### INTERNSHIP REPORT

*A report submitted in partial fulfillment of the requirements for the Award of Degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**By**

**KF SURYA**

**Reg. No.: 711322101058**

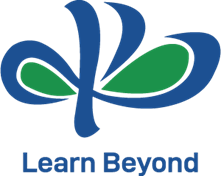
**Under Supervision of**

**Mayur Dev Sewak**

**Head, Training & Internships**

**EI Systems Services**

**(Period : 13/07/2024 to 12/08/2024)**



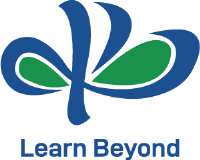
**KPR INSTITUTE OF ENGINEERING AND TECHNOLOGY**

(Autonomous ,NAAC ‘A’)

Avinashi Road, Arasur

**COIMBATORE – 641 407**

**JULY 2024**

KPR Institute of Engineering and Technology

**BONAFIDE CERTIFICATE**

This is to certify that the **Internship report** submitted by **KF SURYA (22AD058)** is work done by him and submitted during the academic year 2023 – 2024, in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in **Artificial Intelligence and Data Science**, at **EI Systems Services.**

**Department IIPC Coordinator Head of the Department**

**Mr. SELVAKUMAR G Dr. SUDHA S V**

**Department of Artificial Intelligence Department of Artificial Intelligence**

**and Data Science and Data Science**

**Place:Coimbatore KPR Institute of Engineering and Technology**

**Date:16/08/2024**



### ACKNOWLEDGEMENT

First I would like to thank **Mayur Dev Sewak Head, Training & Internships, EI Systems Services** for giving me the opportunity to undergo industry training within the organization.

I would also like to thank all the people who worked along with me at Ei Systems **Services** for their patience and openness in sharing knowledge they created for an enjoyable working environment.

It is indeed with a great sense of pleasure and immense sense of gratitude that I acknowledge the help of these individuals.

I am highly indebted to our Chairman **Dr. K.P. RAMASAMY**, our Chief Executive Officer **Dr. A. M. NATARAJAN** and our beloved Principal **Dr. D. SARAVANAN** for the facilities provided to accomplish this internship.

I would like to thank my Head of the Department **Dr. SUDHA S V** for her supportive actions by permitting me for my internship.

I would like to thank **Dr. R. KIRUBA SHANKAR**, Professor & Head IIPC, KPRIET for his support and advice to get an internship and complete the same in the above said organization.

I am extremely grateful to my department IIPC coordinator **Mr. SELVAKUMAR G**, faculty members and friends who helped me in successful completion of this internship.

**KF SURYA**

**22AD058**

## ABSTRACT

During my internship at Ei Systems as an AI Intern, I contributed to the development of advanced artificial intelligence solutions, focusing on retrieval-augmented generation (RAG) systems and medical question-answering applications. My primary responsibilities included designing and implementing two key projects: **MediSaga - RAG Chat bot Application** and **High-Performance CPU RAG System with Quantized Llama2**.

**MediSaga** utilized RAPTOR indexing with the Milvus vector database and the Llama2 language model to create an efficient medical question-answering system. I developed the entire pipeline from PDF text extraction and query expansion to deploying a user-friendly interface via Streamlit.

In the **High-Performance CPU RAG System**, I optimized a retrieval-augmented generation framework using a quantized version of Llama2, enabling efficient performance on CPU-only systems without the need for GPU resources. This project demonstrated significant improvements in computational efficiency, making advanced AI accessible on standard hardware.

Throughout this internship, I honed my skills in natural language processing, vector databases, and AI model optimization, while contributing to real-world applications that push the boundaries of AI capabilities.

### TABLE OF CONTENTS

| **Chapter No** | **Title** | **Page No** |
| --- | --- | --- |
|  | **Offer Letter** | **III** |
|  | **Acknowledgement** | **IV** |
|  | **Abstract** | **V** |
| **1** | **About the Company** | **1** |
| **2** | **Project Summary** | **3** |
| **3** | **Data Flow Diagram** | **7** |
| **4.** | **Code** | **8** |
| **5** | **Image/ Video/ Links - Output** | **21** |
| **6** | **References** | **23** |
| **7** | **Plan of your internship program** | **25** |
| **8** | **Attendance Sheet**  **Industry Student Daily Diary Industry Supervisor Evaluation Sheet** | **29** |
| **9** | **Conclusion** | **32** |

### Chapter 1

### About the Company

### Introduction of the Organization

EISystems is a leading technology training provider in India, specializing in areas such as Cybersecurity, Machine Learning, IoT, Robotics, and Social Media. With a mission to bridge the gap between technology and education, EISystems has impacted around 200,000 students through its comprehensive training programs and outreach initiatives.

### Vision, Mission, and Values

**Vision:** To be a global leader in technology education, fostering innovation and continuous learning.

**Mission:** To provide high-quality, industry-relevant training programs that enhance both theoretical knowledge and practical skills.

**Values:**

* **Excellence:** Commitment to high standards in education and training.
* **Innovation:** Embracing new technologies and practices.
* **Integrity:** Upholding ethical standards and fostering trust.
* **Collaboration:** Partnering with educational institutions and industry leaders.
* **Empowerment:** Equipping individuals with essential skills and confidence.

### Policy of the Organization in Relation to the Intern Role

**Learning and Development:**

* **Structured Programs:** Interns receive well-organized training aligned with their career goals.
* **Skill Enhancement:** Focus on both technical and soft skills.

**Mentor ship and Guidance:**

* **Mentor Assignment:** Each intern has a dedicated mentor for support and career advice.
* **Regular Reviews:** Periodic performance evaluations.

**Work Environment:**

* **Supportive Culture:** Collaborative and inclusive work environment.
* **Professional Development:** Opportunities for workshops and networking.

**Ethics and Conduct:**

* **Professional Standards:** Adherence to ethical and professional conduct.
* **Confidentiality:** Maintenance of sensitive information confidentiality.

**Feedback and Improvement:**

* **Continuous Improvement:** Use of intern feedback to enhance programs.
* **Open Communication:** Encouragement of feedback and addressing challenging

**CHAPTER 2**

**Project Summary**

### Idea Behind the Project

The idea behind MediSaga emerged from the growing need for accessible, accurate, and comprehensive medical information. With the vast amount of medical knowledge available, health care professionals and individuals often face challenges in retrieving relevant information quickly. Traditional search engines and databases may provide a wealth of information but often lack the contextual understanding necessary for precise medical answers. MediSaga addresses this challenge by integrating advanced Retrieval-Augmented Generation (RAG) techniques to create a chatbot capable of processing and answering complex medical queries.

**Objectives:**

1. **Enhanced Information Retrieval:** To improve the efficiency and accuracy of retrieving medical information from extensive textual data.
2. **User-Friendly Interaction:** To provide a seamless and intuitive user interface for querying medical information.
3. **Contextual Understanding:** To leverage advanced NLP and ML models to understand and respond to queries in a contextually relevant manner.

### About the Project

MediSaga is a sophisticated medical question-answering chatbot application that combines various advanced technologies to deliver accurate responses. The project integrates multiple components, each serving a critical role in the overall functionality of the application.

1. **PDF Processing:**
   * **Text Extraction:** Utilizes pdfplumber to extract text from medical textbooks stored in PDF format. This process involves reading and parsing PDFs to convert them into structured text.
   * **Text Chunking:** The extracted text is divided into manageable chunks. This step is crucial for effective indexing and retrieval, ensuring that each chunk contains relevant information for query processing.
2. **Indexing:**
   * **Embedding Generation:** Sentence-Transformers are employed to generate high-quality embeddings for each text chunk. These embeddings capture the semantic meaning of the text, facilitating effective similarity search.
   * **RAPTOR Indexing:** Implements RAPTOR for indexing the text chunks. RAPTOR is a sophisticated method that enhances retrieval by organizing and processing text hierarchically.
   * **Milvus Vector Database:** The embeddings and metadata are stored in Milvus, a high-performance vector database. Milvus supports efficient similarity search and is crucial for handling large-scale data.
3. **Query Processing:**
   * **Similarity Search:** Uses Milvus to perform similarity search based on user queries. This involves comparing the query against indexed text chunks to find the most relevant information.
   * **Re-Ranking:** Retrieves and re-ranks results to ensure the highest relevance. This step involves adjusting the order of retrieved information to prioritize the most pertinent responses.
4. **Response Generation:**
   * **Llama3 Integration:** The re-ranked results are input to the Llama3 model via Ollama. Llama3 is an advanced language model that generates coherent and contextually appropriate answers based on the retrieved information.
   * **Answer Synthesis:** Combines information from multiple sources to formulate a comprehensive response to the user's query.
5. **User Interface:**
   * **Streamlit Interface:** Develops an intuitive chat-based interface using Streamlit. This interface allows users to input their queries and view responses in a user-friendly manner. It also provides information about the sources used for generating the answers.

### Software Used in the Project

The development of MediSaga involved several software tools and libraries, each contributing to the project's success:

* **Python 3.8+:** The primary programming language for developing the application. Python's rich ecosystem of libraries and frameworks was instrumental in implementing various components of the project.
* **Milvus:** An open-source vector database used for storing and querying text embeddings. Milvus enables efficient similarity search and retrieval of relevant information.
* **Sentence-Transformers:** A library for generating embeddings of text segments. It provides pre-trained models that convert text into vector representations suitable for similarity search.
* **PyTorch:** A deep learning framework used for training and deploying models. PyTorch's flexibility and performance were crucial for implementing and fine-tuning the Llama3 model.
* **NLTK & scikit-learn:** Libraries for natural language processing and machine learning. They provided tools for text processing, feature extraction, and model evaluation.
* **pdfplumber:** A Python library used for extracting text from PDF documents. It supports accurate text extraction from complex PDF layouts.
* **Streamlit:** A framework for building interactive web applications. Streamlit facilitated the development of a user-friendly interface for the chatbot.
* **Ollama:** A tool for interacting with the Llama3 model. Ollama simplifies the integration of advanced language models into applications.

### 

### 

### 

### Technical Apparatus Requirements

Before initiating the development of MediSaga, several technical requirements and setups were necessary:

1. **Software Requirements:**
   * **Python 3.8+:** Required for executing project scripts and managing dependencies.
   * **Docker & Docker Compose:** Needed for setting up and running the Milvus vector database.
   * **Milvus Vector Database:** Essential for storing and querying text embeddings.
   * **Sentence-Transformers & PyTorch:** Required for generating embeddings and deploying the Llama3 model.
2. **Hardware Requirements:**
   * **Computational Resources:** Adequate hardware for running the application and models. This includes a robust CPU or GPU for handling model inference and data processing tasks.
   * **Storage:** Sufficient storage for handling large PDF files, indexed data, and model files.
3. **Data Preparation:**
   * **Medical Textbooks:** Access to relevant medical textbooks in PDF format for processing and indexing.
   * **Configuration:** Setup of config.json to manage various parameters, including model paths, Milvus configuration, and other settings.

### Result or Working of the Project

The MediSaga application successfully demonstrates its capabilities in several ways:

* **Effective Information Retrieval:** The application efficiently processes and indexes large volumes of medical text, making it accessible for user queries.
* **Accurate Query Responses:** Utilizes advanced indexing and similarity search techniques to retrieve relevant information and generate precise answers using the Llama3 model.
* **User-Friendly Interface:** Provides an interactive chat-based interface that simplifies user interactions and delivers answers in a clear and accessible manner.

**Performance Evaluation:**

* **Relevance of Answers:** Assessed the quality and relevance of the responses generated by the chatbot. The evaluation involved comparing the answers to expert-reviewed medical information.
* **System Efficiency:** Evaluated the performance of the indexing, retrieval, and response generation processes to ensure timely and accurate results.

### 

### Research Conducted

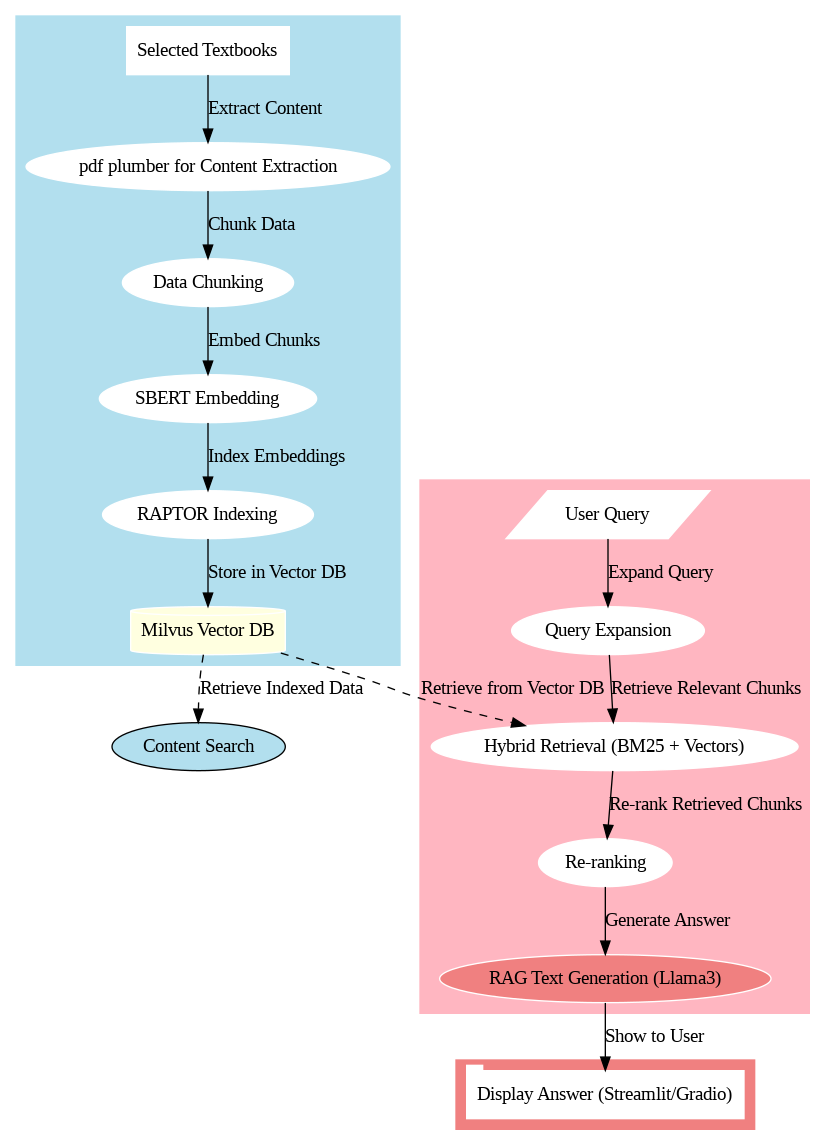
The development of MediSaga involved extensive research in several areas:

1. **RAPTOR Indexing:**
   * **Research Focus:** Investigated advanced text indexing methods to enhance retrieval performance. RAPTOR was chosen for its ability to process and organize text hierarchically, improving search efficiency.
   * **Implementation:** Implemented RAPTOR indexing using Sentence-Transformers embeddings and Milvus for storing and querying indexed data.
2. **Model Integration:**
   * **Llama3 Model:** Researched the capabilities of Llama3 for generating high-quality responses. Integrated the model using Ollama to facilitate interaction and answer generation.
   * **Performance Testing:** Conducted tests to evaluate the model's effectiveness in understanding and responding to medical queries.
3. **System Evaluation:**
   * **Query Processing and Retrieval:** Analyzed the performance of the query processing and retrieval components to ensure they meet the project's accuracy and efficiency goals.
   * **User Feedback:** Gathered feedback from test users to identify areas for improvement and validate the overall functionality of the chatbot.

The MediSaga project represents a significant advancement in the field of medical information retrieval, showcasing the successful application of advanced NLP and ML techniques to address real-world challenges. The combination of sophisticated indexing, retrieval, and response generation processes, along with a user-friendly interface, demonstrates the project's potential to enhance access to accurate medical information.

## Chapter 3

**Data Flow Diagram**



**Fig1: Workflow of Medisaga RAG Chatbot**

## 

## Chapter 4

**Code**

**Code 1: config.json**

The config.json file contains various configuration parameters:

* sbert\_model: The Sentence-BERT model to use for embeddings (default: 'all-MiniLM-L6-v2')
* llm\_model: The language model to use for question answering (default: 'llama2')
* chunk\_size: The size of text chunks for processing (default: 100)
* pdf\_directory: The directory containing the PDF files (default: 'data')
* milvus\_host: Milvus server host (default: 'localhost')
* milvus\_port: Milvus server port (default: '19530')
* collection\_name: The name of the Milvus collection (default: 'medical\_qa')
* vector\_dim: Dimension of the vector embeddings (default: 384)
* top\_k: Number of top results to retrieve (default: 5)
* batch\_size: Batch size for processing (default: 10)

{

"sbert\_model": "all-MiniLM-L6-v2",

"llm\_model": "llama3",

"chunk\_size": 100,

"pdf\_directory": "data",

"milvus\_host": "localhost",

"milvus\_port": "19530",

"collection\_name": "medical\_qa",

"vector\_dim": 384,

"top\_k": 5

}

**Fig2: Code of parameter configuration**

**Code 2: docker-compose.yml**

This Docker Compose file is used to define and orchestrate a multi-container environment, specifically for setting up the Milvus vector database system along with its dependencies, Etcd and Minio.  
  
version: '3.5'

services:

etcd:

container\_name: milvus-etcd

image: quay.io/coreos/etcd:v3.5.5

environment:

- ETCD\_AUTO\_COMPACTION\_MODE=revision

- ETCD\_AUTO\_COMPACTION\_RETENTION=1000

- ETCD\_QUOTA\_BACKEND\_BYTES=4294967296

- ETCD\_SNAPSHOT\_COUNT=50000

volumes:

- ${DOCKER\_VOLUME\_DIRECTORY:-.}/volumes/etcd:/etcd

command: etcd -advertise-client-urls=http://127.0.0.1:2379 -listen-client-urls http://0.0.0.0:2379 --data-dir /etcd

healthcheck:

test: ["CMD", "etcdctl", "endpoint", "health"]

interval: 30s

timeout: 20s

retries: 3

minio:

container\_name: milvus-minio

image: minio/minio:RELEASE.2023-03-20T20-16-18Z

environment:

MINIO\_ACCESS\_KEY: minioadmin

MINIO\_SECRET\_KEY: minioadmin

ports:

- "9001:9001"

- "9000:9000"

volumes:

- ${DOCKER\_VOLUME\_DIRECTORY:-.}/volumes/minio:/minio\_data

command: minio server /minio\_data --console-address ":9001"

healthcheck:

test: ["CMD", "curl", "-f", "http://localhost:9000/minio/health/live"]

interval: 30s

timeout: 20s

retries: 3

standalone:

container\_name: milvus-standalone

image: milvusdb/milvus:v2.3.3

command: ["milvus", "run", "standalone"]

security\_opt:

- seccomp:unconfined

environment:

ETCD\_ENDPOINTS: etcd:2379

MINIO\_ADDRESS: minio:9000

volumes:

- ${DOCKER\_VOLUME\_DIRECTORY:-.}/volumes/milvus:/var/lib/milvus

healthcheck:

test: ["CMD", "curl", "-f", "http://localhost:9091/healthz"]

interval: 30s

start\_period: 90s

timeout: 20s

retries: 3

ports:

- "19530:19530"

- "9091:9091"

depends\_on:

- "etcd"

- "minio"

networks:

default:

name: milvus

**Fig3: Code for Docker Compose for Milvus Vector Database Integration.**

**Code3: Preprocessing.py**

* It contains functions for PDF processing, text chunking, and RAPTOR indexing

import pdfplumber

import nltk

from nltk.tokenize import sent\_tokenize, word\_tokenize

from sentence\_transformers import SentenceTransformer

import numpy as np

from sklearn.mixture import GaussianMixture

from pymilvus import connections, Collection, FieldSchema, CollectionSchema, DataType, utility

import json

import os

import ollama

import logging

from functools import lru\_cache

# Adjust logging

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

logger = logging.getLogger(\_\_name\_\_)

logging.getLogger("httpx").setLevel(logging.WARNING) # Reduce Ollama library logging

# Load configuration

try:

with open('config.json') as config\_file:

config = json.load(config\_file)

except FileNotFoundError:

logger.warning("config.json not found. Using default values.")

config = {

'sbert\_model': 'all-MiniLM-L6-v2',

'llm\_model': 'llama2',

'chunk\_size': 100,

'pdf\_directory': 'data',

'milvus\_host': 'localhost',

'milvus\_port': '19530',

'collection\_name': 'medical\_qa',

'vector\_dim': 384,

'top\_k': 5,

'batch\_size': 10

}

# Initialize SentenceTransformer model

sbert\_model = SentenceTransformer(config['sbert\_model'])

def extract\_text\_from\_pdf(pdf\_path):

text = ""

with pdfplumber.open(pdf\_path) as pdf:

for i, page in enumerate(pdf.pages):

text += f"\n\nPage {i + 1}\n\n"

text += page.extract\_text()

return text

def chunk\_text(text, chunk\_size=config['chunk\_size']):

sentences = sent\_tokenize(text)

chunks = []

current\_chunk = ""

current\_length = 0

for sentence in sentences:

sentence\_length = len(word\_tokenize(sentence))

if current\_length + sentence\_length > chunk\_size:

chunks.append(current\_chunk.strip())

current\_chunk = sentence

current\_length = sentence\_length

else:

current\_chunk += " " + sentence

current\_length += sentence\_length

if current\_chunk:

chunks.append(current\_chunk.strip())

return chunks

def embed\_chunks(chunks):

return sbert\_model.encode(chunks)

@lru\_cache(maxsize=None)

def summarize\_single\_text(text):

response = ollama.generate(

model=config['llm\_model'],

prompt=f"Summarize the following text:\n\n{text}\n\nSummary:"

)

return response['response'].strip()

def summarize\_texts\_ollama(texts):

summaries = []

for i in range(0, len(texts), config['batch\_size']):

batch = texts[i:i+config['batch\_size']]

batch\_summaries = [summarize\_single\_text(text) for text in batch]

summaries.extend(batch\_summaries)

return summaries

def create\_raptor\_index(embeddings, texts, depth=3):

if depth == 0 or len(texts) <= 1:

return texts[0] if texts else ""

n\_components = min(5, len(texts))

gmm = GaussianMixture(n\_components=n\_components, random\_state=0)

gmm.fit(embeddings)

labels = gmm.predict(embeddings)

clusters = {i: [] for i in range(n\_components)}

for i, label in enumerate(labels):

clusters[label].append(texts[i])

summaries = []

for cluster in clusters.values():

if cluster:

cluster\_summary = summarize\_texts\_ollama(cluster)[0]

summaries.append(cluster\_summary)

summary\_embeddings = embed\_chunks(summaries)

return create\_raptor\_index(summary\_embeddings, summaries, depth - 1)

def initialize\_milvus():

try:

connections.connect("default", host=config['milvus\_host'], port=config['milvus\_port'])

if not utility.has\_collection(config['collection\_name']):

fields = [

FieldSchema(name="id", dtype=DataType.INT64, is\_primary=True, auto\_id=True),

FieldSchema(name="embedding", dtype=DataType.FLOAT\_VECTOR, dim=config['vector\_dim']),

FieldSchema(name="metadata", dtype=DataType.JSON)

]

schema = CollectionSchema(fields, "Medical QA collection")

collection = Collection(name=config['collection\_name'], schema=schema)

index\_params = {

"index\_type": "IVF\_FLAT",

"metric\_type": "L2",

"params": {"nlist": 1024}

}

collection.create\_index("embedding", index\_params)

else:

collection = Collection(config['collection\_name'])

collection.load()

return collection

except Exception as e:

logger.error(f"Error initializing Milvus: {e}")

return None

def process\_pdf(pdf\_path, collection):

try:

text = extract\_text\_from\_pdf(pdf\_path)

chunks = chunk\_text(text)

embeddings = embed\_chunks(chunks)

raptor\_index = create\_raptor\_index(embeddings, chunks)

entities = [

{

"embedding": embedding.tolist(),

"metadata": json.dumps({

"text": chunk,

"pdf\_path": pdf\_path,

"raptor\_index": raptor\_index

})

}

for embedding, chunk in zip(embeddings, chunks)

]

collection.insert(entities)

logger.info(f"Processed and indexed PDF: {pdf\_path}")

except Exception as e:

logger.error(f"Error processing PDF {pdf\_path}: {e}")

def process\_pdf\_directory(pdf\_directory, collection):

for filename in os.listdir(pdf\_directory):

if filename.endswith(".pdf"):

pdf\_path = os.path.join(pdf\_directory, filename)

process\_pdf(pdf\_path, collection)

def main():

nltk.download('punkt', quiet=True)

collection = initialize\_milvus()

if collection is None:

logger.error("Failed to initialize Milvus. Exiting.")

return

pdf\_directory = config['pdf\_directory']

if not os.path.exists(pdf\_directory):

logger.error(f"PDF directory {pdf\_directory} does not exist.")

return

process\_pdf\_directory(pdf\_directory, collection)

collection.flush()

logger.info("PDF processing and indexing completed.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Fig4: Code for preprocessing the pdf**

**Code 4: Milvus-Integration.py**

* Implements the query processing and answer generation logic using Milvus.

import numpy as np

from pymilvus import connections, Collection

import json

import ollama

import logging

from sentence\_transformers import SentenceTransformer

# Set up logging

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

logger = logging.getLogger(\_\_name\_\_)

# Load configuration

try:

with open('config.json') as config\_file:

config = json.load(config\_file)

except FileNotFoundError:

logger.warning("config.json not found. Using default values.")

config = {

'sbert\_model': 'all-MiniLM-L6-v2',

'llm\_model': 'llama3',

'milvus\_host': 'localhost',

'milvus\_port': '19530',

'collection\_name': 'medical\_qa',

'top\_k': 5

}

# Initialize SentenceTransformer model

sbert\_model = SentenceTransformer(config['sbert\_model'])

def search\_milvus(collection, query\_vector, top\_k=config['top\_k']):

search\_params = {"metric\_type": "L2", "params": {"nprobe": 10}}

results = collection.search(

data=[query\_vector],

anns\_field="embedding",

param=search\_params,

limit=top\_k,

output\_fields=["metadata"]

)

return results

def summarize\_texts\_ollama(texts):

summaries = []

for text in texts:

response = ollama.generate(

model=config['llm\_model'],

prompt=f"Summarize the following text:\n\n{text}\n\nSummary:"

)

summary = response['response'].strip()

summaries.append(summary)

return summaries

def retrieve\_and\_rank(query, collection):

query\_vector = sbert\_model.encode([query])[0].tolist()

search\_results = search\_milvus(collection, query\_vector)

retrieved\_texts = []

for hits in search\_results:

for hit in hits:

metadata = json.loads(hit.entity.get('metadata'))

retrieved\_texts.append({

"content": metadata['text'],

"raptor\_index": metadata['raptor\_index'],

"pdf\_path": metadata['pdf\_path'],

"distance": hit.distance

})

# Re-rank based on relevance (you can implement more sophisticated re-ranking here)

retrieved\_texts.sort(key=lambda x: x['distance'])

return retrieved\_texts

def generate\_answer(query, context):

response = ollama.generate(

model=config['llm\_model'],

prompt=f"Question: {query}\n\nContext: {context}\n\nAnswer:"

)

return response['response'].strip()

def process\_query(query, collection):

retrieved\_texts = retrieve\_and\_rank(query, collection)

context = " ".join([text['content'] for text in retrieved\_texts])

answer = generate\_answer(query, context)

return answer, retrieved\_texts

def initialize\_milvus():

try:

connections.connect("default", host=config['milvus\_host'], port=config['milvus\_port'])

collection = Collection(config['collection\_name'])

collection.load()

return collection

except Exception as e:

logger.error(f"Error connecting to Milvus: {e}")

return None

if \_\_name\_\_ == "\_\_main\_\_":

collection = initialize\_milvus()

if collection:

# Example usage

query = "What are the symptoms of COVID-19?"

answer, sources = process\_query(query, collection)

print(f"Answer: {answer}")

print("Sources:")

for source in sources[:3]: # Print top 3 sources

print(f"- {source['pdf\_path']} (Relevance: {1 / (1 + source['distance']):.2f})")

else:

print("Failed to initialize Milvus connection.")

**Fig 5: Code for the Milvus Integration for vector storing and retrieval process.**

**Code 5 : UI.py**

* It is a Streamlit-based user interface of chatbot

import streamlit as st

from pymilvus import connections, Collection

import logging

from Milvus-Integration import process\_query, initialize\_milvus

from preprocess import process\_pdf\_directory

import os

import json

# Set up logging

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

logger = logging.getLogger(\_\_name\_\_)

# Load configuration

try:

with open('config.json') as config\_file:

config = json.load(config\_file)

except FileNotFoundError:

logger.warning("config.json not found. Using default values.")

config = {

'pdf\_directory': 'data',

'milvus\_host': 'localhost',

'milvus\_port': '19530',

'collection\_name': 'medical\_qa'

}

def main():

# Create a centered div with a logo

st.markdown("""

<div style="display: flex; justify-content: center;">

<img src="https://www.freepnglogos.com/uploads/medicine-logo-png-1.png" alt="Logo" width="100">

</div>

""", unsafe\_allow\_html=True)

st.markdown("<h1 style='text-align: center;'>MediSaga</h1>", unsafe\_allow\_html=True)

st.markdown("<h2 style='text-align: center;'>RAG Chatbot Application</h2>",unsafe\_allow\_html=True)

st.sidebar.title("MediSaga")

st.sidebar.write('''This is RAG Chatbot Application.

It uses Llama3 for the operation.''')

st.sidebar.markdown("<h2>Sources:</h2>",unsafe\_allow\_html= True)

# Define the path to the data folder

data\_folder = "data"

# Get a list of all PDF files in the data folder

pdf\_files = [f for f in os.listdir(data\_folder) if f.endswith('.pdf')]

# Formatting the list of books with paths in Markdown

books\_list = "\n".join([f"- [{pdf\_file}]({os.path.join(data\_folder, pdf\_file)})" for pdf\_file in pdf\_files])

st.sidebar.markdown(books\_list)

collection = initialize\_milvus()

if collection is None:

st.error("Failed to connect to Milvus. Please check the logs for more details.")

return

# Process existing PDFs

pdf\_directory = config['pdf\_directory']

if not os.path.exists(pdf\_directory):

st.error(f"PDF directory {pdf\_directory} does not exist.")

return

# Initialize chat history

if "messages" not in st.session\_state:

st.session\_state.messages = []

# Display chat messages from history on app rerun

for message in st.session\_state.messages:

with st.chat\_message(message["role"]):

st.markdown(message["content"])

# React to user input

if prompt := st.chat\_input("What is your medical question?"):

# Display user message in chat message container

st.chat\_message("user").markdown(prompt)

# Add user message to chat history

st.session\_state.messages.append({"role": "user", "content": prompt})

with st.spinner("Searching and generating answer..."):

answer, sources = process\_query(prompt, collection)

# Display assistant response in chat message container

with st.chat\_message("assistant"):

st.markdown(answer)

st.subheader("Sources:")

for source in sources[:3]: # Display top 3 sources

st.write(f"- {os.path.basename(source['pdf\_path'])} (Relevance: {1 / (1 + source['distance']):.2f})")

# Add assistant response to chat history

st.session\_state.messages.append({"role": "assistant", "content": answer})

collection.release()

if \_\_name\_\_ == "\_\_main\_\_":

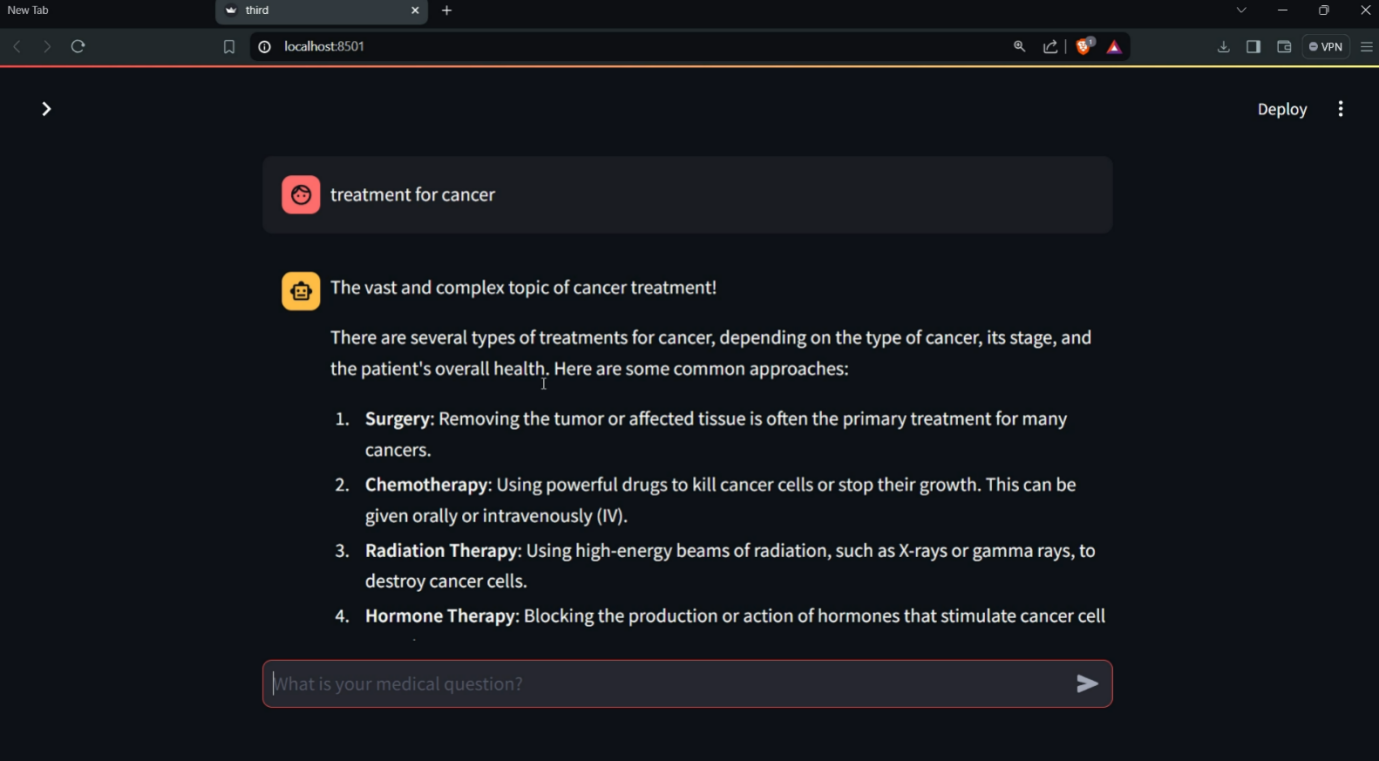
main()

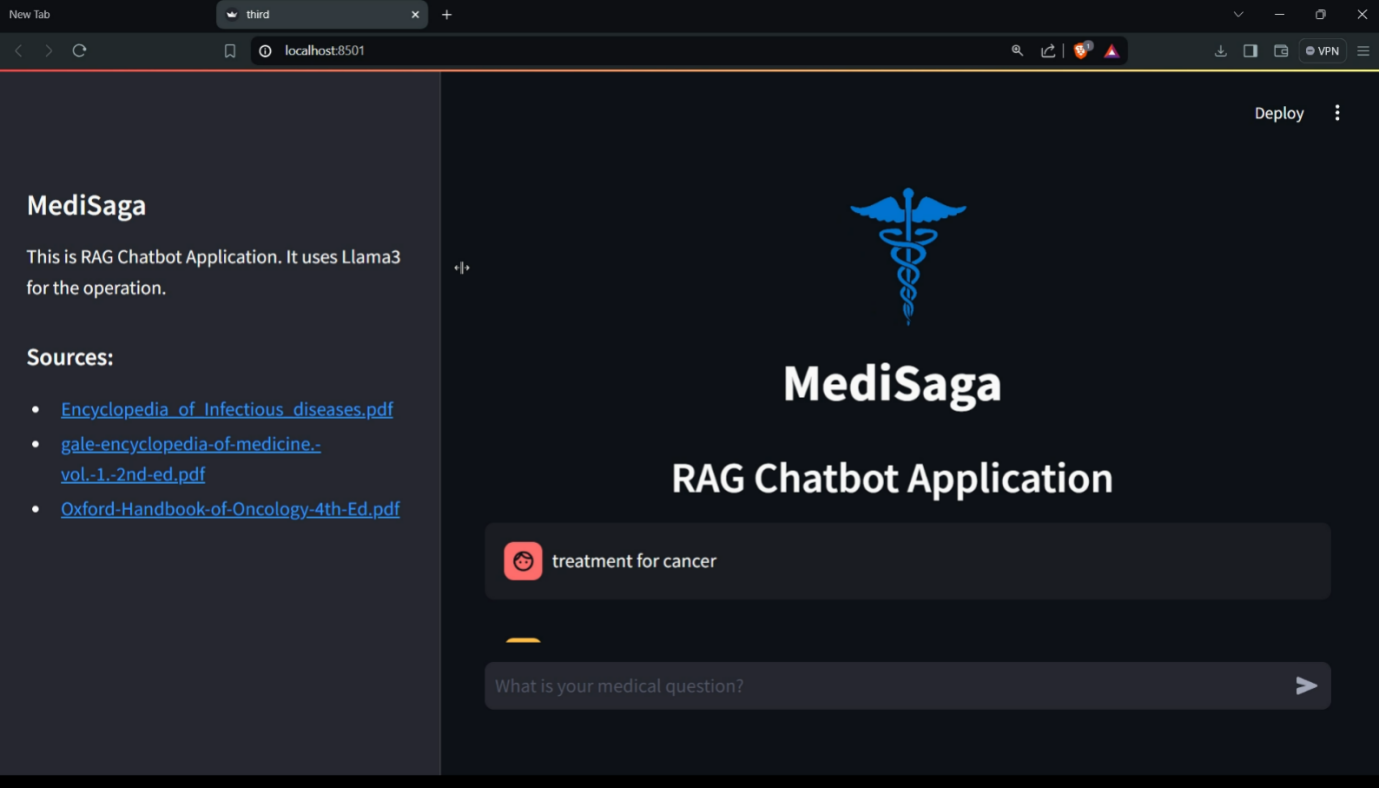
**Fig 6: Code for Streamlit based User-Interface**

**Chapter 5**

**Images / Video/ Links -Output**



****



**Fig7: User Interface of Chatbot**

**Linkedin post Link:** <https://www.linkedin.com/posts/kf-surya_nlp-ai-milvus-activity-7221854199649607680-ThTC?utm_source=share&utm_medium=member_desktop>

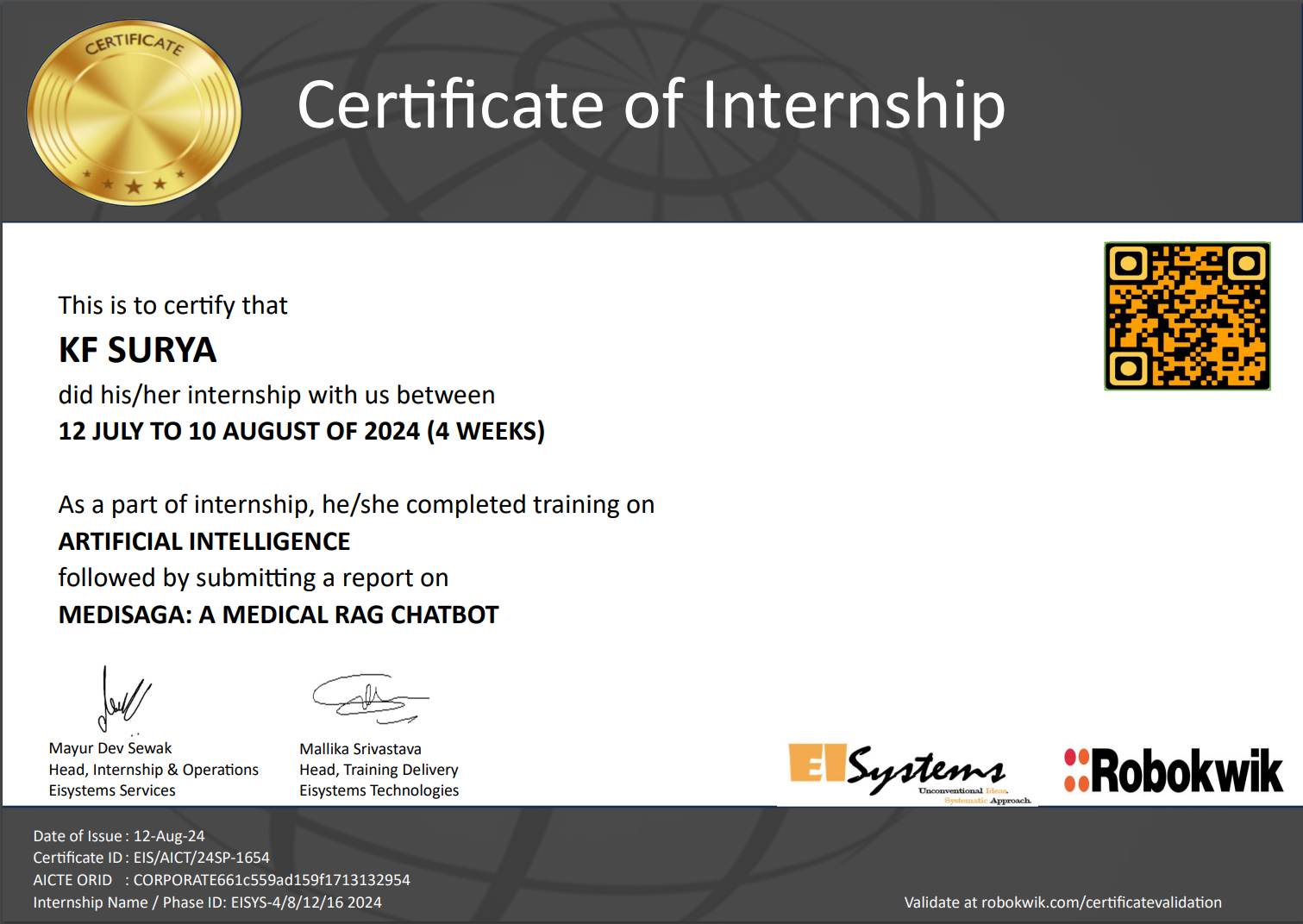
**Github Link:** <https://github.com/Surya-KF/MediSaga-RAG-Chatbot-Application>

**Chapter6**

**References**

Here are some reference links for the key technologies used in this project:

1. **PDFPlumber**:
   * PDFPlumber is a tool for extracting text, tables, and metadata from PDFs. It allows for more precise control over PDF extraction compared to other tools like PyPDF2.
   * [PDFPlumber Documentation](https://github.com/jsvine/pdfplumber)
2. **Sentence-BERT (SBERT)**:
   * SBERT is a modification of the BERT network that uses Siamese and Triplet networks to derive semantically meaningful sentence embeddings. It can be used for tasks like clustering and semantic search.
   * [SBERT Documentation](https://www.sbert.net/)
3. **RAPTOR Indexing**:
   * RAPTOR is a recursive abstractive processing technique that builds a hierarchical index by clustering and summarizing text data, making it efficient for retrieval tasks.
   * [RAPTOR Research Paper](https://arxiv.org/abs/2401.18059)
4. **Milvus Vector Database**:
   * Milvus is an open-source vector database for similarity search and AI applications. It supports storing and managing large-scale vector data and provides efficient retrieval mechanisms.
   * [Milvus Documentation](https://milvus.io/docs)
5. **Dense Passage Retrieval (DPR)**:
   * DPR is a retrieval method that encodes queries and passages into dense vectors and performs similarity search for efficient and accurate retrieval in open-domain question answering systems.
   * [DPR Documentation](https://www.sbert.net/docs/pretrained-models/dpr.html)



## Chapter 7

**Plan of your internship program**

**WEEKLY OVERVIEW OF INTERNSHIP ACTIVITIES**

Week No: \_\_1\_

|  |  |  |  |
| --- | --- | --- | --- |
| **Day & Date** | **Brief Description of Daily Activity** | **Learning Outcome** | **Person In-Charge** |
| Day 1  14 July 2024 | Introduction to Python syntax, operators, and variable declaration | Learned about indentation, dynamic typing, and naming conventions (snake\_case) | Mallika Srivastava |
| Day 2  15 July 2024 | Explored print() function, comments, number and string data types | Mastered string manipulation techniques, including slicing | Mallika Srivastava |
| Day 3  16 July 2024 | Worked with lists and tuples, practicing various operations | Understood the difference between mutable and immutable data structures | Mallika Srivastava |
| Day 4  17 July 2024 | Created and manipulated dictionaries and sets | Learned efficient ways to handle unique, unordered collections of data | Mallika Srivastava |
| Day 5  16 July 2024 | Practiced boolean operations, user input, and typecasting | Understood the importance of data type conversion in programming | Mallika Srivastava |

**Important**  
A day without a training session is used for revision and research purposes.

Week No: \_\_2\_

|  |  |  |  |
| --- | --- | --- | --- |
| **Day & Date** | **Brief Description of Daily Activity** | **Learning Outcome** | **Person In-Charge** |
| Day 1 | Studied and implemented control structures (if-elif-else) | Mastered decision-making in Python programs | Mallika Srivastava |
| Day 2 | Explored different types of loops and list comprehensions | Learned to write more efficient and Pythonic code | Mallika Srivastava |
| Day 3 | Defined and worked with functions, including lambda functions | Understood the concept of code reusability and anonymous functions | Mallika Srivastava |
| Day 4 | Created custom modules and explored built-in Python modules | Learned about code organization and leveraging Python's standard library | Mallika Srivastava |
| Day 5 | Introduction to Object-Oriented Programming concepts | Grasped the fundamentals of classes, objects, and inheritance | Mallika Srivastava |

**Important**  
A day without a training session is used for revision and research purposes.

Week No: \_\_3\_   
(1/2/3/4/5/6/7/8/9/10/11/12/13/14/15/16)

|  |  |  |  |
| --- | --- | --- | --- |
| Day & Date | Brief Description of Daily Activity | Learning Outcome | Person In-Charge |
| Day 1 | Worked with NumPy arrays and Pandas DataFrames | Learned efficient data manipulation and analysis techniques | Mallika Srivastava |
| Day 2 | Created various plots using Matplotlib and observed ML model demos | Understood data visualization techniques and basic ML workflows | Mallika Srivastava |
| Day 3 | Started a data science project, implementing train-test splits | Learned about data preprocessing and model evaluation techniques | Mallika Srivastava |
| Day 4 | Explored advanced ML concepts and model deployment strategies | Understood Support Vector Machines and basics of deploying ML models | Mallika Srivastava |
| Day 5 | Completed final project and underwent comprehensive assessment | Consolidated learning and identified areas for future improvement | Mallika Srivastava |

**Important**  
A day without a training session is used for revision and research purposes.

Week No: \_\_4\_

|  |  |  |  |
| --- | --- | --- | --- |
| Day & Date | Brief Description of Daily Activity | Learning Outcome | Person In-Charge |
| Day 1 | Preprocessed the 3 medical Textbook pdf | Learned advanced text preprocessing techniques and PDF data extraction | Mr. G. Selvakumar |
| Day 2 | Implemented BERT for text embedding and set up Milvus as the vector database | Understood how to create and store vector representations of text | Mr. G. Selvakumar |
| Day 3 | Developed raptor indexing for efficient information retrieval | Learned about advanced indexing techniques for large datasets | Mr. G. Selvakumar |
| Day 4 | Created UI using Streamlit and integrated Ollama to use Llama3 as the LLM | Gained experience in building user interfaces and working with large language models | Mr. G. Selvakumar |
| Day 5 | Integrated NLTK, PyTorch, and Sentence Transformers for enhanced NLP capabilities | Learned to combine various NLP tools and libraries for a comprehensive solution | Mr. G. Selvakumar |

**Important**  
A day without a training session is used for revision and research purposes.

# Industry Institute Partnership Cell

#### KPR Institute of Engineering and Technology

(Autonomous, NAAC – ‘A’)

Avinashi Road, Arasur, Coimbatore – 641 407, Tamil Nadu

**STUDENT’S DAILY DIARY/ DAILY LOG**

|  |  |  |  |
| --- | --- | --- | --- |
| Dept./Division | AD | Area / domain of internship | Cloud Computing |
| Name of Industry Supervisor / Head | EI Systems Services | | |
| Mayur Dev Sewak | | |
| Day 1 (July 12, 2024) - Python Fundamentals:   * Introduced to Python syntax, operators, and variable declaration. Learned about indentation, dynamic typing, and snake\_case naming conventions under Mallika Srivastava's guidance. Wrote initial Python scripts, focusing on code readability and basic language features.   Day 2 (July 13, 2024) - String Manipulation and Data Types:   * Explored core data types, emphasizing string manipulation. Mastered the print() function, comments, and string methods. Practiced string slicing and formatting techniques, including f-strings.   Day 3 (July 14, 2024) - Lists and Tuples:   * Focused on compound data types. Learned list and tuple operations, methods, and use cases. Completed exercises involving nested structures, reinforcing understanding of mutable and immutable sequences.   Day 4 (July 15, 2024) - Dictionaries and Sets:   * Studied dictionaries and sets, understanding their roles in data storage and manipulation. Practiced key-value pair operations and set methods. Applied these structures in data processing scenarios.   Day 5 (July 16, 2024) - Boolean Operations and Type Conversion:   * Delved into boolean operations, user input handling, and type conversion. Wrote complex boolean expressions and interactive programs. Completed a mini-project incorporating all learned concepts.   Day 6 (July 19, 2024) - Control Structures:   * Mastered if-elif-else statements and match-case structures. Implemented nested conditions and ternary operators. Solved coding challenges focusing on complex decision-making logic. | | | |

# Industry Institute Partnership Cell

#### KPR Institute of Engineering and Technology

(Autonomous, NAAC – ‘A’)

Avinashi Road, Arasur, Coimbatore – 641 407, Tamil Nadu

|  |
| --- |
| Day 7 (July 20, 2024) - Loops and List Comprehensions:   * Explored while and for loops, including loop control statements. Learned list comprehensions, comparing their efficiency and readability to traditional loops. Completed exercises demonstrating proficiency in both approaches.   Day 8 (July 21, 2024) - Functions and Lambda Expressions:   * Defined functions with various parameter types and return values. Studied variable scope and the LEGB rule. Explored lambda functions and their applications with built-in functions. Refactored earlier code to improve structure and reusability.   Day 9 (July 22, 2024) - Modules and Libraries:   * Created custom modules and explored built-in and third-party libraries. Set up virtual environments and used pip for package management. Completed a project combining multiple modules to solve a real-world problem.   Day 10 (July 23, 2024) - Introduction to OOP:   * Learned OOP principles: encapsulation, inheritance, and polymorphism. Defined classes, created objects, and implemented inheritance. Explored special methods and multiple inheritance concepts.   Day 11 (July 26, 2024) - Data Manipulation with NumPy and Pandas:   * Utilized NumPy for array operations and Pandas for data manipulation. Practiced data loading, cleaning, and analysis techniques. Completed a data preparation exercise using a real-world dataset.   Day 12 (July 27, 2024) - Data Visualization and ML Basics:   * Created various plots using Matplotlib and Seaborn. Introduced to machine learning concepts and implemented a simple linear regression model using scikit-learn. Observed demos of more complex ML models.   Day 13 (July 28, 2024) - Data Science Project Initiation:   * Began a data science project, focusing on problem definition and data preprocessing. Implemented train-test splits and conducted exploratory data analysis. Applied visualization skills to gain insights from the data.   Day 14 (July 29, 2024) - Advanced ML Concepts:   * Studied advanced model evaluation metrics and feature engineering techniques. Implemented ensemble methods and Support Vector Machines. Explored cross-validation and model interpretability concepts. |

# Industry Institute Partnership Cell

#### KPR Institute of Engineering and Technology

(Autonomous, NAAC – ‘A’)

Avinashi Road, Arasur, Coimbatore – 641 407, Tamil Nadu

|  |
| --- |
| Day 15 (July 30, 2024) - Project Completion and Assessment:   * Refined models using hyperparameter tuning techniques. Finalized the project with a comprehensive report. Discussed model deployment strategies and underwent a thorough assessment of machine learning skills and concepts.   Day 16 (August 2, 2024) - Text Preprocessing:   * Under Mr. G. Selvakumar's guidance, preprocessed medical textbook PDFs. Applied advanced text cleaning, normalization, and tokenization techniques. Developed a custom preprocessing pipeline for medical text data.   Day 17 (August 3, 2024) - Text Embedding and Vector Databases:   * Implemented BERT for contextual text embeddings. Set up Milvus as a vector database for efficient storage and retrieval of high-dimensional data. Practiced similarity searches on medical text embeddings.   Day 18 (August 4, 2024) - Advanced Indexing Techniques:   * Explored approximate nearest neighbor search algorithms. Implemented raptor indexing for large-scale similarity search tasks. Compared performance of different indexing methods on the medical text dataset.   Day 19 (August 5, 2024) - UI Development and LLM Integration:   * Developed a Streamlit web application for the medical text query system. Integrated Ollama to utilize Llama3 as an LLM. Implemented prompt engineering techniques to enhance LLM outputs.   Day 20 (August 6, 2024) - NLP Tools Integration:   * Integrated NLTK for advanced text processing, PyTorch for custom neural network implementation, and Sentence Transformers for enhanced embeddings. Conducted comprehensive testing of the final system on various medical queries. |

## Chapter 8

**Conclusion**

This four-week internship has been an intensive and highly rewarding experience, providing a comprehensive journey through the realms of Python programming, data science, machine learning, and natural language processing.

The program began with a solid foundation in Python fundamentals, progressively building towards more complex concepts and applications. The initial focus on core programming principles seamlessly transitioned into practical data manipulation and analysis techniques using industry-standard libraries such as NumPy and Pandas. The introduction to machine learning concepts and tools, including scikit-learn, not only provided theoretical knowledge but also hands-on experience in implementing and evaluating models.

The latter half of the internship delved into advanced natural language processing techniques, culminating in the development of a sophisticated medical text query system. This project integrated cutting-edge technologies such as BERT embeddings, vector databases, and large language models, demonstrating the practical applications of NLP in the healthcare domain.

Throughout the internship, there was a strong emphasis on project-based learning, allowing for the application of theoretical concepts to real-world scenarios. The progression from basic programming to complex system development has significantly enhanced my technical skills and problem-solving abilities.

Key takeaways from this internship include:

1. Proficiency in Python programming and its application in data science

2. Practical experience with machine learning workflows and model evaluation

3. Advanced knowledge in NLP techniques and their implementation

4. Exposure to state-of-the-art technologies like BERT and LLMs

5. Development of a functional, end-to-end NLP system for medical text analysis

This internship has not only bolstered my technical capabilities but also provided valuable insights into the software development lifecycle and best practices in data science and AI projects. The skills and knowledge acquired during this period have laid a strong foundation for my future career in the field of data science and artificial intelligence.