations of text sequences.
* Vector Indexing Algorithm: The project uses an indexing algorithm, likely implemented in the Chroma library, to efficiently store and retrieve vector representations of document content. This indexing algorithm enables fast and scalable similarity search over the document embeddings.
* Retrieval-Based Question Answering Algorithm: The project employs a retrieval-based question-answering algorithm, implemented in the RetrievalQA module, to generate responses to user queries based on the indexed document content. This algorithm likely utilizes techniques such as nearest neighbor search to find relevant document chunks and language model chains to generate responses.
* Language Model Algorithms: The project leverages language model algorithms, such as those provided by the Ollama and LLMChain modules, for contextual understanding and response generation. These algorithms may be based on transformer architectures and fine-tuned on large text corpora to understand and generate human-like responses

### Flow Chart

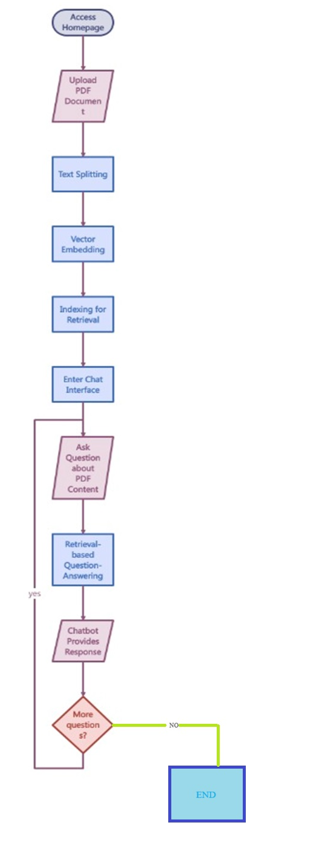


Fig.4.2.1: Flow Chart

## Chapter 5

## Implementation

## 5.1 Code

## App.py

# \_\_import\_\_('pysqlite3')

# import sys

# sys.modules['sqlite3'] = sys.modules.pop('pysqlite3')

import os

import shutil

from utils import \*

from uuid import uuid4

from fastapi.staticfiles import StaticFiles

from fastapi.templating import Jinja2Templates

from langchain.prompts import PromptTemplate

from langchain.chains.llm import LLMChain

from fastapi.responses import JSONResponse, RedirectResponse

from fastapi import FastAPI, Response, Request, File, UploadFile

import chromadb

from langchain.chains import RetrievalQA

from langchain\_community.llms import Ollama

from langchain\_community.vectorstores import Chroma

from langchain\_community.document\_loaders import PyPDFLoader

from langchain\_community.embeddings import HuggingFaceEmbeddings

from langchain.text\_splitter import RecursiveCharacterTextSplitter

from langchain.chains.combine\_documents.stuff import StuffDocumentsChain

app = FastAPI()

templates = Jinja2Templates(directory="templates")

app.mount("/static", StaticFiles(directory="static"), name="static")

directory = 'index\_store'

text\_splitter = RecursiveCharacterTextSplitter(chunk\_size=2000, chunk\_overlap=200)

llm = Ollama(model="llama2", base\_url="http://127.0.0.1:11434", verbose=True, temperature=0.6, )

embeddings = HuggingFaceEmbeddings(model\_name="sentence-transformers/all-MiniLM-L6-v2", model\_kwargs={'device': 'cpu'})

client = chromadb.PersistentClient(path=directory)

prompt\_template = """Use the following pieces of context to answer the question at the end. Please follow the following rules:

1. If the question is to request links, please only return the source links with no answer.

2. If you don't know the answer, don't try to make up an answer. Just say \*\*I can't find the final answer but you may want to check the following links\*\* and add the source links as a list.

3. If you find the answer, write the answer in a concise way and add the list of sources that are \*\*directly\*\* used to derive the answer. Exclude the sources that are irrelevant to the final answer.

{context}

Question: {question}

Helpful Answer:"""

QA\_CHAIN\_PROMPT = PromptTemplate.from\_template(prompt\_template)

llm\_chain = LLMChain(llm=llm, prompt=QA\_CHAIN\_PROMPT, callbacks=None, verbose=True)

document\_prompt = PromptTemplate(

input\_variables=["page\_content", "source"],

template="Context:\ncontent:{page\_content}\nsource:{source}",

)

combine\_documents\_chain = StuffDocumentsChain(

llm\_chain=llm\_chain,

document\_variable\_name="context",

document\_prompt=document\_prompt,

callbacks=None,

)

@app.get("/")

async def index(request: Request, response: Response):

response = templates.TemplateResponse(request=request, name='index.html', response=response)

response.set\_cookie("cookie", uuid4())

return response

@app.post("/upload/")

async def upload(request: Request, response: Response, file: UploadFile = File(...)):

if file.content\_type != "application/pdf":

return JSONResponse({"error": "File not allowded!"})

if 'cookie' not in request.cookies.keys():

return JSONResponse({"error": "Cookie not found!"})

session\_cookie = request.cookies.get('cookie')

file\_location = f"./media/{file.filename}"

with open(file\_location, "wb+") as file\_object:

shutil.copyfileobj(file.file, file\_object)

loader = PyPDFLoader(file\_location)

pages = loader.load\_and\_split(text\_splitter)

vector\_index = Chroma.from\_documents(documents=pages, embedding=embeddings, persist\_directory=directory, collection\_name=create\_hash(session\_cookie))

vector\_index.persist()

redirect\_url = request.url\_for('chat')

return JSONResponse({"filename": file.filename, "redirect\_url": redirect\_url.\_\_str\_\_()})

@app.get('/chat')

async def chat(request: Request, response: Response):

return templates.TemplateResponse(request=request, name='chat.html', response=response)

@app.post('/message')

async def message(request: Request, response: Response):

if 'cookie' not in request.cookies.keys():

return JSONResponse({"error": "Cookie not found!"})

session\_cookie = request.cookies.get('cookie')

vectordb = Chroma(persist\_directory=directory, embedding\_function=embeddings, collection\_name=create\_hash(session\_cookie))

retriever = vectordb.as\_retriever(search\_type="similarity", search\_kwargs={"k":6})

data = await request.form()

message = data.get('message')

qa = RetrievalQA(

combine\_documents\_chain=combine\_documents\_chain,

callbacks=None,

verbose=False,

retriever=retriever,

return\_source\_documents=False,

)

response = qa.invoke(message)

return JSONResponse(response)

**CHAT.HTML**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Chat with PDF</title>

<link href="static/main.css" rel="stylesheet">

</link>

<link href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.1.3/css/bootstrap.min.css" rel="stylesheet">

</link>

</head>

<body class="hero-anime">

<div class="navigation-wrap bg-light start-header start-style">

<div class="container">

<div class="row">

<div class="col-12">

<nav class="navbar navbar-expand-md navbar-light">

<div class="website-name">Chat With PDF</div>

<button class="navbar-toggler" type="button" data-toggle="collapse"

data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav ml-auto py-4 py-md-0">

<li class="nav-item pl-4 pl-md-0 ml-0 ml-md-4">

<a class="nav-link" href="/" id="home-button">Home</a>

</li>

</ul>

</div>

</nav>

</div>

</div>

</div>

</div>

<div class="section full-height">

<div class="absolute-center">

<div class="section">

<div class="container">

<div class="row">

<div class="col-12">

<section class="msger">

<main class="msger-chat" id="message-body">

<div class="msg left-msg">

<div class="msg-bubble">

<div class="msg-info">

<div class="msg-info-name">BOT</div>

</div>

<div class="msg-text">

Hi, welcome to Chat with PDF! How may I help you?

</div>

</div>

</div>

</main>

<div class="d-flex justify-content-center" >

<div class="alert alert-warning col-2" id="copyClipBoard">

<div class="spinner-border" role="status">

<span class="visually-hidden">Thinking...</span>

</div>

</div>

</div>

<form class="msger-inputarea">

<input type="text" class="msger-input" placeholder="Enter your message..." id="text-input">

<button type="submit" class="msger-send-btn" id="send-message">Send</button>

</form>

</section>

</div>

</div>

</div>

</div>

</div>

</div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-cookie/1.4.1/jquery.cookie.min.js"></script>

<script>

$(document).ready(function () {

var cookieValue = $.cookie('cookie');

var loadingBlock = $("#copyClipBoard");

loadingBlock.hide();

if (cookieValue == undefined){

window.location.href = '/';

}

let messageBody = $("#message-body");

const userMessage = (message) => {

return `

<div class="msg right-msg">

<div class="msg-bubble">

<div class="msg-info">

<div class="msg-info-name">User</div>

</div>

<div class="msg-text">

${message}

</div>

</div>

</div>

`

}

const responseMessage = (message) => {

return `

<div class="msg left-msg">

<div class="msg-bubble">

<div class="msg-info">

<div class="msg-info-name">BOT</div>

</div>

<div class="msg-text">

${message}

</div>

</div>

</div>

`

}

$("#send-message").click(function (e) {

e.preventDefault();

if (cookieValue == undefined){

window.location.href = '/';

}

var formData = new FormData();

formData.append('message', $('#text-input').val());

messageBody.append(userMessage($('#text-input').val()));

$('#text-input').val('');

loadingBlock.show();

$.ajax({

url: '/message',

type: 'POST',

data: formData,

contentType: false,

processData: false,

success: function (response) {

if (response.error) {

alert(response.error)

window.location.href = '/';

} else {

messageBody.append(responseMessage(response.result));

loadingBlock.hide();

}

},

error: function (xhr, status, error) {

$("#status").text("Error uploading file: " + error);

}

});

});

});

</script>

<script>

(function ($) {

"use strict";

$(function () {

var header = $(".start-style");

$(window).scroll(function () {

var scroll = $(window).scrollTop();

if (scroll >= 10) {

header.removeClass('start-style').addClass("scroll-on");

} else {

header.removeClass("scroll-on").addClass('start-style');

}

});

});

//Animation

$(document).ready(function () {

$('body.hero-anime').removeClass('hero-anime');

});

//Menu On Hover

$('body').on('mouseenter mouseleave', '.nav-item', function (e) {

if ($(window).width() > 750) {

var \_d = $(e.target).closest('.nav-item'); \_d.addClass('show');

setTimeout(function () {

\_d[\_d.is(':hover') ? 'addClass' : 'removeClass']('show');

}, 1);

}

});

//Switch light/dark

$("#switch").on('click', function () {

if ($("body").hasClass("dark")) {

$("body").removeClass("dark");

$("#switch").removeClass("switched");

}

else {

$("body").addClass("dark");

$("#switch").addClass("switched");

}

});

})(jQuery);

</script>

</body>

</html>

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Chat with PDF</title>

<link href="static/main.css" rel="stylesheet">

</link>

<link href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.1.3/css/bootstrap.min.css" rel="stylesheet">

</link>

</head>

<body class="hero-anime">

<div class="navigation-wrap bg-light start-header start-style">

<div class="container">

<div class="row">

<div class="col-12">

<nav class="navbar navbar-expand-md navbar-light">

<div class="website-name">Chat With PDF</div>

<button class="navbar-toggler" type="button" data-toggle="collapse"

data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav ml-auto py-4 py-md-0">

</ul>

</div>

</nav>

</div>

</div>

</div>

</div>

<div class="section full-height">

<div class="absolute-center">

<div class="section">

<div class="container">

<div class="row">

<div class="col-12">

<input type="file" id="fileInput" accept="application/pdf">

<button id="uploadButton" class="btn btn-success">Upload</button>

<div id="status"></div>

<div id="next-button">

<a href="/chat">Chat</a>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-cookie/1.4.1/jquery.cookie.min.js"></script>

<script>

$("#next-button").hide();

$(document).ready(function(){

$("#uploadButton").click(function(e){

e.preventDefault()

var fileInput = document.getElementById('fileInput');

var file = fileInput.files[0];

var formData = new FormData();

formData.append('file', file);

$.ajax({

url: '/upload/',

type: 'POST',

data: formData,

contentType: false,

processData: false,

success: function(response){

if(response.error){

alert(response.error)

} else {

$("#status").text("File uploaded successfully. Filename: " + response.filename);

$("#next-button").show();

window.location.href = response.redirect\_url

}

},

error: function(xhr, status, error){

$("#status").text("Error uploading file: " + error);

}

});

});

});

</script>

</body>

</html>

**Main.css**

@import url('https://fonts.googleapis.com/css?family=Poppins:100,100i,200,200i,300,300i,400,400i,500,500i,600,600i,700,700i,800,800i,900,900i&subset=devanagari,latin-ext');

/\* #Primary

================================================== \*/

body {

font-family: 'Poppins', sans-serif;

font-size: 16px;

line-height: 24px;

font-weight: 400;

color: #212112;

background-image: url('https://s3-us-west-2.amazonaws.com/s.cdpn.io/1462889/pat-back.svg');

background-position: center;

background-repeat: repeat;

background-size: 7%;

background-color: #fff;

overflow-x: hidden;

transition: all 200ms linear;

}

::selection {

color: #fff;

background-color: #8167a9;

}

::-moz-selection {

color: #fff;

background-color: #8167a9;

}

/\* #Navigation

================================================== \*/

.start-header {

opacity: 1;

transform: translateY(0);

padding: 20px 0;

box-shadow: 0 10px 30px 0 rgba(138, 155, 165, 0.15);

-webkit-transition: all 0.3s ease-out;

transition: all 0.3s ease-out;

}

.start-header.scroll-on {

box-shadow: 0 5px 10px 0 rgba(138, 155, 165, 0.15);

padding: 10px 0;

-webkit-transition: all 0.3s ease-out;

transition: all 0.3s ease-out;

}

.start-header.scroll-on .navbar-brand img {

height: 24px;

-webkit-transition: all 0.3s ease-out;

transition: all 0.3s ease-out;

}

.navigation-wrap {

position: fixed;

width: 100%;

top: 0;

left: 0;

z-index: 1000;

-webkit-transition: all 0.3s ease-out;

transition: all 0.3s ease-out;

}

.navbar {

padding: 0;

}

.navbar-brand img {

height: 28px;

width: auto;

display: block;

filter: brightness(10%);

-webkit-transition: all 0.3s ease-out;

transition: all 0.3s ease-out;

}

.navbar-toggler {

float: right;

border: none;

padding-right: 0;

}

.navbar-toggler:active,

.navbar-toggler:focus {

outline: none;

}

.navbar-light .navbar-toggler-icon {

width: 24px;

height: 17px;

background-image: none;

position: relative;

border-bottom: 1px solid #000;

transition: all 300ms linear;

}

.navbar-light .navbar-toggler-icon:after,

.navbar-light .navbar-toggler-icon:before {

width: 24px;

position: absolute;

height: 1px;

background-color: #000;

top: 0;

left: 0;

content: '';

z-index: 2;

transition: all 300ms linear;

}

.navbar-light .navbar-toggler-icon:after {

top: 8px;

}

.navbar-toggler[aria-expanded="true"] .navbar-toggler-icon:after {

transform: rotate(45deg);

}

.navbar-toggler[aria-expanded="true"] .navbar-toggler-icon:before {

transform: translateY(8px) rotate(-45deg);

}

.navbar-toggler[aria-expanded="true"] .navbar-toggler-icon {

border-color: transparent;

}

.nav-link {

color: #212121 !important;

font-weight: 500;

transition: all 200ms linear;

}

.nav-item:hover .nav-link {

color: #8167a9 !important;

}

.nav-item.active .nav-link {

color: #777 !important;

}

.nav-link {

position: relative;

padding: 5px 0 !important;

display: inline-block;

}

.nav-item:after {

position: absolute;

bottom: -5px;

left: 0;

width: 100%;

height: 2px;

content: '';

background-color: #8167a9;

opacity: 0;

transition: all 200ms linear;

}

.nav-item:hover:after {

bottom: 0;

opacity: 1;

}

.nav-item.active:hover:after {

opacity: 0;

}

.nav-item {

position: relative;

transition: all 200ms linear;

}

/\* #Primary style

================================================== \*/

.bg-light {

background-color: #fff !important;

transition: all 200ms linear;

}

.section {

position: relative;

width: 100%;

display: block;

}

.full-height {

height: 100vh;

}

.over-hide {

overflow: hidden;

}

.absolute-center {

position: absolute;

top: 50%;

left: 0;

width: 100%;

margin-top: 40px;

transform: translateY(-50%);

z-index: 20;

}

h1 {

font-size: 48px;

line-height: 1.2;

font-weight: 700;

color: #212112;

text-align: center;

}

p {

text-align: center;

margin: 0;

padding-top: 10px;

opacity: 1;

transform: translate(0);

transition: all 300ms linear;

transition-delay: 1700ms;

}

body.hero-anime p {

opacity: 0;

transform: translateY(40px);

transition-delay: 1700ms;

}

h1 span {

display: inline-block;

transition: all 300ms linear;

opacity: 1;

transform: translate(0);

}

body.hero-anime h1 span:nth-child(1) {

opacity: 0;

transform: translateY(-20px);

}

body.hero-anime h1 span:nth-child(2) {

opacity: 0;

transform: translateY(-30px);

}

body.hero-anime h1 span:nth-child(3) {

opacity: 0;

transform: translateY(-50px);

}

body.hero-anime h1 span:nth-child(4) {

opacity: 0;

transform: translateY(-10px);

}

body.hero-anime h1 span:nth-child(5) {

opacity: 0;

transform: translateY(-50px);

}

body.hero-anime h1 span:nth-child(6) {

opacity: 0;

transform: translateY(-20px);

}

body.hero-anime h1 span:nth-child(7) {

opacity: 0;

transform: translateY(-40px);

}

body.hero-anime h1 span:nth-child(8) {

opacity: 0;

transform: translateY(-10px);

}

body.hero-anime h1 span:nth-child(9) {

opacity: 0;

transform: translateY(-30px);

}

body.hero-anime h1 span:nth-child(10) {

opacity: 0;

transform: translateY(-20px);

}

h1 span:nth-child(1) {

transition-delay: 1000ms;

}

h1 span:nth-child(2) {

transition-delay: 700ms;

}

h1 span:nth-child(3) {

transition-delay: 900ms;

}

h1 span:nth-child(4) {

transition-delay: 800ms;

}

h1 span:nth-child(5) {

transition-delay: 1000ms;

}

h1 span:nth-child(6) {

transition-delay: 700ms;

}

h1 span:nth-child(7) {

transition-delay: 900ms;

}

h1 span:nth-child(8) {

transition-delay: 800ms;

}

h1 span:nth-child(9) {

transition-delay: 600ms;

}

h1 span:nth-child(10) {

transition-delay: 700ms;

}

body.hero-anime h1 span:nth-child(11) {

opacity: 0;

transform: translateY(30px);

}

body.hero-anime h1 span:nth-child(12) {

opacity: 0;

transform: translateY(50px);

}

body.hero-anime h1 span:nth-child(13) {

opacity: 0;

transform: translateY(20px);

}

body.hero-anime h1 span:nth-child(14) {

opacity: 0;

transform: translateY(30px);

}

body.hero-anime h1 span:nth-child(15) {

opacity: 0;

transform: translateY(50px);

}

h1 span:nth-child(11) {

transition-delay: 1300ms;

}

h1 span:nth-child(12) {

transition-delay: 1500ms;

}

h1 span:nth-child(13) {

transition-delay: 1400ms;

}

h1 span:nth-child(14) {

transition-delay: 1200ms;

}

h1 span:nth-child(15) {

transition-delay: 1450ms;

}

#switch,

#circle {

cursor: pointer;

-webkit-transition: all 300ms linear;

transition: all 300ms linear;

}

#switch {

width: 60px;

height: 8px;

border: 2px solid #8167a9;

border-radius: 27px;

background: #000;

position: relative;

display: block;

margin: 0 auto;

text-align: center;

opacity: 1;

transform: translate(0);

transition: all 300ms linear;

transition-delay: 1900ms;

}

body.hero-anime #switch {

opacity: 0;

transform: translateY(40px);

transition-delay: 1900ms;

}

#circle {

position: absolute;

top: -11px;

left: -13px;

width: 26px;

height: 26px;

border-radius: 50%;

background: #000;

}

.switched {

border-color: #000 !important;

background: #8167a9 !important;

}

.switched #circle {

left: 43px;

box-shadow: 0 4px 4px rgba(26, 53, 71, 0.25), 0 0 0 1px rgba(26, 53, 71, 0.07);

background: #fff;

}

.nav-item .dropdown-menu {

transform: translate3d(0, 10px, 0);

visibility: hidden;

opacity: 0;

max-height: 0;

display: block;

padding: 0;

margin: 0;

transition: all 200ms linear;

}

.nav-item.show .dropdown-menu {

opacity: 1;

visibility: visible;

max-height: 999px;

transform: translate3d(0, 0px, 0);

}

.dropdown-menu {

padding: 10px !important;

margin: 0;

font-size: 13px;

letter-spacing: 1px;

color: #212121;

background-color: #fcfaff;

border: none;

border-radius: 3px;

box-shadow: 0 5px 10px 0 rgba(138, 155, 165, 0.15);

transition: all 200ms linear;

}

.dropdown-toggle::after {

display: none;

}

.dropdown-item {

padding: 3px 15px;

color: #212121;

border-radius: 2px;

transition: all 200ms linear;

}

.dropdown-item:hover,

.dropdown-item:focus {

color: #fff;

background-color: rgba(129, 103, 169, .6);

}

body.dark {

color: #fff;

background-color: #1f2029;

}

body.dark .navbar-brand img {

filter: brightness(100%);

}

body.dark h1 {

color: #fff;

}

body.dark h1 span {

transition-delay: 0ms !important;

}

body.dark p {

color: #fff;

transition-delay: 0ms !important;

}

body.dark .bg-light {

background-color: #14151a !important;

}

body.dark .start-header {

box-shadow: 0 10px 30px 0 rgba(0, 0, 0, 0.15);

}

body.dark .start-header.scroll-on {

box-shadow: 0 5px 10px 0 rgba(0, 0, 0, 0.15);

}

body.dark .nav-link {

color: #fff !important;

}

body.dark .nav-item.active .nav-link {

color: #999 !important;

}

body.dark .dropdown-menu {

color: #fff;

background-color: #1f2029;

box-shadow: 0 5px 10px 0 rgba(0, 0, 0, 0.25);

}

body.dark .dropdown-item {

color: #fff;

}

body.dark .navbar-light .navbar-toggler-icon {

border-bottom: 1px solid #fff;

}

body.dark .navbar-light .navbar-toggler-icon:after,

body.dark .navbar-light .navbar-toggler-icon:before {

background-color: #fff;

}

body.dark .navbar-toggler[aria-expanded="true"] .navbar-toggler-icon {

border-color: transparent;

}

/\* #Media

================================================== \*/

@media (max-width: 767px) {

h1 {

font-size: 38px;

}

.nav-item:after {

display: none;

}

.nav-item::before {

position: absolute;

display: block;

top: 15px;

left: 0;

width: 11px;

height: 1px;

content: "";

border: none;

background-color: #000;

vertical-align: 0;

}

.dropdown-toggle::after {

position: absolute;

display: block;

top: 10px;

left: -23px;

width: 1px;

height: 11px;

content: "";

border: none;

background-color: #000;

vertical-align: 0;

transition: all 200ms linear;

}

.dropdown-toggle[aria-expanded="true"]::after {

transform: rotate(90deg);

opacity: 0;

}

.dropdown-menu {

padding: 0 !important;

background-color: transparent;

box-shadow: none;

transition: all 200ms linear;

}

.dropdown-toggle[aria-expanded="true"]+.dropdown-menu {

margin-top: 10px !important;

margin-bottom: 20px !important;

}

body.dark .nav-item::before {

background-color: #fff;

}

body.dark .dropdown-toggle::after {

background-color: #fff;

}

body.dark .dropdown-menu {

background-color: transparent;

box-shadow: none;

}

}

/\* #Link to page

================================================== \*/

.logo {

position: absolute;

bottom: 30px;

right: 30px;

display: block;

z-index: 100;

transition: all 250ms linear;

}

.logo img {

height: 26px;

width: auto;

display: block;

filter: brightness(10%);

transition: all 250ms linear;

}

body.dark .logo img {

filter: brightness(100%);

}

.website-name {

font-family: 'Anton', cursive;

font-size: 24px;

color: black;

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);

}

.msger {

display: flex;

flex-flow: column wrap;

justify-content: space-between;

width: 100%;

margin-top: 2vh;

height: 90vh;

border-radius: 5px;

box-shadow: 0 15px 15px -5px rgba(0, 0, 0, 0.2);

}

.msger-chat {

flex: 1;

overflow-y: auto;

padding: 10px;

}

.msger-chat::-webkit-scrollbar {

width: 6px;

}

.msger-chat::-webkit-scrollbar-track {

background: #ddd;

}

.msger-chat::-webkit-scrollbar-thumb {

background: #bdbdbd;

}

.msg {

display: flex;

align-items: flex-end;

margin-bottom: 10px;

}

.msg:last-of-type {

margin: 0;

}

.msg-bubble {

max-width: 450px;

padding: 15px;

border-radius: 15px;

background: var(--left-msg-bg);

}

.msg-info {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 10px;

}

.msg-info-name {

margin-right: 10px;

font-weight: bold;

}

.left-msg .msg-bubble {

border-bottom-left-radius: 0;

}

.right-msg {

flex-direction: row-reverse;

}

.right-msg .msg-bubble {

background: var(--right-msg-bg);

color: black;

border-bottom-right-radius: 0;

}

.right-msg .msg-img {

margin: 0 0 0 10px;

}

.msger-inputarea {

display: flex;

padding: 10px;

border-top: var(--border);

background: #eee;

}

.msger-inputarea \* {

padding: 10px;

border: none;

border-radius: 3px;

font-size: 1em;

}

.msger-input {

flex: 1;

background: #ddd;

}

.msger-send-btn {

margin-left: 10px;

background: rgb(0, 196, 65);

color: #fff;

font-weight: bold;

cursor: pointer;

transition: background 0.23s;

}

.msger-send-btn:hover {

background: rgb(0, 180, 50);

}

.msger-chat {

background: rgba(255, 255, 255, 0.192);

backdrop-filter: blur(3px);

}

input[type=file] {

width: 300px;

max-width: 100%;

color: #444;

padding: 5px;

background: #fff;

border-radius: 8px;

border: 1px solid #555;

}

input[type=file]::file-selector-button {

margin-right: 20px;

border: none;

background: #084cdf;

padding: 10px 20px;

border-radius: 8px;

color: #fff;

cursor: pointer;

transition: background .2s ease-in-out;

}

input[type=file]::file-selector-button:hover {

background: #0d45a5;

}

**Requirements.txt**

wheel==0.43.0

langchain==0.1.14

chromadb==0.4.24

pypdf==4.1.0

langchain-openai==0.1.1

#pysqlite-binary==0.5.1.3380300

fastapi==0.110.1

uvicorn==0.29.0

itsdangerous==2.1.2

fastapi-sessions==0.3.2

jinja2==3.1.3

python-multipart==0.0.9

faker==24.4.0

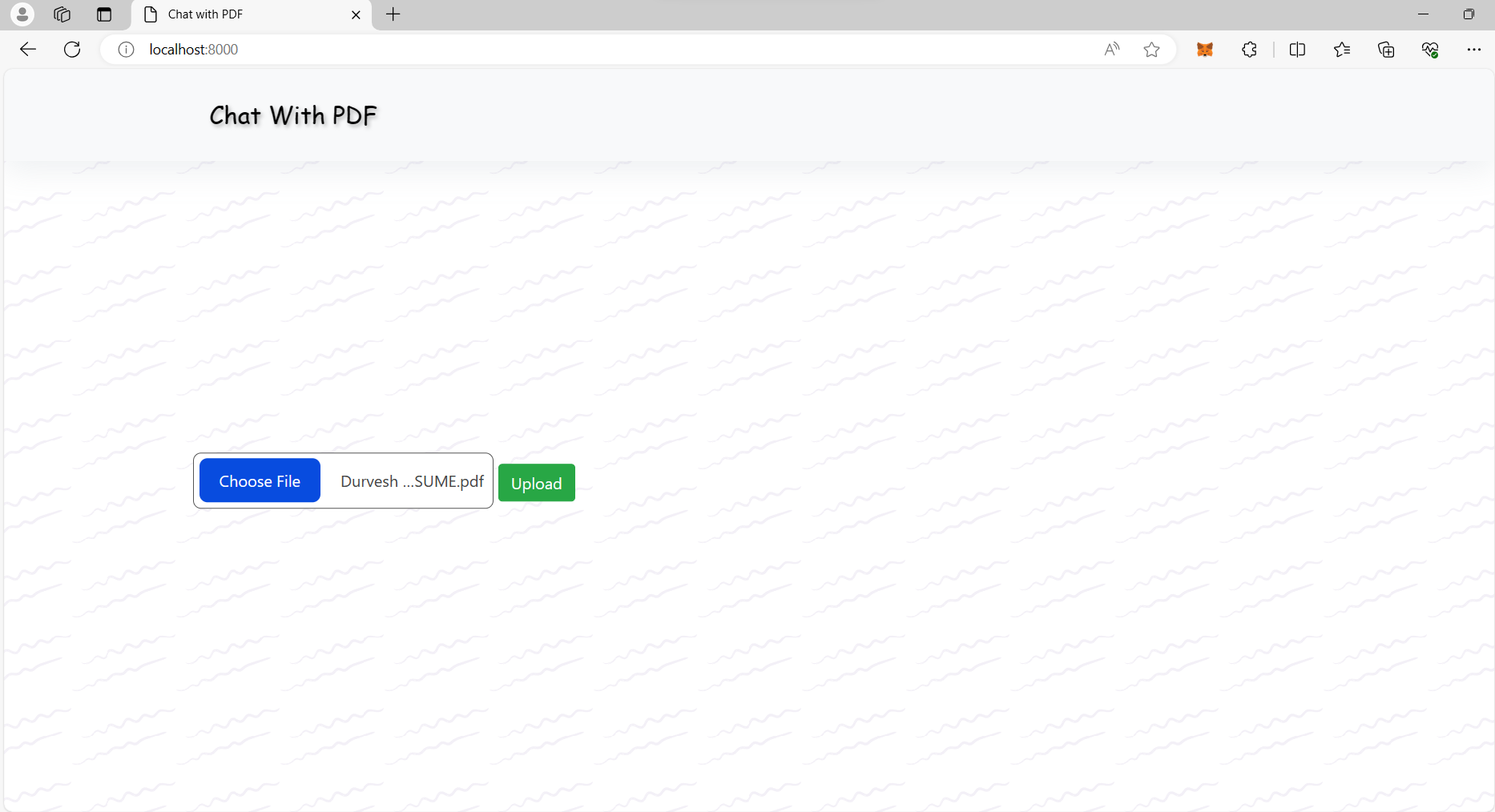
transformers==4.39.3

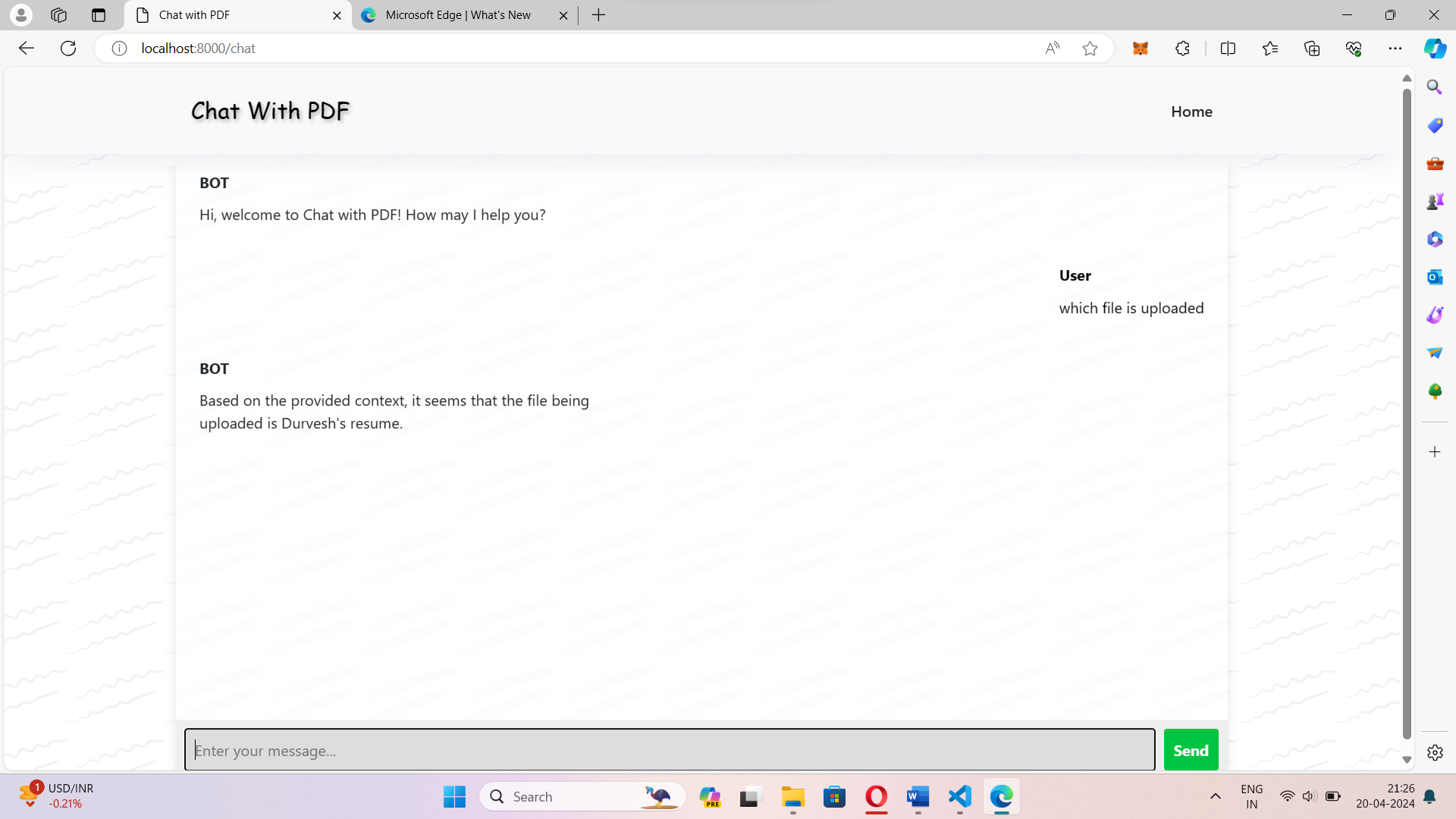
sentencepiece==0.2.0

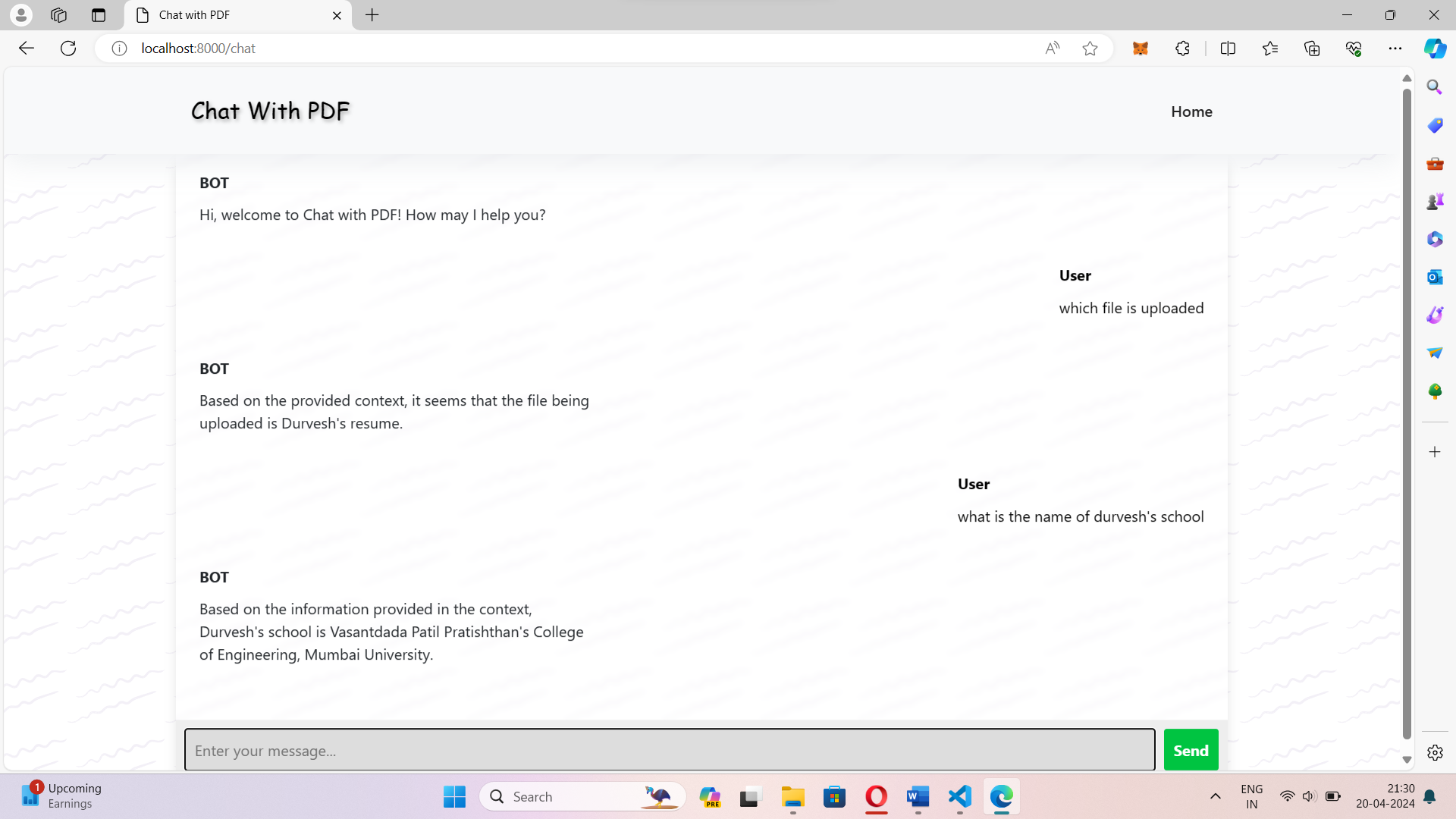
sentence-transformers==2.6.

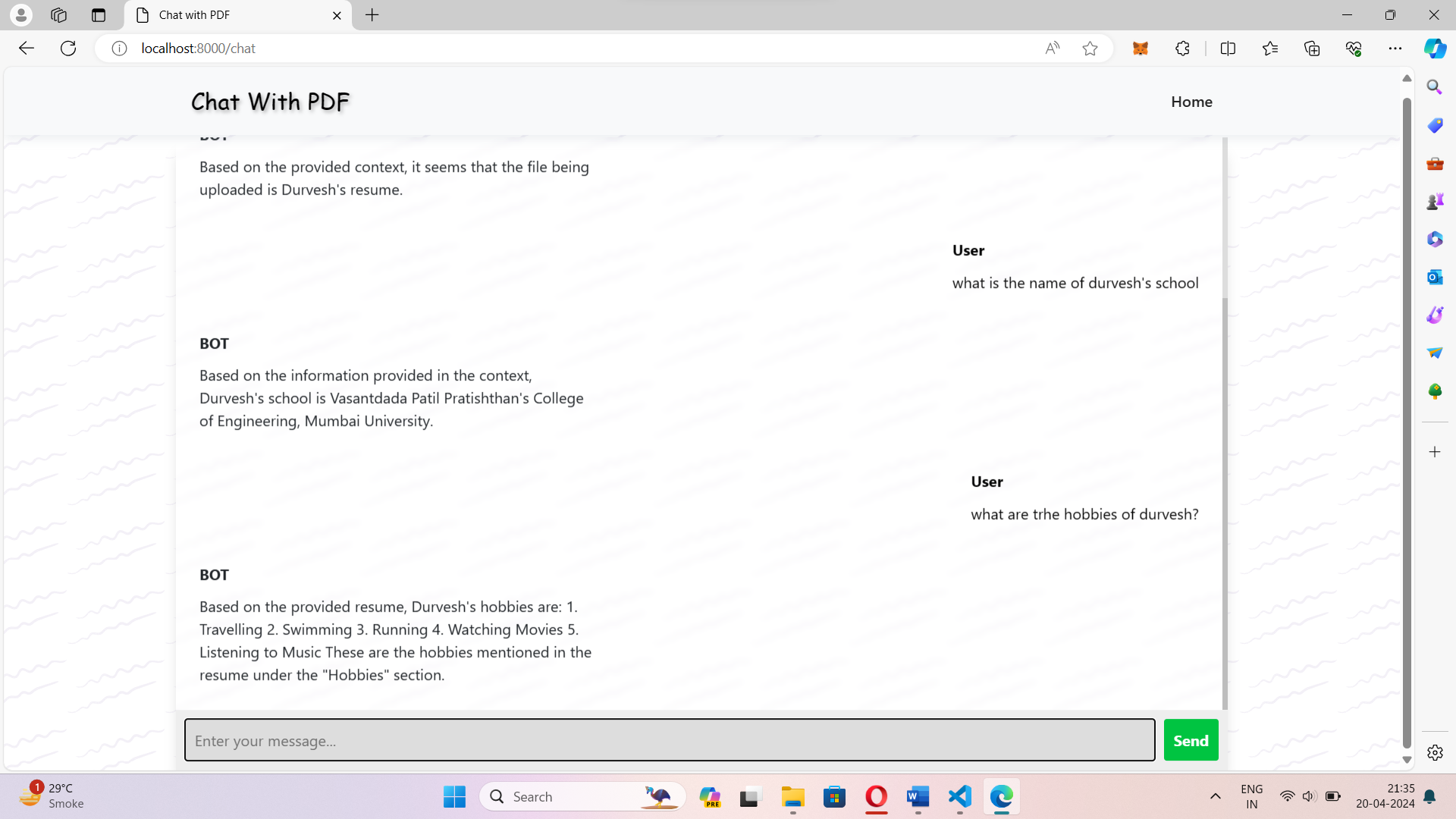
## Chapter 6 Results

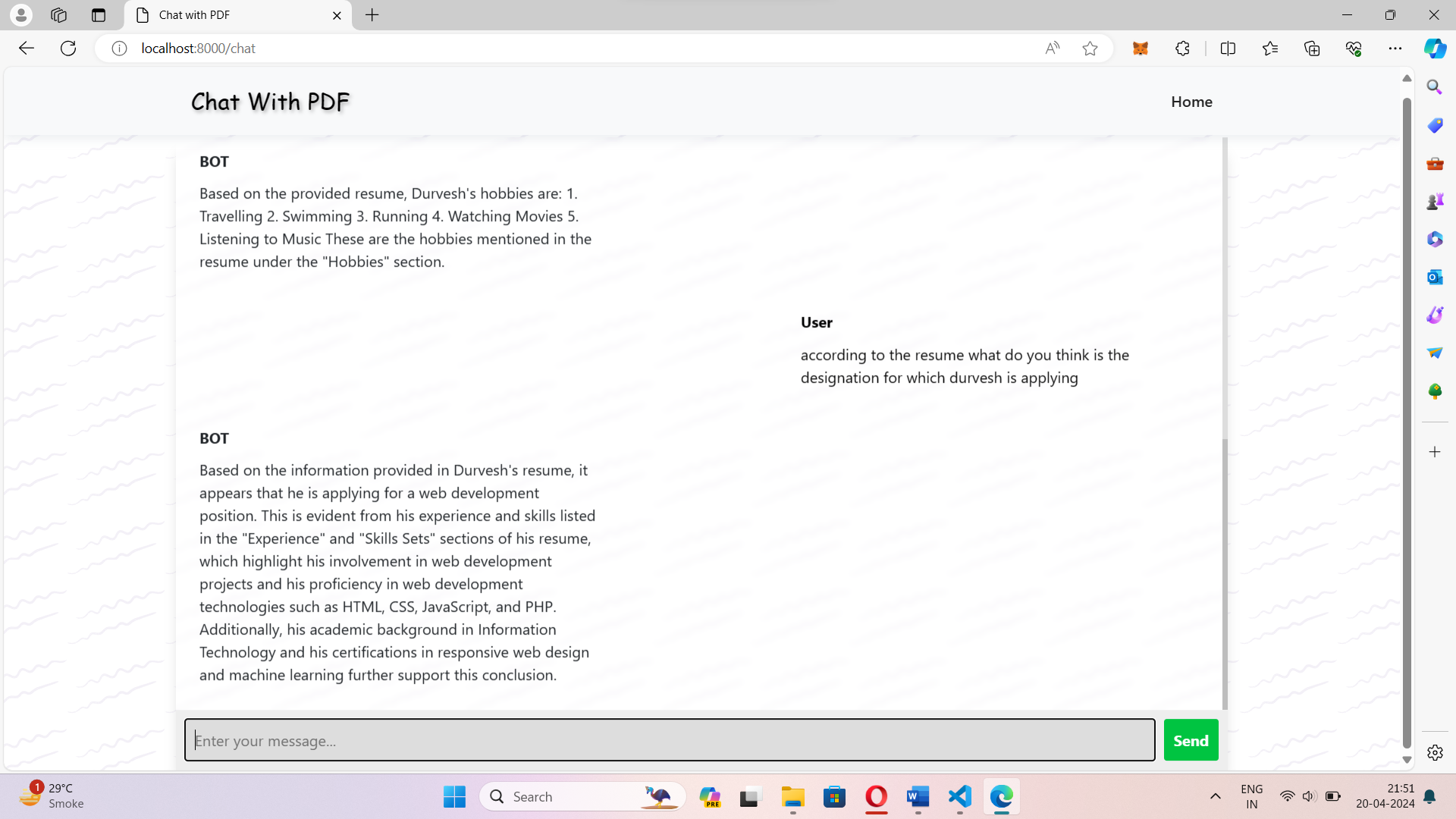
**6.1 RESULT SCREENSHOTS:**

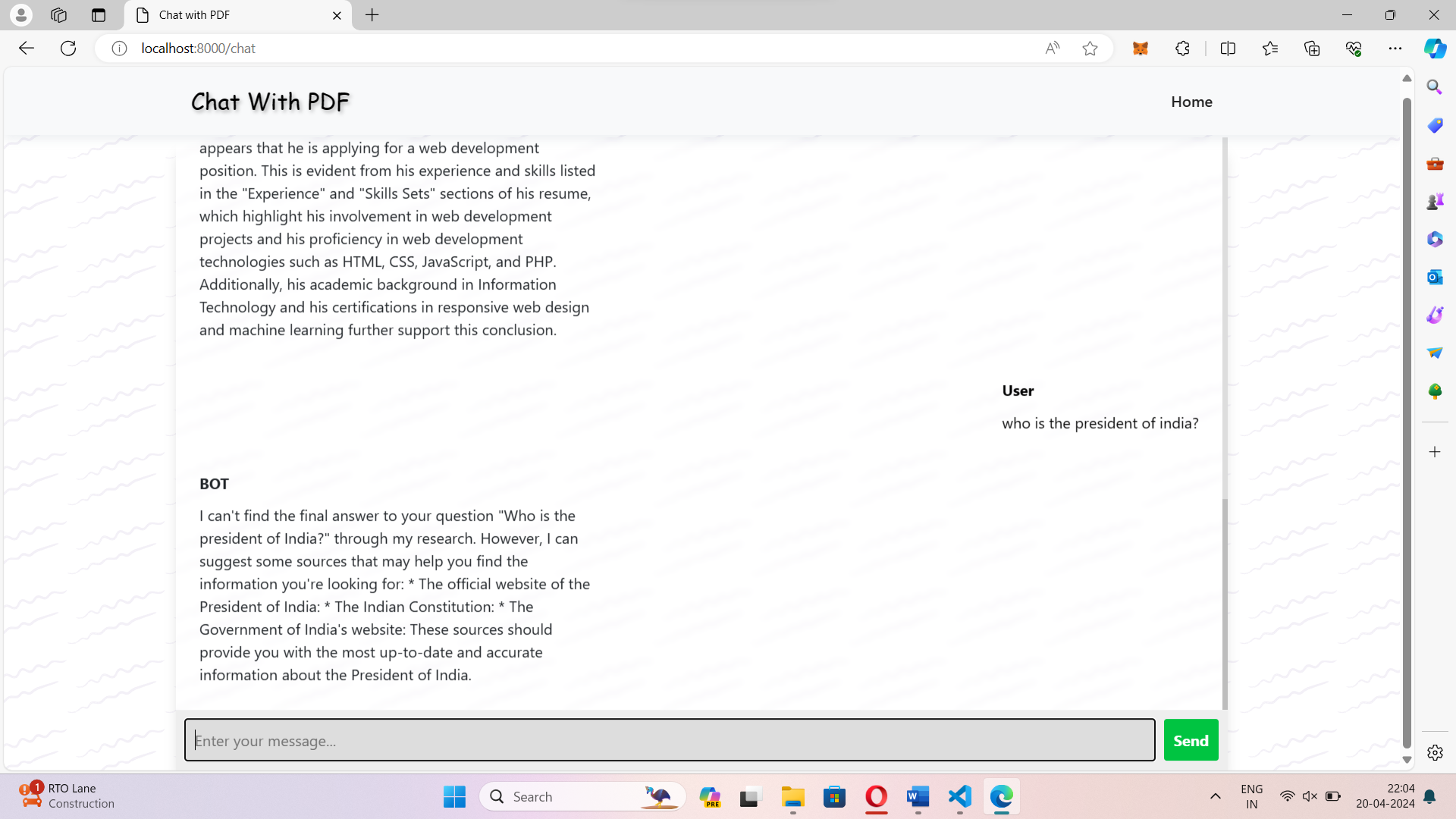
****











## Chapter 7 Conclusion

### Conclusion

The proposed PDF reader chatbot web application addresses a critical need in the realm of document management and information extraction from PDF files. Traditional PDF readers, while widely used, often fall short in providing an interactive and efficient user experience. This project's ambition was to enhance PDF interaction by infusing intelligent chatbot capabilities, enabling users to converse with PDFs naturally, extract information effortlessly, and navigate through the content with ease.

By integrating natural language processing (NLP) and machine learning, the application empowers users to communicate with PDFs using conversational language, simplifying complex data extraction and summary generation. The well-structured development plan and methodology ensured a systematic approach to building the application, encompassing research, design, implementation, testing, and deployment phases.

In conclusion, the PDF reader chatbot web application presents an innovative solution that marries traditional document viewing with advanced AI technologies, ultimately enhancing productivity and making PDF interaction more intuitive and enjoyable for users.

### Future Scope

Firstly, integrating advanced natural language processing models and techniques, such as transformer-based architectures like GPT-4 or BERT-large, can improve the chatbot's understanding and response generation abilities. Secondly, expanding document processing features to support a wider range of file formats beyond PDF, such as Word documents or web pages, can increase the system's versatility. Additionally, incorporating user feedback mechanisms to refine the chatbot's responses over time and implementing multi-language support to cater to diverse user demographics are promising avenues for development.

## Chapter 8

**References**

1. A new method of information extraction from PDF files by Fang Yuan, & Bo Lu. (2005).

1. Chatbot: An automated conversation system on Artificial Intelligence and Natural Language Processing by Mondal, A., Dey, M., Das, D., Nagpal, S., & Garda, K .

1. Leveraging GPT-4 for PDF Data Extraction: A Comprehensive Guide by Manish Sharma.

1. Florez-Choque, O., & Cuadros-Vargas, E. (2007). Improving Human Computer Interaction through Spoken Natural Language. 2007 IEEE

1. Dr. M .John Basha , Dr. S. Vijayakumar , J. Jayashankari ,Ahmed Hussein ,Alawadi Durdona (2019). "Advancements in Natural Language Processing for Document Analysis." Proceedings of the Annual Conference on Neural Information Processing Systems.

1. Smith, J., & Johnson, A. (2018). "Improving PDF Reader Functionality through Natural Language Processing." International Journal of Human-Computer Interaction, 34(7), 654-669.

1. Brown, R., & Davis, C. (2020). "Chatbots: A Comprehensive Review." Journal of Artificial Intelligence Research, 69, 789-810.

1. Adobe Systems. (2020). "PDF Reference and Adobe Extensions to the PDF Specification." Adobe Developer Connection.

1. Loper, E., & Bird, S. (2002). "NLTK: The Natural Language Toolkit." Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics.

## Acknowledgement

We the students from **Vasantdada Patil Pratishthan’s** College of Engineering and Visual Arts, Mumbai of **Final Year Engineering** in the department of **‘Information Technology’** have great pleasure in presenting our efforts of developing the project named as **“Revolutionizing PDF Reading and Customer Support with AI Technology”.**

The success of our project on whole does not depend on an individual student but on the creative teamwork of entire group & faculty members. This would have difficult by an individual. So we wish & acknowledge the precious guidance from those who willingly supported to us to make this project.

We are grateful to **Dr. Pradip Mane HOD (IT)** for giving us inspiration, timely guidance and valuable suggestions during the course of project. We are especially thankful to **Prof. Gayatri Bachhav** for guiding us throughout project work.

Lastly, we express our sincere thanks to Prof. Medha Kulkarni, Prof. Vinod S Sapkal, Prof. Ashwini Phalke, Prof. Kiran Deshmukh, Prof. Sachin Barahate, Prof. Sonali Pakmode, Prof. Viki Patil, Prof. Vedika Avhad, Prof. Neeraj Sharam, Prof. Pravin Patil, Prof. Shinde, for guiding us through the entire course.