LIFECAPSULE: AN AI-POWERED MEMORY MANAGEMENT APPLICATION

A MINI-PROJECT REPORT

Submitted by

ANAND S **2116220701026**
AVINASH S **2116220701034**

in partial fulfilment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

RAJALAKSHMI ENGINEERING COLLEGE

AUTONOMOUS, CHENNAI

NOV/DEC, 2023

BONAFIDE CERTIFICATE

Certified that this mini project **"LifeCapsule: An AI-Powered Memory Management Application**" is the bonafide work of "**ANAND S (2116210701245), AVINASH S (2116220701034)"** who carried out the project work under my supervision.

SIGNATURE

Mr.N. Duraimurugan,

Assistant Professor,

Computer Science & Engineering,

Rajalakshmi Engineering College

Thandalam, Chennai -602105.

Submitted for the End semester practical examination to be held on

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

I express my sincere thanks to my beloved and honorable chairman **MR. S. MEGANATHAN** and the chairperson **DR. M. THANGAM MEGANATHAN** for their timely support and encouragement.

I am greatly indebted to my respected and honorable principal **Dr. S. N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by my head of the department **Dr. P. KUMAR,** and my Academic Head **Dr. R. SABITHA,** for being ever supporting force during my project work.

I also extend my sincere and hearty thanks to my internal guide **Mr. N. DURAIMURUGAN** for his valuable guidance and motivation during the completion of this project.

My sincere thanks to my family members, friends and other staff members of Computer Science and Engineering.

Anand S (2116220701026)

Avinash S (2116220701034)

ABSTRACT

LifeCapsule integrates advanced AI techniques to enable memory management and personalized insights for users. The application combines Llama 3.2, LangChain, and Chroma to create a journaling tool that offers seamless natural language-based memory retrieval. This paper details the proposed system's architecture, algorithms, and implementation, emphasizing its potential in assisting users with mental well-being and self-reflection. Performance metrics demonstrate LifeCapsule’s efficiency in managing and retrieving user-generated memories while maintaining data privacy.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ABSTRACT	**4**
1	**INTRODUCTION**	**6**
	1.1 INTRODUCTION	6
	1.2 SCOPE OF THE WORK	6
	1.3 PROBLEM STATEMENT	6
	1.4 AIM AND OBJECTIVES OF THE PROJECT	7
2	**SYSTEM SPECIFICATIONS**	**8**
	2.1 HARDWARE SPECIFICATIONS	8
	2.2 SOFTWARE SPECIFICATIONS	8
3	**ARCHITECTURE DIAGRAM**	**9**
4	**MODULE DESCRIPTION**	**10**
5	**SYSTEM DESIGN**	**11**
	5.1 USECASE DIAGRAM	11
	5.2 E-R MODEL	12
	5.3 DATAFLOW DIAGRAM	13
	5.4 ACTIVITY DIAGRAM	15
6	**CODING**	**18**
7	**SCREENSHOTS**	**20**
8	**CONCLUSION**	**26**
	REFERENCES	27

CHAPTER 1

1.1 INTRODUCTION

LifeCapsule is a transformative tool that allows users to store, manage, and retrieve personal memories through a conversational interface. It addresses the increasing need for digital tools that promote self-reflection and mental well-being.

1.2 SCOPE OF THE WORK

The scope of LifeCapsule includes providing secure, private, and efficient memory management. It focuses on personal journaling with advanced AI features like query-based retrieval, summary generation, and emotional insight analysis.

1.3 PROBLEM STATEMENT

In the modern digital age, people rely heavily on journaling applications to document their lives and reflect on past experiences. However, many existing solutions are limited in functionality, offering only basic text storage without providing meaningful insights or easy retrieval of memories. These applications often depend on cloud storage, raising significant concerns about data privacy and security. Users need a more advanced tool that not only stores their personal entries securely but also enables them to retrieve memories contextually, analyze patterns, and offer personalized reflections. Furthermore, the lack of natural language-based interaction makes the process less intuitive and engaging, leaving users without a truly impactful journaling experience.

1.4 AIM AND OBJECTIVES OF THE PROJECT

The aim of LifeCapsule is to create an innovative, AI-powered journaling application that enhances personal memory management by combining secure data storage with advanced natural language-based retrieval and analysis. The objectives include enabling users to store their daily events seamlessly, retrieve past entries using context-aware AI models, and gain meaningful insights into their memories to promote self-reflection and growth. LifeCapsule ensures data privacy by processing all information locally on the user's device while offering a user-friendly interface for an engaging journaling experience. This holistic approach bridges the gap between technology and personal well-being, empowering users with a tool that not only preserves their memories but also makes them actionable and insightful.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

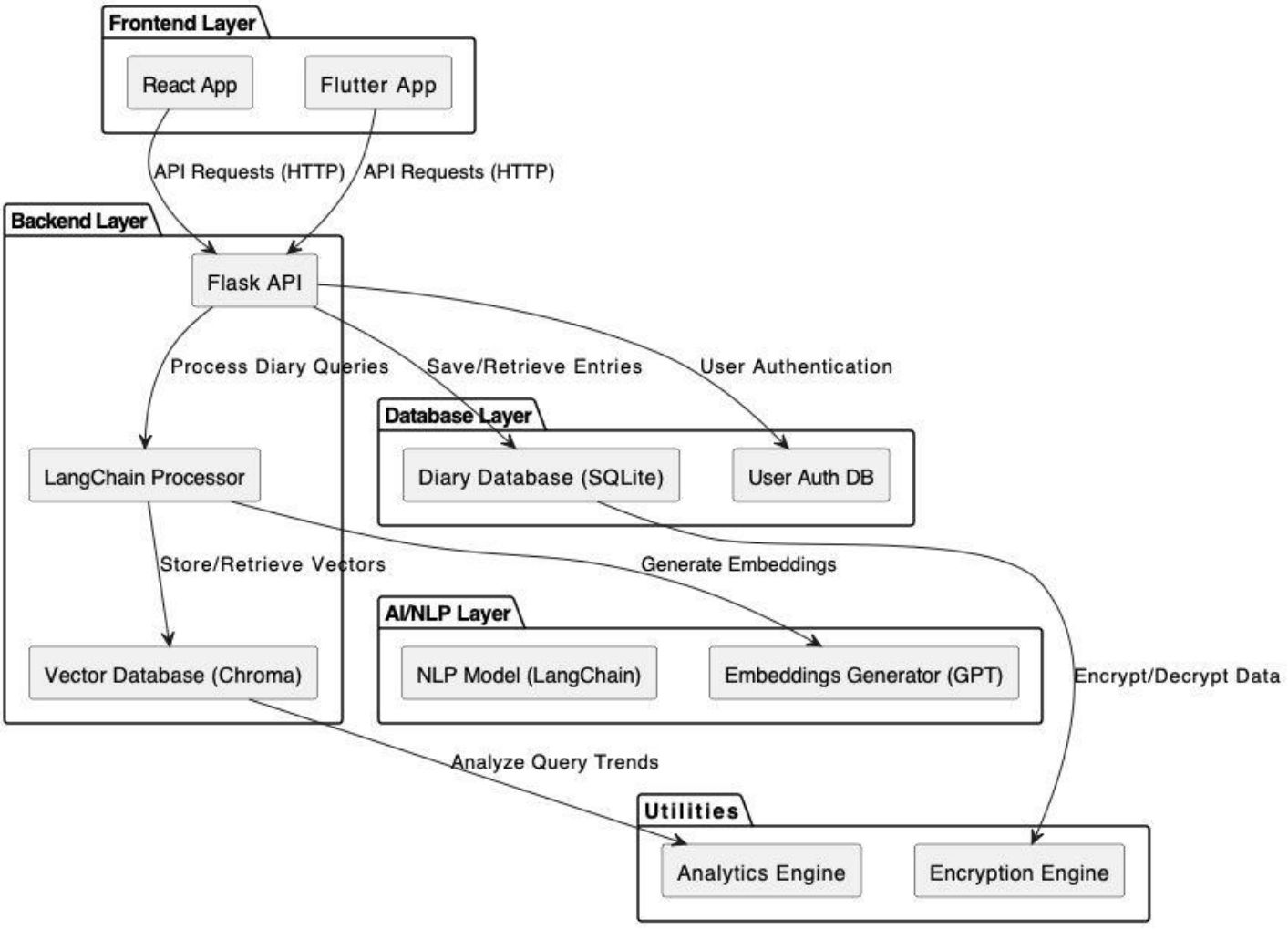
Processor	:	Intel i5 or higher
Memory Size	:	8GB RAM or more
HDD	:	20GB free disk space

2.2 SOFTWARE SPECIFICATIONS

Operating System	:	Windows 10 or Higher
Front – End	:	React.js
Back – End	:	Flask (Python)
Database	:	Chroma for vector storage
Language Model	:	Llama 3.2

CHAPTER 3

ARCHITECTURE DIAGRAM



CHAPTER 4

MODULE DESCRIPTION

4.1. User Registration and Login Module:

This module allows users to register and create an account on the website. Users can provide their basic information, such as name, email address, and password, to create an account. After successful registration, users can log in to the website using their name, registered email address and password.

4.2. Diary Module:

The course module of a college management system allows faculty to manage.
Organize course materials, assignments, grades, and student communication.
It also enables students to access and submit coursework, view grades, and interact.

4.3. Query Module:

This module provides basic information about our website, it has all the contents and the creator details.

4.4. Personal Assistant Module

This module has the details of statement or legal document (in privacy law) that discloses some or all of the ways a party gathers, uses, discloses, and manages a customer or client's data.

4.5. Review Module:

The Review module allows users to review courses they have purchased. Users can provide feedback and share their experience with other users, helping them make informed purchase decision.

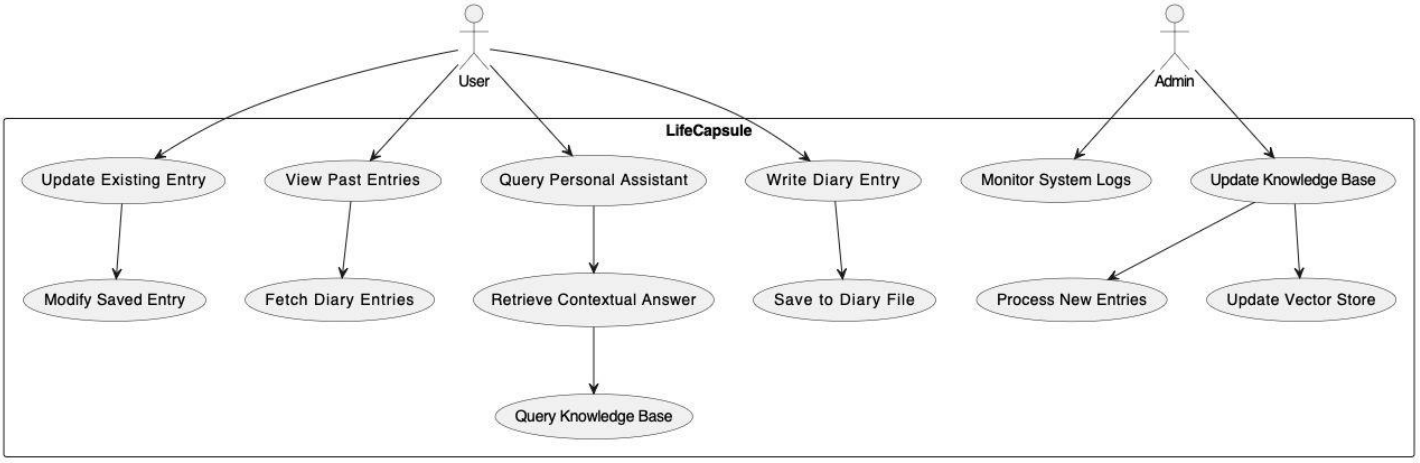
4.6. Admin Module (Optional):

The Admin Dashboard module provides the website's admin with complete control over the website's content and user management. The module allows the admin to add products, categories, and brands. The admin can also view products, user details, manage orders, and track the website's performance.

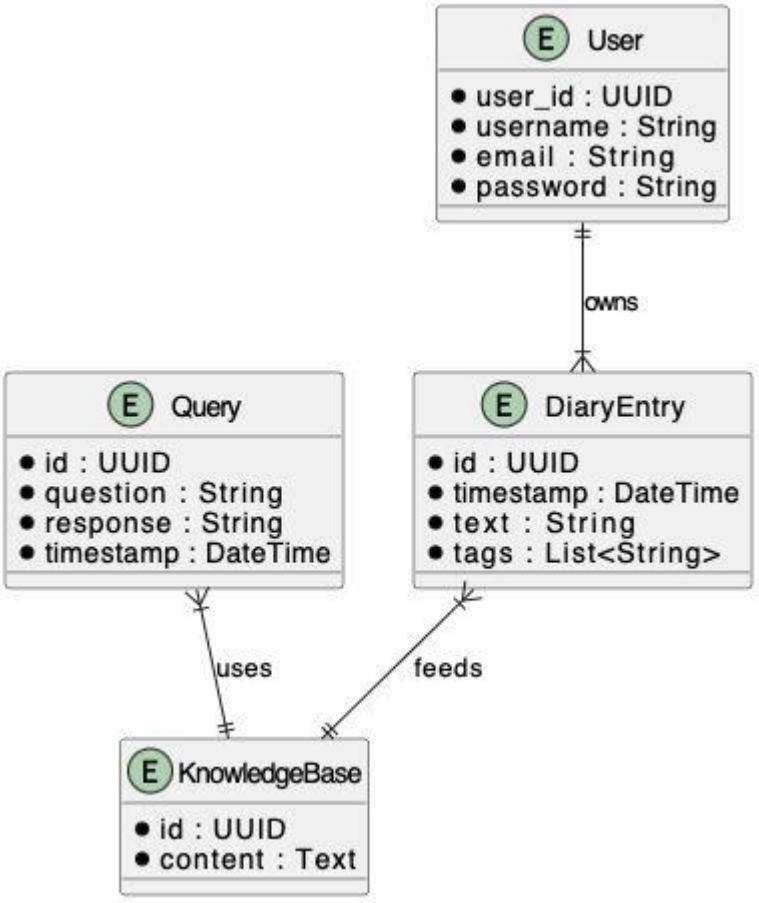
CHAPTER 5

SYSTEM DESIGN

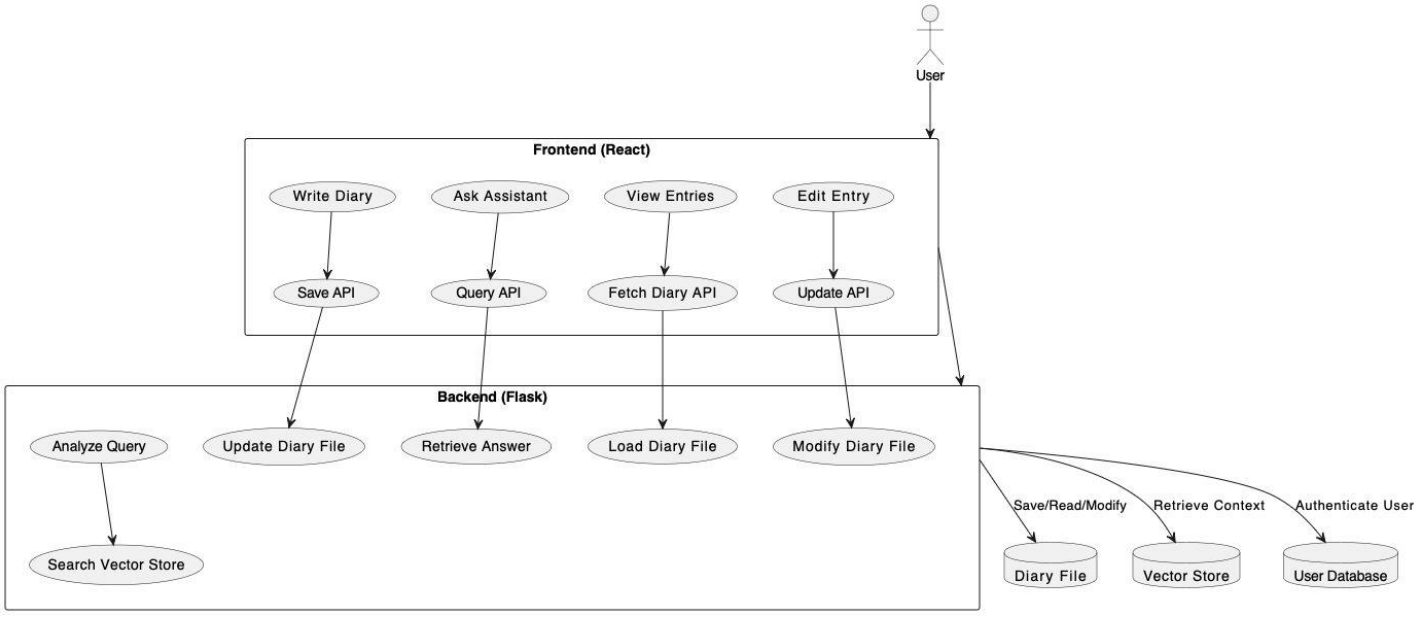
5.1 USE CASE DIAGRAM



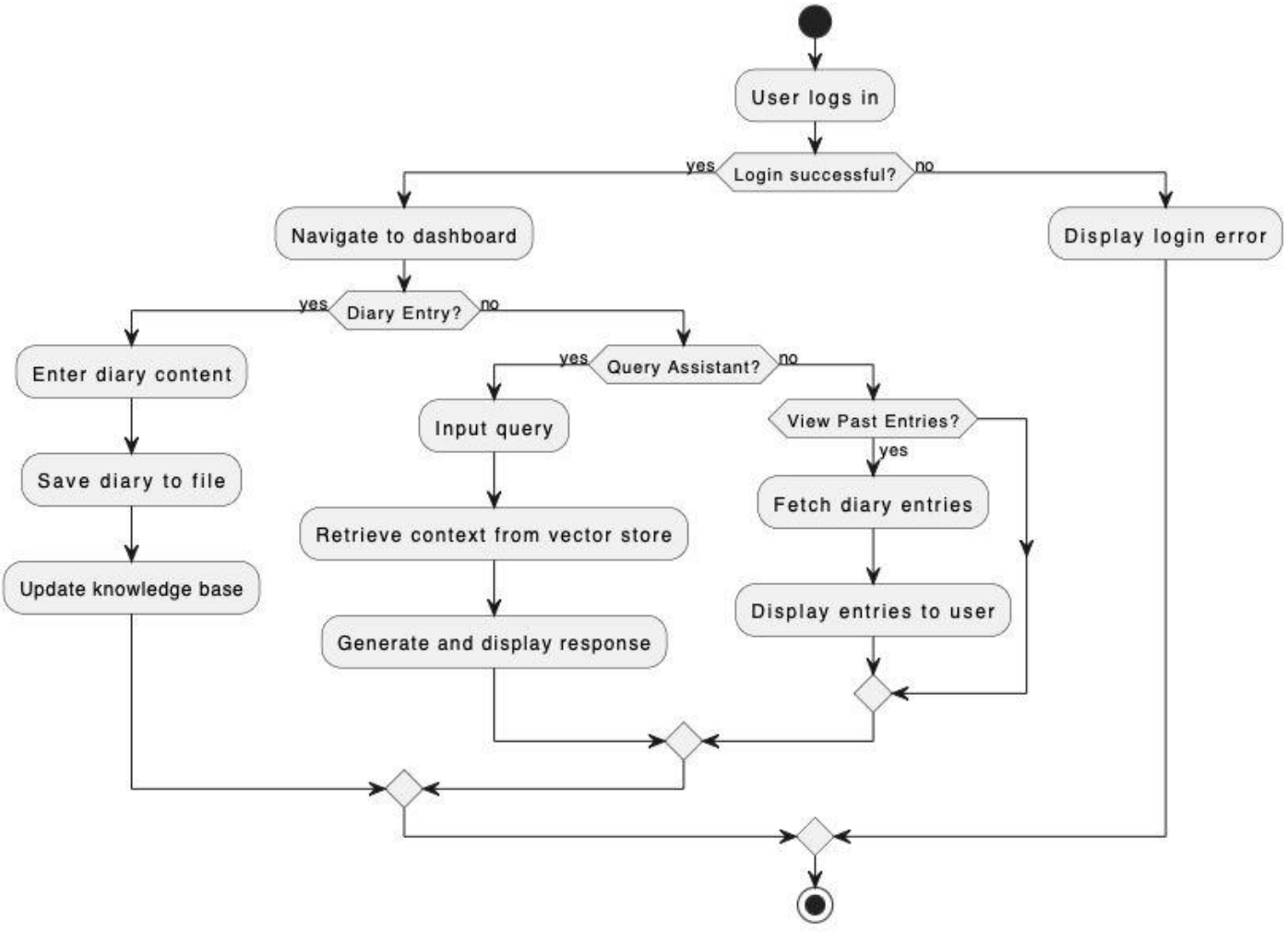
5.2 ER DIAGRAM



5.3 DFD DIAGRAM



5.4 ACTIVITY DIAGRAM



CHAPTER 6

SAMPLE CODING

App.js (REACT.js)

```
import React, { useState } from 'react';
import './App.css';

function App() {
  const [activePage, setActivePage] = useState('diary');
  const [sidebarOpen, setSidebarOpen] = useState(false);
  const [diaryEntry, setDiaryEntry] = useState('');
  const [query, setQuery] = useState('');
  const [assistantResponse, setAssistantResponse] = useState('');

  const handleHover = () => setSidebarOpen(true);
  const handleLeave = () => setSidebarOpen(false);

  const submitDiaryEntry = async () => {
    try {
      const response = await fetch('http://localhost:5000/save_diary', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ entry: diaryEntry }),
      });
      if (response.ok) {
        alert('Diary entry saved!');
        setDiaryEntry('');
      } else {
        alert('Failed to save the diary entry.');
      }
    } catch (error) {
      console.error('Error saving diary entry:', error);
    }
  };

  const askAssistant = async () => {
```

```
  try {
    const response = await fetch('http://localhost:5000/analyze_diary', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ query }),
    });
    const data = await response.json();
    setAssistantResponse(data.answer || 'No response received.');
  } catch (error) {
    console.error('Error querying the assistant:', error);
    setAssistantResponse('Failed to get a response. Please try again later.');
  }
};

return (
  <div className="app">
    <header className="header">
      <h1>LifeCapsule</h1>
    </header>
    <div
      className={`sidebar ${sidebarOpen ? 'open' : ''}`}
      onMouseEnter={handleHover}
      onMouseLeave={handleLeave}
    >
      <ul>
        <li
          onClick={() => setActivePage('diary')}
          className={activePage === 'diary' ? 'active' : ''}
        >
          <span className={`icon ${sidebarOpen ? '' : 'collapsed'}`}>📖</span>
          {sidebarOpen && <span className="menu-text">Diary Entry</span>}
        </li>
        <li
          onClick={() => setActivePage('assistant')}
          className={activePage === 'assistant' ? 'active' : ''}
        >
          <span className={`icon ${sidebarOpen ? '' : 'collapsed'}`}>🤖</span>
          {sidebarOpen && <span className="menu-text">Personal Assistant</span>}
        </li>
```

```
      </ul>
    </div>
    <div className="content">
      {activePage === 'diary' ? (
        <div className="diary">
          <h2 className="section-title">Diary Entry</h2>
          <textarea
            placeholder="Write your diary entry here..."
            value={diaryEntry}
            onChange={(e) => setDiaryEntry(e.target.value)}
            className="diary-input"
          />
          <button onClick={submitDiaryEntry} className="submit-button">
            Save Entry
          </button>
        </div>
      ) : (
        <div className="assistant">
          <h2 className="section-title">Personal Assistant</h2>
          <input
            type="text"
            placeholder="Ask something..."
            value={query}
            onChange={(e) => setQuery(e.target.value)}
            className="query-input"
          />
          <button onClick={askAssistant} className="ask-button">
            Submit Query
          </button>
          <div className="response-box">{assistantResponse}</div>
        </div>
      )}
    </div>
  </div>
 );
}

export default App;
```

App.py(Python)

```
from flask import Flask, request, jsonify
from flask_cors import CORS
from langchain.prompts import PromptTemplate
from langchain_ollama import OllamaLLM
from langchain_community.vectorstores import Chroma
from langchain_community.embeddings import OllamaEmbeddings
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.chains import RetrievalQA
import os
from datetime import datetime
import torch  # Import PyTorch

# Flask app setup
app = Flask(__name__)
CORS(app)

# Constants
DIARY_FILE_PATH = "./diary.txt"
DEVICE = "cuda" if torch.cuda.is_available() else "cpu"

# Global Variables
vectorstore = None
qa_chain = None

# Initialize the Ollama model
ollama_llm = OllamaLLM(model="llama3.2:3b", streaming=False, device=DEVICE)

# Initialize Embeddings
embeddings = OllamaEmbeddings(model="llama3.2:3b", device=DEVICE)

# Initialize Text Splitter
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)

# Function to load diary entries
def load_diary_entries(file_path):
    if os.path.exists(file_path):
        with open(file_path, 'r') as file:
            return file.read()
    return ""

# Function to save diary entry
def save_diary_entry(file_path, entry):
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    with open(file_path, 'a') as file:
```

```
        file.write(f"\n[{timestamp}] {entry}")

# Update the knowledge base
def update_knowledge_base():
    global vectorstore, qa_chain
    diary_content = load_diary_entries(DIARY_FILE_PATH)

    if diary_content.strip():
        # Split diary content
        split_texts = text_splitter.split_text(diary_content)

        # Create a vector store with the embeddings
        vectorstore = Chroma.from_texts(split_texts, embeddings)

        # Create a retrieval chain
        qa_chain = RetrievalQA.from_chain_type(
            ollama_llm,
            retriever=vectorstore.as_retriever(),
            return_source_documents=True
        )
        print("Knowledge base updated.")
    else:
        vectorstore = None
        qa_chain = None
        print("No entries found in the diary. Knowledge base is empty.")

# Process a user query using the Ollama model
def process_prompt_with_model(prompt):
    # Define a prompt template
    prompt_template = PromptTemplate(
        input_variables=["user_input"],
        template="User input: {user_input}. Please provide a response."
    )

    # Format the input
    formatted_prompt = prompt_template.format(user_input=prompt)

    # Use the Ollama model to generate a response
    try:
        result = ollama_llm.invoke(formatted_prompt)
        return result
    except Exception as e:
        print(f"Error invoking Ollama model: {e}")
        return "Error generating a response from the model."

# Analyze the diary
def analyze_diary(query):
```

```
    if not query.strip():
        return "Please provide a valid question."

    if not qa_chain:
        return "The knowledge base is empty. Please add diary entries first."

    try:
        # Query the knowledge base
        result = qa_chain.invoke({"query": query})
        return result.get("result", "I couldn't find a relevant answer in your diary.")
    except Exception as e:
        print(f"Error analyzing diary: {e}")
        return "Something went wrong. Please try again later."

# Initialize the knowledge base on server start
update_knowledge_base()

# API Endpoint to save diary entries
@app.route('/save_diary', methods=['POST'])
def save_diary():
    data = request.json
    entry = data.get('entry', '').strip()

    if entry:
        save_diary_entry(DIARY_FILE_PATH, entry)
        update_knowledge_base()  # Update the knowledge base with the new entry
        return jsonify({"status": "success"}), 200
    return jsonify({"status": "error", "message": "No entry provided."}), 400

# API Endpoint to analyze diary
@app.route('/analyze_diary', methods=['POST'])
def analyze_diary_endpoint():
    query = request.json.get('query', '').strip()

    if query:
        response = analyze_diary(query)
        return jsonify({"answer": response}), 200
    return jsonify({"status": "error", "message": "No query provided."}), 400

# API Endpoint for prompt-based queries using Ollama model
@app.route('/prompt_query', methods=['POST'])
def prompt_query():
    data = request.json
    prompt = data.get('prompt', '').strip()

    if prompt:
        response = process_prompt_with_model(prompt)
```

```
        return jsonify({"response": response}), 200
    return jsonify({"status": "error", "message": "No prompt provided."}), 400

# Run the Flask app
if __name__ == '__main__':
    app.run(debug=True)
```

App.CSS (CSS FILE)

```
/* General Styling */
body {
  margin: 0;
  font-family: 'Roboto', sans-serif;
  background-color: #121212; /* Dark background for a modern look */
  color: #e0e0e0; /* Soft white text for readability */
}

/* App Container */
.app {
  display: flex;
  flex-direction: column;
  height: 100vh;
}

/* Header */
.header {
  background-color: #1f1f1f; /* Darker header for separation */
  color: #00bcd4; /* Neon cyan text for contrast */
  padding: 20px;
  text-align: center;
  font-size: 1.8rem;
  font-weight: bold;
  letter-spacing: 1.5px;
  text-transform: uppercase;
}

/* Sidebar */
.sidebar {
  width: 60px; /* Collapsed width */
  background-color: #1f1f1f;
  color: #e0e0e0;
  position: fixed;
```

```
  height: 100%;
  display: flex;
  flex-direction: column;
  justify-content: center;
  transition: width 0.3s ease, background-color 0.3s ease;
}

.sidebar.open {
  width: 250px; /* Expanded width */
}

.sidebar ul {
  list-style: none;
  padding: 0;
}

.sidebar li {
  display: flex;
  align-items: center;
  padding: 15px;
  cursor: pointer;
  transition: background-color 0.2s ease;
}

.sidebar li:hover,
.sidebar li.active {
  background-color: #333333; /* Slightly lighter hover state */
}

.icon {
  font-size: 1.5rem;
  margin-right: 10px;
  color: #00bcd4; /* Neon cyan icons */
  transition: color 0.3s ease;
}

.collapsed {
  margin-right: 0;
}

.menu-text {
  font-size: 1.1rem;
  font-weight: bold;
  letter-spacing: 0.8px;
  color: #e0e0e0;
  transition: opacity 0.3s ease;
}
```

```
/* Content */
.content {
  margin-left: 60px; /* Adjusts dynamically */
  padding: 20px;
  transition: margin-left 0.3s ease;
}

.sidebar.open + .content {
  margin-left: 250px; /* Pushes content when sidebar expands */
}

/* Section Titles */
.section-title {
  font-size: 2rem;
  margin-bottom: 10px;
  color: #00e676; /* Neon green for section headers */
  font-weight: bold;
  text-transform: uppercase;
}

.section-subtitle {
  font-size: 1.2rem;
  font-weight: 600;
  color: #757575; /* Softer grey for subtitles */
  margin-bottom: 20px;
  text-transform: uppercase;
}

/* Input Fields */
.diary-input,
.query-input {
  width: 100%;
  padding: 15px;
  margin-bottom: 15px;
  font-size: 1.1rem;
  color: #ffffff;
  background-color: #1e1e1e; /* Dark input box */
  border: 1px solid #333333;
  border-radius: 8px;
  box-sizing: border-box;
}

.diary-input::placeholder,
.query-input::placeholder {
  color: #9e9e9e; /* Placeholder text */
}
```

```
.diary-input:focus,
.query-input:focus {
  outline: none;
  border-color: #00bcd4; /* Highlight on focus */
  box-shadow: 0 0 5px #00bcd4;
}

/* Buttons */
.submit-button,
.ask-button {
  padding: 12px 25px;
  font-size: 1.1rem;
  font-weight: bold;
  color: #ffffff;
  background-color: #00bcd4; /* Neon cyan buttons */
  border: none;
  border-radius: 8px;
  cursor: pointer;
  transition: background-color 0.3s ease, transform 0.2s ease;
}

.submit-button:hover,
.ask-button:hover {
  background-color: #00e676; /* Neon green hover state */
  transform: scale(1.05); /* Slight button pop effect */
}

.submit-button:active,
.ask-button:active {
  transform: scale(1); /* Reset on click */
}

/* Response Box */
.response-box {
  margin-top: 20px;
  padding: 20px;
  background-color: #1e1e1e;
  border-radius: 8px;
  border: 1px solid #333333;
  color: #00e676; /* Neon green for text responses */
  font-size: 1rem;
  line-height: 1.5;
  font-family: 'Courier New', monospace; /* Digital text feel */
}

/* Responsive Design */
```

```
@media (max-width: 768px) {
  .content {
    margin-left: 60px; /* Always collapsed on smaller screens */
  }

  .sidebar {
    width: 60px;
  }

  .sidebar.open {
    width: 200px;
  }

  .section-title {
    font-size: 1.5rem;
  }

  .submit-button,
  .ask-button {
    font-size: 1rem;
  }
}
```

CHAPTER 7

SCREEN SHOTS

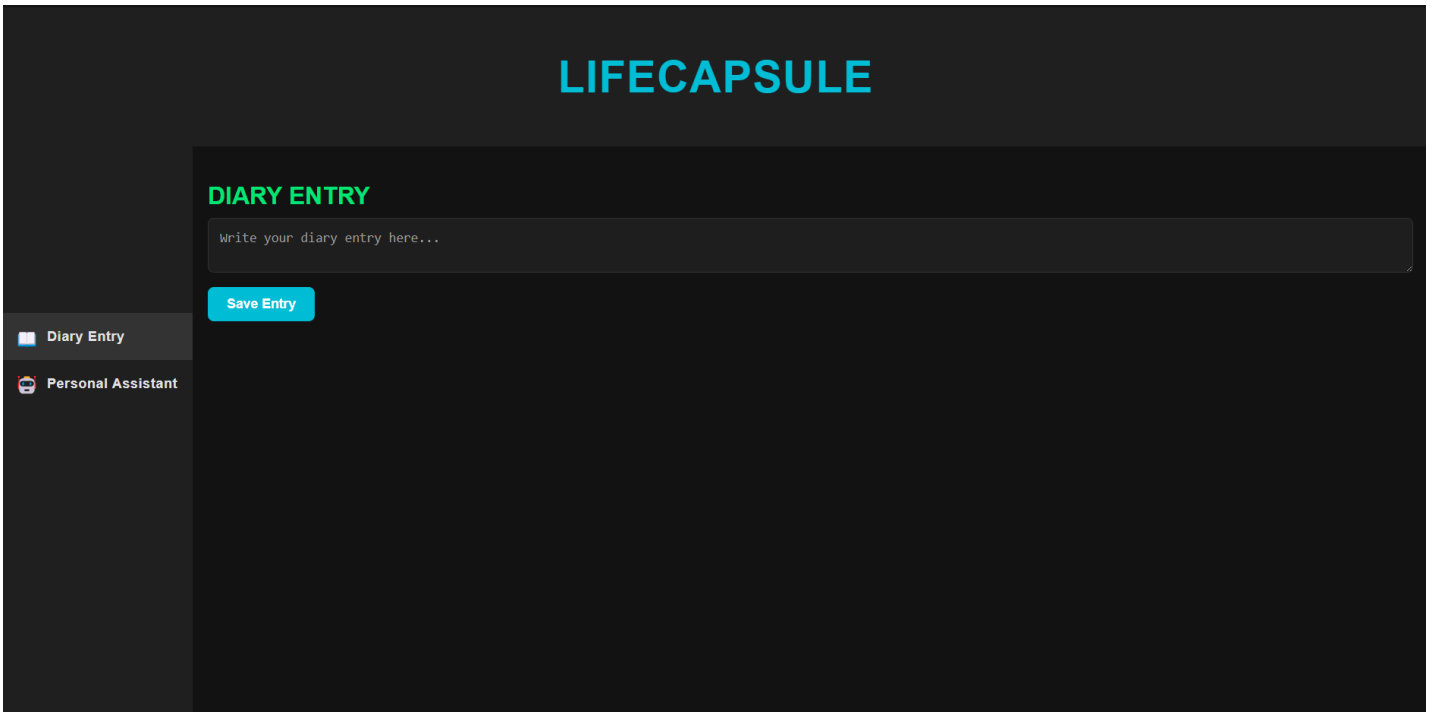


Fig. 7.1. Diary Entry Page

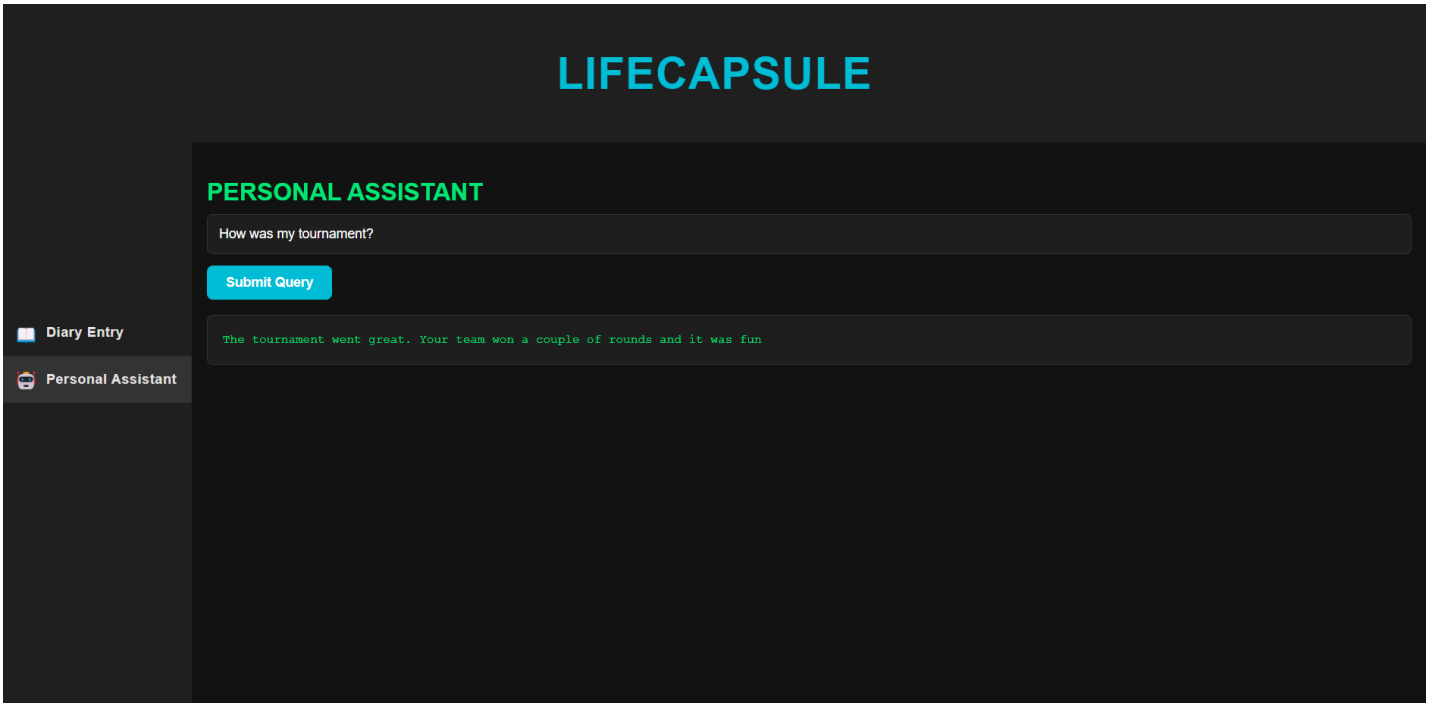


Fig. 7.2. Personal Assistant Page

CHAPTER 8

CONCLUSION

LifeCapsule represents a significant advancement in personal journaling applications by leveraging the power of AI to provide users with a meaningful and interactive way to manage their memories. Unlike conventional journaling tools, LifeCapsule integrates natural language processing models such as Llama 3.2 with LangChain and Chroma to deliver accurate and context-aware memory retrieval. This innovation enables users to relive past experiences, analyze recurring patterns, and draw valuable insights, all while ensuring that their data remains private and secure through local processing. By focusing on user-centric design and advanced technology, LifeCapsule has the potential to redefine how people document and reflect on their lives.

One of the key strengths of LifeCapsule lies in its ability to promote mental well-being through self-reflection. By offering personalized insights and summarizing key moments from users' lives, the application helps users gain a deeper understanding of their emotions, behaviors, and personal growth. This not only fosters a stronger connection with their memories but also encourages habits like gratitude and mindfulness. Moreover, the application's efficient architecture ensures fast retrieval and seamless user interaction, making it a practical and reliable tool for everyday use.

Looking ahead, LifeCapsule holds immense potential for future enhancements. Integrating features like sentiment analysis, mood tracking, and adaptive learning algorithms could make the application even more tailored to individual users' needs. Furthermore, incorporating optional cloud backup and multi-platform synchronization would broaden its usability for a global audience. By continuously evolving and incorporating emerging technologies, LifeCapsule can become an indispensable tool for personal memory management and self-improvement, catering to users in the fast-paced digital age.

REFERENCES

1. J. Hoffmann, S. Borgeaud, A. Mensch, et al., "Training Compute-Optimal Large Language Models," DeepMind, 2022.
2. A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention is All You Need," in *Proc. 31st Int. Conf. Neural Information Processing Systems*, 2017, pp. 5998-6008.
3. S. Borgeaud, A. Mensch, J. Hoffmann, et al., "Improving Language Models by Retrieving from Trillions of Tokens," arXiv preprint, 2021.
4. H. Kaplan, et al., "Scaling Laws for Neural Language Models," arXiv preprint, 2020.
5. Y. Du, Y. Huang, A. M. Dai, et al., "GLaM: Efficient Scaling of Language Models with Mixture-of-Experts," arXiv preprint, 2021.
6. A. Radford, et al., "Improving Language Understanding by Generative Pre-Training," OpenAI, 2018.
7. D. Fedus, B. Zoph, and N. Shazeer, "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity," arXiv preprint, 2021.
8. T. Kwiatkowski, et al., "Natural Questions: A Benchmark for Question Answering Research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453-466, 2019.
9. P. Kudo and J. Richardson, "SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing," in *Proc. 2018 Conf. Empirical Methods in Natural Language Processing*, 2018, pp. 66-71.
10. N. Goyal, et al., "Efficient Large Scale Language Modeling with Mixtures of Experts," arXiv preprint, 2021.
11. K. R. Sowmia, S. Poonkuzhali, "Artificial Neural Networks for Employability Prediction," Journal of Environmental Protection and Ecology, vol. 24, no. 2, pp. 671–686, 2023.
12. A. D. Ekawati, "Predictive Analytics in Employee Churn: A Systematic Literature Review," *Journal of Management Information and Decision Sciences*, vol. 22, no. 4, pp. 387, 2019.
13. F. Suleman, "The Employability Skills of Higher Education Graduates: Insights into Conceptual Frameworks and Methodological Options," *High Educ*, vol. 76, pp. 263, 2018.