AI-Powered College Chatbot

Project Based Learning (PBL) Report

For

Cantilever Internship

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

BY
Name of the students
G.Sleeva Kumar
K.Akshith
K.Srikar

HT.No:
23R11A05A4
23R11A0B6
23R11A05B7

Under the guidance of
Adgaonker Shashank

Department of Computer Science and Engineering

Accredited by NBA

GEETHANJALI COLLEGE OF ENGINEEERING AND TECHNOLOGY
(UGC AUTONOMOUS)

(Affiliated to J.N.T.U.H ,Approved by AICTE , New Delhi)

CHEERYAL(V), KEESARA(M), MEDCHAL DISTRICT, TS-501301

MAY- 2025

TABLE OF CONTENTS:

1.Introduction

1.1 About the project

In today's fast-paced academic environments, students, parents, and visitors often seek quick and accurate information about college facilities, departments, events, and processes. Traditional methods like navigating complex websites or waiting for administrative responses can be time-consuming and frustrating.

To address this need, we developed the "AI-Powered College Chatbot", an intelligent and user-friendly web-based assistant designed to provide instant answers to college-related queries. The chatbot leverages Natural Language Processing (NLP) and Machine Learning (ML) to understand user input and respond accordingly, creating an engaging and efficient communication experience.

The system is built using Python Flask as the backend framework and HTML/CSS with JavaScript for the frontend interface. At its core, the chatbot is trained on a curated dataset of frequently asked questions, structured in an intents.json file. These intents cover a wide range of topics—from admission procedures, hostel facilities, and placement stats to faculty details, fests, and infrastructure.

User queries are processed through a TfidfVectorizer and classified using a Logistic Regression model. Once the intent is identified, the chatbot dynamically returns a relevant response, which is elegantly displayed using a real-time chat interface. The interface is styled for a dark theme, with smooth user experience enhancements like typing effects and auto-scrolling.

This project showcases a practical application of Machine Learning, NLP, and Web Development, and serves as an effective solution for automating student helpdesks and improving information accessibility.

1.2 Project outcomes and objectives

🔹 Project Objectives

Streamline College Information Access
Develop a chatbot-based web application that allows students and visitors to get instant answers to common college-related queries without manual intervention.

Integrate Machine Learning with Web Technologies
Use a trained intent classification model to accurately detect user queries and serve context-aware responses within a Flask-powered web framework.

Implement NLP-Driven Intent Recognition
Utilize Natural Language Processing techniques, such as TfidfVectorizer and Logistic Regression, to analyze user input and determine intent with precision.

Provide Dynamic and Responsive Interaction
Design an interactive chat UI using HTML, CSS, and JavaScript to ensure a smooth, user-friendly experience with real-time response updates and effects.

Enable Modular Expansion
Structure the chatbot with a scalable and flexible architecture that can easily incorporate new intents, responses, or integrate with college databases in the future.
.

🔹 Expected Outcomes

✅ A fully functional chatbot web app built using Flask and HTML/CSS/JavaScript

✅ Real-time interaction with users via a visually rich and responsive chat interface

✅ Accurate response generation using trained NLP models and structured intent data

✅ Instant access to information on admissions, fees, departments, events, and more

✅ A reusable framework for expanding chatbot features or deploying across institutions

2. System Design

🎓 **AI-Powered College Chatbot System Overview**
The system uses a **client-server architecture**, where students interact with a chatbot interface, and the backend processes their questions using a local AI model (e.g., **Gemma 2B** via **Ollama**) to provide helpful and relevant answers.

🔧 **Main Components:**

💭 **User Interface (Frontend)**
- **Built with HTML, CSS, and JavaScript**
- Presents a friendly **chat interface** to students
- Accepts natural language queries (e.g., "What are the library hours?" or "How can I apply for a scholarship?")
- Sends queries to the backend using **AJAX** or **WebSocket** (for real-time interaction)
- Displays chatbot responses dynamically in the chat window

Web Server (Flask Backend):

- Developed using Python Flask
- Receives input data, formats it into a prompt
- Sends the prompt to the Ollama API using requests module

🧠 **AI Model (Gemma 2B via Ollama)**
- Local language model running on an **Ollama server**
- Processes college-related queries like:
- Course requirements
- Admission deadlines
- Club activities
- Faculty office hours
- Generates relevant and natural responses
- Can be fine-tuned or prompted with **college-specific FAQs** or documents for higher accuracy

📦 **Optional Enhancements**
- **Admin dashboard** for uploading/update FAQs and college data
- **User authentication** (student login) for personalized responses
- **Knowledge base integration** (PDFs, course catalogs, etc.)
- **Multilingual support** for international students
- **Event/calendar integration** with Google Calendar or internal systems

Workflow Diagram

User (Browser)

↓

[Chat Interface (HTML/JS)] → User submits a question (e.g., “When is the next exam?”)

↓

[Flask App] → Formats user input into a natural language prompt

↓

[Ollama API] → Processes prompt using Gemma 2B to generate college-specific response

↓

[Flask App] → Receives AI-generated reply

↓

[JSON Response / Jinja2 Template] → Dynamically displays chatbot answer on the web page

2.2 Modules

◆ 1. User Interface Module

- Provides a responsive, chat-style web interface for user interaction
- Accepts user queries via an input box and send button
- Built using HTML, CSS, and JavaScript
- Features a modern dark theme with typing animation and scroll behavior

◆ 2. Intent Recognition Module

- Receives and processes input text from the frontend
- Uses a trained Logistic Regression model to classify intent
- Converts input to vector form using TfidfVectorizer
- Identifies the matching tag from intents.json

◆ 3. Response Generation Module

- Fetches an appropriate response based on the predicted intent
- Randomly selects a reply from a list of predefined answers
- Ensures responses are conversational and informative
- Returns a default fallback message for unrecognized inputs

◆ 4. Backend Communication Module

- Developed using Python Flask as the backend framework
- Receives JSON requests from the frontend through a POST route (/get_response)
- Calls the chatbot logic to generate a response
- Sends the response back to the frontend in JSON format

◆ 5. Frontend Rendering Module

- Dynamically renders both user and bot messages inside a chat box
- Uses JavaScript to simulate real-time typing animations
- Styles messages with CSS for clarity and responsiveness
- Displays an error message if the backend response fails

🔷 6. Model Training & Update Module

- Responsible for training the chatbot’s intent classification model
- Uses scikit-learn’s TfidfVectorizer and Logistic Regression for efficient learning
- Parses intents.json to extract patterns and tags for supervised training
- Saves the trained model and vectorizer as .pkl files for reuse
- Can be rerun anytime new intents or training data are added
- Ensures the chatbot stays up-to-date with evolving user needs and topics

2.3 🔧 Backend Design

The backend of the College Chatbot is developed using Python Flask. It is structured to be modular, efficient, and seamlessly integrated with a trained machine learning model for real-time intent recognition and response generation.

🔹 Key Components

Flask Web Server

- Acts as the communication layer between the frontend chat interface and backend logic
- Listens for POST requests from the chat UI (via /get_response route)

Routes:
- '/' → Serves the main chatbot interface (index.html)
- '/get_response' → Accepts JSON input and returns a chatbot reply

Message Processing Logic
- Receives user input via request.json
- Passes the message to the intent classifier for prediction
- Matches predicted tag to the correct response from intents.json

Intent Classification Engine
- Loads pretrained Logistic Regression model and TfidfVectorizer (via Pickle)
- Converts raw user text into TF-IDF feature vectors
- Predicts intent tag based on learned patterns
- Retrieves and returns a relevant response or fallback message

Error Handling
- Catches and manages backend errors such as:
- Empty or malformed input
- Model loading failure
- Unexpected prediction errors
- Returns a user-friendly error response to the frontend

Template Rendering (Optional)
- Uses render_template to serve the main HTML page
- Ensures consistent structure and centralized asset management (CSS, JS)
- Frontend dynamically updates via JavaScript; no full page reloads required

🔧 **Tech Used:**

🧠 **Machine Learning & NLP**

- **scikit-learn** – for training a Logistic Regression model
- **TfidfVectorizer** – to convert user input into numerical features for intent classification

🌐 **Backend**

- **Flask** – lightweight Python web framework to handle routing and server logic
- **Pickle** – for saving and loading trained ML models and vectorizers
- **JSON** – to manage intent-response mappings via intents.json

💻 **Frontend**

- **HTML/CSS** – for structured layout and sleek dark-mode UI design
- **JavaScript** – for dynamic user interaction, real-time chat functionality, and API calls
- **SVG Icons** – to enhance the visual appeal of the chat interface

🔧 **Backend Code: app.py**

```
from flask import Flask, render_template, request, jsonify
import chatbot  # your chatbot logic in chatbot.py


app = Flask(__name__)


@app.route("/")
def home():
    return render_template("index.html")


@app.route("/get_response", methods=["POST"])
def get_response():
    user_msg = request.json.get("message")
    bot_reply = chatbot.get_response(user_msg)  # call your chatbot function
    return jsonify({"response": bot_reply})


if __name__ == "__main__":
    app.run(debug=True)
```

🔧 **Core Chatbot Logic: chatbot.py**

```
import json
import random
import pickle
import os

# Define paths relative to this script's location
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
INTENTS_PATH = os.path.join(BASE_DIR, 'data', 'intents.json')
MODEL_PATH = os.path.join(BASE_DIR, 'models', 'intent_classifier.pkl')
VECTORIZER_PATH = os.path.join(BASE_DIR, 'models', 'vectorizer.pkl')
# Load the intents JSON
with open(INTENTS_PATH, 'r') as file:
    intents = json.load(file)

# Load trained model and vectorizer
with open(MODEL_PATH, 'rb') as f:
    model = pickle.load(f)

with open(VECTORIZER_PATH, 'rb') as f:
    vectorizer = pickle.load(f)

def get_response(user_input):
    """
    Predict intent from user_input and return a random response.
    """
    # Transform user input using the loaded vectorizer
    user_input_vect = vectorizer.transform([user_input])

    # Predict the intent tag
    intent_tag = model.predict(user_input_vect)[0]

    # Find the intent responses
    for intent in intents['intents']:
        if intent['tag'] == intent_tag:
            return random.choice(intent['responses'])
    # Default response if no intent matched
    return "Sorry, I didn't understand that."
     # Optional: If you want a CLI chatbot for quick testing
    if __name__ == "__main__":
    print("College Chatbot is running! Type 'quit' to exit.")
    while True:
        user_input = input("You: ")
        if user_input.lower() == 'quit':
            print("Chatbot: Goodbye!")
            break
        response = get_response(user_input)
        print(f"Chatbot: {response}")
```

3. Implementation

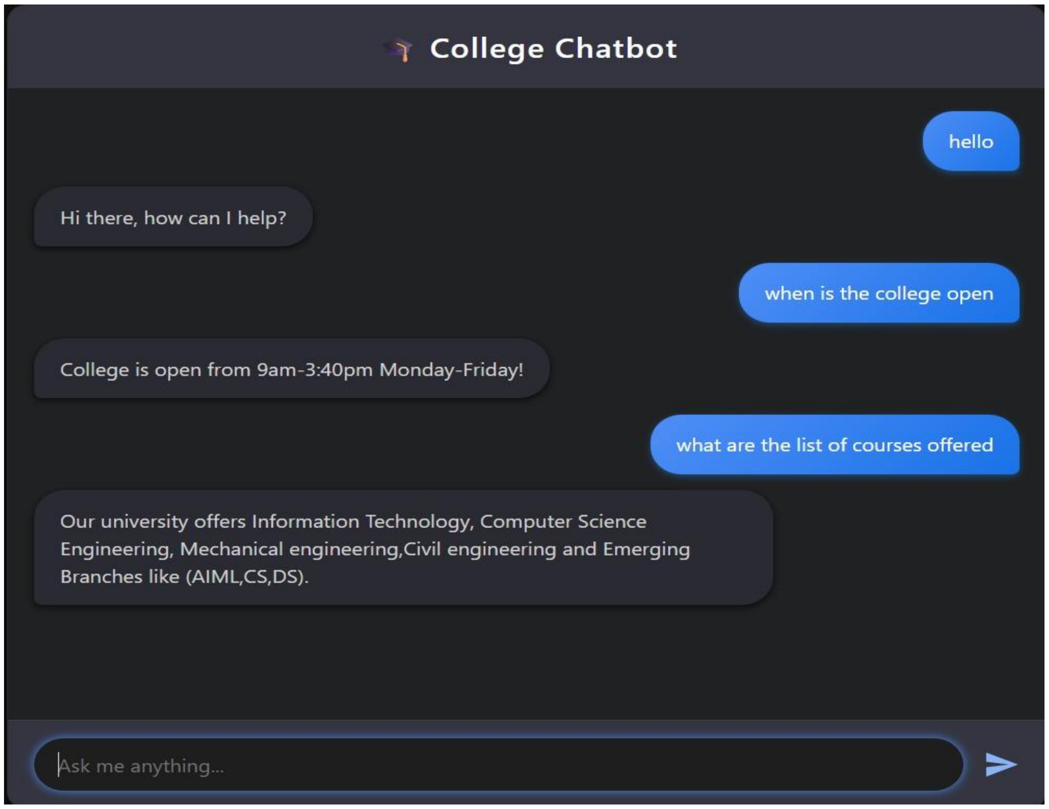
◆ 1. User Input Module

Implementation:
The user interface of the chatbot was developed using a modern HTML/CSS layout designed to resemble a chat window. Instead of using a traditional form, the user types their query into a text input field at the bottom of the chat interface.

Components Used:
- `<input>` – for capturing user messages
- `<button>` – for sending the message
- `<div>` containers – for rendering user and bot messages in the chat window

Interaction Flow:
- User enters a query like “What is the college timing?”
- On clicking the send button or pressing Enter, the query is sent via JavaScript `fetch()` to the Flask backend.
- The backend predicts the intent and returns a relevant response, which is then dynamically displayed in the chat box.

Validation:
- The input field is trimmed and checked for non-empty values before sending.
- Empty or whitespace-only messages are blocked to ensure meaningful input.



hello

Hi there, how can I help?

when is the college open

College is open from 9am-3:40pm Monday-Friday!

what are the list of courses offered

Our university offers Information Technology, Computer Science Engineering, Mechanical engineering,Civil engineering and Emerging Branches like (AIML,CS,DS).

🔹 2. Intent Detection Module:

Implementation:
When a user submits a message through the chat interface, it is sent to the Flask backend via a POST request using JavaScript's fetch() API.

Function:
The message is passed to the get_response() function in chatbot.py, where it is transformed into a vector using the preloaded TfidfVectorizer.

```
user_input_vect = vectorizer.transform([user_input])
```

This vectorized input is then used by the trained LogisticRegression model to predict the intent tag associated with the message.

```
intent_tag = model.predict(user_input_vect)[0]
```

Response Retrieval:
Once the intent is identified, the corresponding responses from intents.json are retrieved, and one is randomly selected to return to the user.
Example:
If a user asks, *"What are the college timings?"*, the model matches it to the hours tag and returns:
"College is open from 9am-3:40pm Monday-Friday!"

3. Backend Integration Module:
Implementation:
The backend is powered by a Flask application defined in app.py. It listens for user requests and routes them to appropriate handler functions.
Key Components:
- Flask handles routing (@app.route decorators).
- chatbot.py is imported and used for NLP-based intent detection.
- JSON is used for structured response formatting.

```
@app.route("/get_response", methods=["POST"])
def get_response():
   user_msg = request.json.get("message")
   bot_reply = chatbot.get_response(user_msg)
   return jsonify({"response": bot_reply})
```

Error Handling:
- Basic error handling is in place to ensure fallback messages are returned if no intent is matched.
- The system defaults to:

"Sorry, I didn't understand that."
when the input doesn't match any predefined intent.

Model Used:

- LogisticRegression from scikit-learn
- TfidfVectorizer for feature extraction

Configuration:

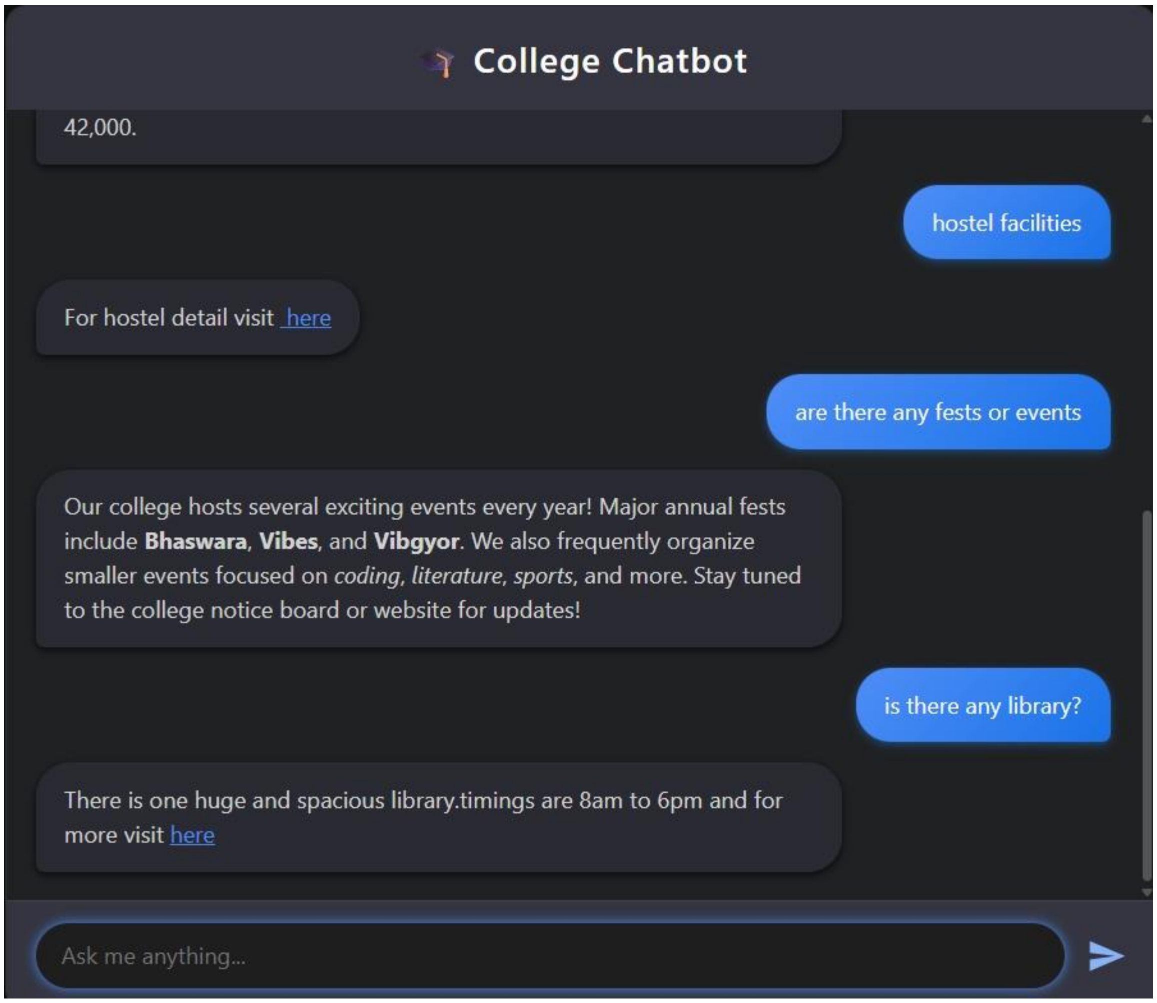
- The trained model and vectorizer are preloaded from .pkl files at app startup for fast response.
- No external API is required, making the system lightweight and fully local.

◆ **4. Response Handling Module**

Implementation:

Once the Flask backend receives the user's message from the frontend, it processes the input using the trained NLP model and returns a corresponding response. The JavaScript code in the frontend listens for this response, parses the returned JSON, and displays it dynamically in the chat box.

Sample Output:



College Chatbot

42,000.

hostel facilities

For hostel detail visit here

are there any fests or events

Our college hosts several exciting events every year! Major annual fests include **Bhaswara**, **Vibes**, and **Vibgyor**. We also frequently organize smaller events focused on *coding*, *literature*, *sports*, and more. Stay tuned to the college notice board or website for updates!

is there any library?

There is one huge and spacious library.timings are 8am to 6pm and for more visit here

🔹 5. Output Display Module

Implementation:
The chatbot uses Flask's Jinja2 for rendering the initial HTML structure, and JavaScript DOM manipulation for dynamically updating the chat conversation in real-time after each user input.
Unlike traditional form submissions, this chatbot uses asynchronous fetch requests to get predictions and display them directly on the same page without reloading.

UI Behavior:

- ✅ **If a valid prediction is returned:**
  The response is formatted as a styled chat bubble on the left (bot side), displayed in the .chat-box container.
- ⚠️ **If an error occurs:**
  A red-colored message (styled using the .bot-message class with optional enhancements) is shown to inform the user something went wrong (e.g., API failure or internal issue).

3.2 Code:

app.py

```
from flask import Flask, render_template, request, jsonify
import chatbot  # your chatbot logic in chatbot.py

app = Flask(__name__)

@app.route("/")
def home():
    return render_template("index.html")

@app.route("/get_response", methods=["POST"])
def get_response():
    user_msg = request.json.get("message")
    bot_reply = chatbot.get_response(user_msg)  # call your chatbot function
    return jsonify({"response": bot_reply})

if __name__ == "__main__":
    app.run(debug=True)
```

Chatbox.py

```
import json
import random
import pickle
import os

# Define paths relative to this script's location
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
INTENTS_PATH = os.path.join(BASE_DIR, 'data', 'intents.json')
MODEL_PATH = os.path.join(BASE_DIR, 'models', 'intent_classifier.pkl')
VECTORIZER_PATH = os.path.join(BASE_DIR, 'models', 'vectorizer.pkl')

# Load the intents JSON
with open(INTENTS_PATH, 'r') as file:
    intents = json.load(file)

# Load trained model and vectorizer
with open(MODEL_PATH, 'rb') as f:
    model = pickle.load(f)
```

```
with open(VECTORIZER_PATH, 'rb') as f:
    vectorizer = pickle.load(f)

def get_response(user_input):
    """
    Predict intent from user_input and return a random response.
    """
    # Transform user input using the loaded vectorizer
    user_input_vect = vectorizer.transform([user_input])

    # Predict the intent tag
    intent_tag = model.predict(user_input_vect)[0]

    # Find the intent responses
    for intent in intents['intents']:
        if intent['tag'] == intent_tag:
            return random.choice(intent['responses'])

    # Default response if no intent matched
    return "Sorry, I didn't understand that."

# Optional: If you want a CLI chatbot for quick testing
if __name__ == "__main__":
    print("College Chatbot is running! Type 'quit' to exit.")
    while True:
        user_input = input("You: ")
        if user_input.lower() == 'quit':
            print("Chatbot: Goodbye!")
            break
        response = get_response(user_input)
        print(f"Chatbot: {response}")
```

Train.py

```
import json
import random
import pickle
import os
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

# Load the intents JSON
with open("data/intents.json") as file:
    intents = json.load(file)

# Prepare data
inputs = []
labels = []

for intent in intents["intents"]:
    for pattern in intent["patterns"]:
        inputs.append(pattern)
        labels.append(intent["tag"])

# Vectorize input data
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(inputs)
y = labels

# Train the model
model = LogisticRegression()
model.fit(X, y)

# Save the model and vectorizer
os.makedirs("models", exist_ok=True)
with open("models/intent_classifier.pkl", "wb") as f:
    pickle.dump(model, f)

with open("models/vectorizer.pkl", "wb") as f:
    pickle.dump(vectorizer, f)

print("✅ Model trained and saved.")
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>College Chatbot</title>
<style>
/* Your existing styles here... */
* {
box-sizing: border-box;
margin: 0;
padding: 0;
}

html, body {
height: 100%;
font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
background-color: #121212;
display: flex;
justify-content: center;
align-items: center;
color: #e0e0e0;
}

.chat-container {
background-color: #202123;
width: 800px;
height: 700px;
border-radius: 15px;
box-shadow: 0 10px 30px rgba(0,0,0,0.7);
display: flex;
flex-direction: column;
overflow: hidden;
border: 1px solid #333;
}

.chat-header {
background-color: #343541;
padding: 20px;
color: white;
font-size: 24px;
font-weight: 600;
text-align: center;
flex-shrink: 0;
user-select: none;
letter-spacing: 1px;
}
```

```
.chat-box {
flex-grow: 1;
padding: 20px;
overflow-y: auto;
background-color: #202123;
display: flex;
flex-direction: column;
gap: 14px;
scrollbar-width: thin;
scrollbar-color: #555 transparent;
}

.chat-box::-webkit-scrollbar {
width: 8px;
}
.chat-box::-webkit-scrollbar-track {
background: transparent;
}
.chat-box::-webkit-scrollbar-thumb {
background-color: #555;
border-radius: 4px;
}

.message {
max-width: 75%;
padding: 14px 20px;
border-radius: 24px;
line-height: 1.5;
word-wrap: break-word;
font-size: 16px;
white-space: pre-wrap;
}

.user-message {
background: linear-gradient(135deg, #4f8ef7, #1a73e8);
color: white;
align-self: flex-end;
border-bottom-right-radius: 6px;
box-shadow: 0 2px 6px rgba(30, 136, 229, 0.4);
}

.bot-message {
background-color: #2a2b32;
color: #d3d3d3;
align-self: flex-start;
border-bottom-left-radius: 6px;
box-shadow: 0 2px 6px rgba(0,0,0,0.7);
font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
white-space: pre-wrap;
}
```

```
.chat-input-area {
display: flex;
padding: 15px 20px;
background-color: #343541;
border-top: 1px solid #444;
flex-shrink: 0;
}

.chat-input {
flex: 1;
padding: 12px 18px;
border: none;
border-radius: 24px;
font-size: 16px;
background-color: #121212;
color: #e0e0e0;
outline: none;
box-shadow: inset 0 0 5px #000;
transition: box-shadow 0.3s ease;
font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

.chat-input:focus {
box-shadow: 0 0 8px #4f8ef7;
background-color: #1e1e1e;
}

.send-btn {
background-color: transparent;
border: none;
margin-left: 15px;
cursor: pointer;
color: #8ab4f8;
font-size: 26px;
display: flex;
align-items: center;
justify-content: center;
transition: color 0.3s ease;
user-select: none;
}

.send-btn:hover {
color: #aecbfa;
}

.send-btn:active {
color: #669df6;
}
```

```
@keyframes blink {
0%, 100% { opacity: 1; }
50% { opacity: 0; }
}
.cursor {
display: inline-block;
background-color: #8ab4f8;
width: 8px;
margin-left: 3px;
animation: blink 1s step-start infinite;
border-radius: 2px;
vertical-align: bottom;
}
.bot-message a {
color: #4f8ef7;
text-decoration: underline;
cursor: pointer;
}

.bot-message a:hover {
color: #82b1ff;
text-decoration: none;
}
</style>
</head>
<body>
<div class="chat-container">
<div class="chat-header">🎓 College Chatbot</div>
<div class="chat-box" id="chat-box"></div>
<div class="chat-input-area">
<input type="text" id="user-input" class="chat-input" placeholder="Ask me anything..."
autocomplete="off" />
<button class="send-btn" aria-label="Send message" onclick="sendMessage()">
<svg xmlns="http://www.w3.org/2000/svg" height="28" width="28" viewBox="0 0 24
24" fill="currentColor">
<path d="M2.01 21L23 12 2.01 3 2 10l15 2-15 2z"/>
</svg>
</button>
</div>
</div>

<script>
function typeHTML(element, html, delay = 30) {
element.innerHTML = "";
const tempDiv = document.createElement("div");
tempDiv.innerHTML = html;
return new Promise((resolve) => {
function typeNode(node, parent, done) {
if (node.nodeType === Node.TEXT_NODE) {
let text = node.textContent;
let i = 0;
function typeChar() {
if (i < text.length) {
parent.appendChild(document.createTextNode(text.charAt(i)));
i++;
setTimeout(typeChar, delay);
} else {
done();
}
}
typeChar();
} else if (node.nodeType === Node.ELEMENT_NODE) {
const el = document.createElement(node.nodeName);
for (let attr of node.attributes) {
el.setAttribute(attr.name, attr.value);
}
parent.appendChild(el);

let children = Array.from(node.childNodes);
function typeChildren(idx = 0) {
if (idx < children.length) {
typeNode(children[idx], el, () => typeChildren(idx + 1));
} else {
done();
}
}
typeChildren();
} else {
done();
}
}

let nodes = Array.from(tempDiv.childNodes);
function typeAll(idx = 0) {
if (idx < nodes.length) {
typeNode(nodes[idx], element, () => typeAll(idx + 1));
} else {
resolve();
}
}
typeAll();
});
}
```

```
function sendMessage() {
const inputField = document.getElementById("user-input");
const message = inputField.value.trim();
if (!message) return;

const chatBox = document.getElementById("chat-box");

// Add user message
const userDiv = document.createElement("div");
userDiv.className = "message user-message";
userDiv.textContent = message;
chatBox.appendChild(userDiv);

inputField.value = "";
chatBox.scrollTop = chatBox.scrollHeight;

// Call your Flask backend
fetch("/get_response", {
method: "POST",
body: JSON.stringify({ message }),
headers: {
"Content-Type": "application/json",
},
})
.then((res) => res.json())
.then(async (data) => {
const botDiv = document.createElement("div");
botDiv.className = "message bot-message";
chatBox.appendChild(botDiv);
chatBox.scrollTop = chatBox.scrollHeight;
await typeHTML(botDiv, data.response, 30);
chatBox.scrollTop = chatBox.scrollHeight;
})
.catch(() => {
const errorDiv = document.createElement("div");
errorDiv.className = "message bot-message";
errorDiv.textContent = "❌ Error: Unable to get a response. Try again later.";
chatBox.appendChild(errorDiv);
chatBox.scrollTop = chatBox.scrollHeight;
});
document.getElementById("user-input").addEventListener("keypress", function (e) {
if (e.key === "Enter") {
sendMessage();
}
});
</script>
</body>
</html>
```

3.3 Data

Intents.json

```
{
"intents":[
{
"tag": "greeting",
"patterns": [
"Hi",
"Is anyone there?",
"Hello",
"Good day",
"What's up",
"heyy",
"whatsup",
"??? ??? ??"
],
"responses": [
"Hello!",
"Hi there, how can I help?",
"Good to see you again!"
],
"context_set": ""
},{
"tag": "status",
"patterns": [
"How are you",
"how are ya",
"how are you doing",
"how do you do",
"how r u"
],
"responses": [
"I'm doing great, thanks for asking! How can I assist you today?",
"I'm good! How can I help you?",
"Feeling helpful as always 😄"
],
"context_set": ""
},
{
"tag": "goodbye",
"patterns": [
"cya",
"see you",
"bye bye",
"See you later",
"Goodbye",
"I am Leaving",
"Bye",
"Have a Good day",
"talk to you later",
"ttyl",
"i got to go",
"gtg"
],
"responses": [
"Sad to see you go :(",
"Talk to you later",
"Goodbye!",
"Come back soon"
],
"context_set": ""
},
{
"tag": "creator",
"patterns": [
"what is the name of your developers",
"what is the name of your creators",
"what is the name of the developers",
"what is the name of the creators",
"who created you",
"your developers",
"your creators",
"who are your developers",
"developers",
"you are made by",
"you are made by whom",
"who created you",
"who create you",
"creators",
"who made you",
"who designed you"
],
"responses": [
"I was created by Akshith"
],
"context_set": ""
},
{
"tag": "name",
"patterns": [
"your name",
"do you have a name",
"what are you called",
"what is your name",
"what should I call you",
"whats your name?"
],
"responses": [
"You can call me Jarvis.",
"I'm Jarvis"
],
"context_set": ""
},
{
"tag": "identity",
"patterns": [
"what are you",
"who are you",
"who is this",
"what am i chatting to",
"who am i talking to"
],
"responses": [
"I'm your AI-powered college assistant chatbot, here to help you with college-
related questions."
],
"context_set": ""
},
{
"tag": "hours",
"patterns": [
"timing of college",
"what is college timing",
"working days",
"when are you guys open",
"what are your hours",
"hours of operation",
"when is the college open",
"college timing",
"what about college timing",
"is college open on saturday",
"tell something about college timing",
"what is the college hours",
"when should i come to college",
"when should i attend college",
"what is my college time",
"college timing",
"timing college"
],
"responses": [
"College is open from 9am-3:40pm Monday-Friday!"
],
"context_set": ""
},
```

{
"tag": "number",
"patterns": [
"more info",
"contact info",
"how to contact college",
"college telephone number",
"college number",
"What is your contact no",
"Contact number?",
"how to call you",
"College phone no?",
"how can i contact you",
"Can i get your phone number",
"how can i call you",
"phone number",
"phone no",
"call"
],
"responses": [
"You can contact Academic Section by calling : 9182058194"
],
"context_set": ""
},
{
"tag": "course",
"patterns": [
"list of courses",
"list of courses offered",
"list of courses offered in",
"what are the courses offered in your college?",
"courses?",
"courses offered",
"courses offered in geethanjali?",
"what are courses in UNI?",
"branches available in UNI?",
"can you tell me the courses available in UNI?",
"can you tell me the branches available in UNI?",
"computer engineering?",
"computer",
"Computer engineering?",
"it",
"IT",
"Information Technology",
"AI/Ml",
"Mechanical engineering",
"Chemical engineering",
"Civil engineering"
],

```
{
"tag": "fees",
"patterns": [
"information about fee",
"information on fee",
"tell me the fee",
"college fee",
"fee per semester",
"what is the fee of each semester",
"what is the fees of each year",
"what is fee",
"what is the fees",
"how much is the fees",
"fees for first year",
"fees",
"about the fees",
"tell me something about the fees",
"What is the fees of hostel",
"how much is the fees",
"hostel fees",
"fees for AC room",
"fees for Ac room for girls",
"fees for non-Ac room for girls",
"fees for Ac room for boys",
"fees for non-Ac room for boys"
],
"responses": [
"The Tuition fees for B.Tech is 1,20,000 Rs, M.Tech is 65,000 and MBA is 42,000."
],
"context_set": ""
},
{
"tag": "location",
"patterns": [
"where is the college located",
"college is located at",
"where is college",
"where is college located",
"address of college",
"how to reach college",
"college location",
"college address",
"wheres the college",
"how can I reach college",
"whats is the college address",
"what is the address of college",
"address",
"location"
],
```

"responses": [
"Cheeryal Village, Keesara Mandal, Hyderabad, Telangana 501301."
],
"context_set": ""
},
{
"tag": "hostel",
"patterns": [
"hostel facility",
"hostel servive",
"hostel location",
"hostel address",
"hostel facilities",
"hostel fees",
"Does college provide hostel",
"Is there any hostel",
"Where is hostel",
"do you guys have hostel",
"hostel",
"hostel capacity",
"what is the hostel fee",
"how to get in hostel",
"what is the hostel address",
"how far is hostel from college",
"hostel college distance",
"where is the hostel",
"how big is the hostel",
"distance between college and hostel",
"distance between hostel and college"
],
"responses": [
"For hostel detail visit <a target=\"_blank\" href=\"https://gcet.edu.in/hostel\"> here</a>"
],
"context_set": ""
},
{
"tag": "event",
"patterns": [
"events organised",
"list of events",
"list of events organised in college",
"list of events conducted in college",
"What events are conducted in college",
"Are there any event held at college",
"Events?",
"functions",
"what are the events",
"tell me about events",
"what about events"
],

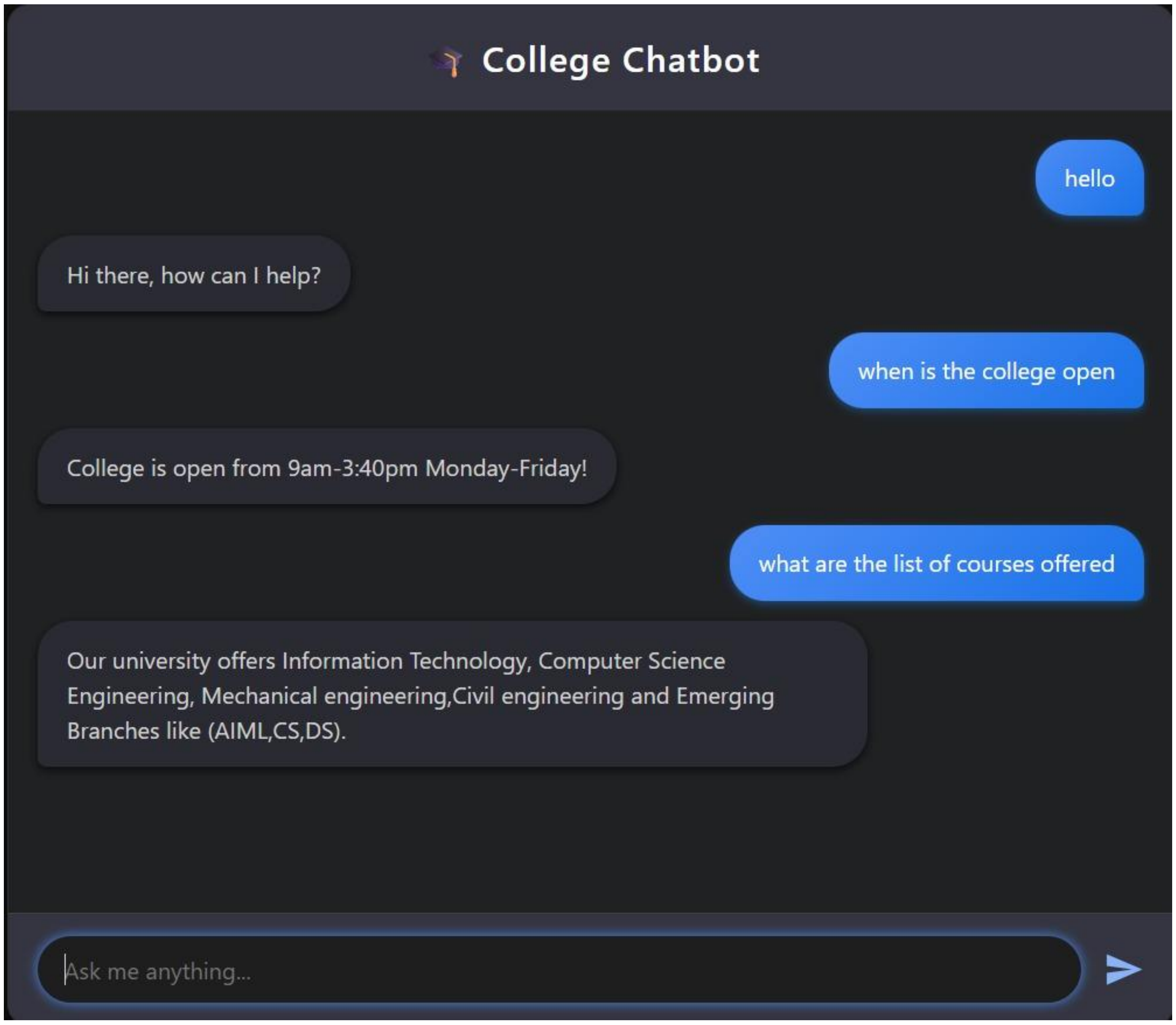
```
"responses": [
"Our college hosts several exciting events every year! Major annual events include
<strong>Bhaswara</strong>, <strong>Vibes</strong>, and <strong>Vibgyor</strong>. We also
frequently organize smaller events focused on <em>coding</em>, <em>literature</em>,
<em>sports</em>, and more. Stay tuned to the college notice board or website for updates!"
],
"context_set": ""
},
{
"tag": "fests",
"patterns": [
"fests organised",
"list of fests",
"list of fests organised in college",
"list of fests conducted in college",
"What fests are conducted in college",
"Are there any fests held at college",
"fests?",
"functions",
"what are the fests",
"tell me about fests",
"what about fests"
],
"responses": [
"Our college hosts several exciting events every year! Major annual fests include
<strong>Bhaswara</strong>, <strong>Vibes</strong>, and <strong>Vibgyor</strong>. We also
frequently organize smaller events focused on <em>coding</em>, <em>literature</em>,
<em>sports</em>, and more. Stay tuned to the college notice board or website for updates!"
],
"context_set": ""
},
{
"tag": "document",
"documents needed at the time of admission",
"documents needed during admission",
"documents required for admision",
"documents required at the time of admission",
"documents required during admission",
"What document are required for admission",
"Which document to bring for admission",
"documents",
"what documents do i need",
"what documents do I need for admission",
"documents needed"
],
"responses": [
"You can contact Academic Section by calling : 9182058194 for those details."
],
"context_set": ""
},
{
"tag": "block",
"patterns": [
"size of campus",
"building size",
"How many floors does college have",
"floors in college",
"floors in college",
"how tall is UNI's College of Engineering college building",
"floors"
],
"responses": [
"To know about campus visit <a target=\"_blank\"
href=\"https://gcet.edu.in/infrastucture\"> here</a>"
],
"context_set": ""
},
{
"Syllabus for IT",
"what is the Information Technology syllabus",
"syllabus",
"timetable",
"what is IT syllabus",
"syllabus",
"What is next lecture"
],
"responses": [
"To know about syllabus visit <a target=\"_blank\"
href=\"https://www.gctcportal.in/p/syllabus.html\"> here</a>"
],
"context_set": ""
},
{
"tag": "library",
"patterns": [
"is there any library",
"library facility",
"library facilities",
"do you have library",
"where can i get books",
"book facility",
"Where is library",
"Library",
"Library information",
"Library books information",
"Tell me about library",
"how many libraries"
],
"responses": [
"There is one huge and spacious library.timings are 8am to 6pm and for more visit <a
target=\"blank\" href=\"https://sites.google.com/view/gcetlibraryinf/Library-
Profile\">here</a>"
],
"context_set": ""
},
{
"tag": "infrastructure",
"patterns": [
"how is college infrastructure",
"infrastructure",
"college infrastructure",
"Tell me about college infrastructure."
],
"responses": [
"Our University has Excellent Infrastructure. Campus is clean. Good IT Labs With Good Speed
of Internet connection"
],
"context_set": ""
},
{
"tag": "canteen",
"patterns": [
"food facilities",
"canteen facilities",
"canteen facility",
"is there any canteen",
"Is there a cafetaria in college",
"Does college have canteen",
"Where is canteen",
"where is cafetaria",
"canteen",
"Food",
"Cafetaria"
],
"responses": [
"Our university has canteen with variety of food available. To take a look at it visit <a
target=\"blank\" href=\"https://gcet.edu.in/canteen\">here</a>"
],
"context_set": ""
},
{
"tag": "menu",
"patterns": [
"food menu",
"food in canteen",
"Whats there on menu",
"what is available in college canteen",
"what foods can we get in college canteen",
"food variety",
"What is there to eat?"
],
"responses": [
"we serve Franky, Puffs, Samosa, meals, fast food and many more on menu"
],
"context_set": ""
},
{
"tag": "placements",
"patterns": [
"What is college placement",
"Which companies visit in college",
"What is average package",
"companies visit",
"package",
"About placement",
"placements?",
"recruitment",
"companies"
],
"responses": [
"To know about placement visit <a target=\"_blank\"
href=\"https://gcet.edu.in/placements\">here</a> or <a target=\"_blank\"
href=\"https://www.gctcportal.in/p/placements.html\">here</a>"
],
"context_set": ""
},
{
"tag": "ithod",
"patterns": [
"Who is HOD",
"Where is HOD",
"it hod",
"name of it hod"
],
"responses": [
"All engineering departments have their own respective hod's"
],
"context_set": ""
},
{
"tag": "computerhod",
"patterns": [
"Who is computer HOD",
"Where is computer HOD",
"computer hod",
"name of computer hod"
],
"responses": [
"All engineering departments have only one hod XYZ who available on (PLACE NAME)"
],
"context_set": ""
},
{
"tag": "extchod",
"patterns": [
"Who is extc HOD",
"Where is extc HOD",
"extc hod",
"name of extc hod"
],
"responses": [
"Different school wise hod are different.So be more clear with your school or
department"
],
"context_set": ""
},
{
"tag": "principal",
"patterns": [
"what is the name of principal",
"what is the principal name",
"principal name",
"Who is college principal",
"Where is principal's office",
"principal",
"name of principal"
],
"responses": [
"Dr.Sagar is college principal",
"Dr.Sagar is college principal and if you need any help then call your branch hod
first.That is more appropriate"
],
"context_set": ""
},
{
"tag": "sem",
"patterns": [
"exam dates",
"exam schedule",
"When is semester exam",
"Semester exam timetable",
"sem",
"semester",
"exam",
"when is exam",
"exam timetable",
"exam dates",
"when is semester"
],
"responses": [
"Here is the Academic Calendar <a target=\"_blank\" href=\"https://www.gcetece.org/copy-of-
regulations-and-syllabus\">website</a> or <a target=\"_blank\"
href=\"https://gcet.edu.in/exam_announcements\">website</a>"
],
"context_set": ""
},
{
"tag": "admission",
"patterns": [
"what is the process of admission",
"what is the admission process",
"How to take admission in your college",
"What is the process for admission",
"admission",
"admission process"
],
"responses": [
"There are two ways 1.Through EAMCET exam 2.Through Management."
],
"context_set": ""
},
{
"tag": "scholarship",
"patterns": [
"scholarship",
"Is scholarship available",
"scholarship engineering",
"scholarship it",
"scholarship ce",
"scholarship mechanical",
"scholarship civil",
"scholarship chemical",
"scholarship for AI/ML",
"available scholarships",
"scholarship for computer engineering",
"scholarship for IT engineering",
"scholarship for mechanical engineering",
"scholarship for civil engineering",
"scholarship for chemical engineering",
"list of scholarship",
"comps scholarship",
"IT scholarship",
"mechanical scholarship",
"civil scholarship",
"chemical scholarship",
"automobile scholarship",
"first year scholarship",
"second year scholarship",
"third year scholarship",
"fourth year scholarship"
],
"responses": [
"Many government scholarships are supported by our university."
],
"context_set": ""
},
{
"tag": "facilities",
"patterns": [
"What facilities college provide",
"College facility",
"What are college facilities",
"facilities",
"facilities provided"
],
"responses": [
"Our university's Engineering department provides fully AC Lab with internet
connection, smart classroom, Auditorium, library,canteen,etc."
],
"context_set": ""
},
{
"tag": "college intake",
"patterns": [
"max number of students",
"number of seats per branch",
"number of seats in each branch",
"maximum number of seats",
"maximum students intake",
"What is college intake",
```

],
"responses": [
"For college intake details visit <a target=\"_blank\"
href=\"https://www.gctcportal.in/p/programmes-offered-by-college.html\">website</a>"
],
"context_set": ""
},
{
"tag": "uniform",
"patterns": [
"college dress code",
"college dresscode",
"what is the uniform",
"can we wear casuals",
"Does college have an uniform",
"Is there any uniform",
"uniform",
"what about uniform",
"do we have to wear uniform"
],
"responses": [
"There's no specific uniform policy. But some of the dresses are prohibited 1.Torn
jeans 2.Round neck Shirts/T-shirts for boys"
],
"context_set": ""
},
{
"tag": "committee",
"patterns": [
"what are the different committe in college",
"different committee in college",
"Are there any committee in college",
"Give me committee details",
"committee",
"how many committee are there in college"
],
"responses": [
"For college committee details visit <a target=\"_blank\"
href=\"https://gcet.edu.in/committees\">here</a>"
],
"context_set": ""
},
{
"tag": "random",
"patterns": [
"I love you",
"Will you marry me",
"Do you love me"
],

```
"responses": [
"I am not program for this, please ask appropriate query"
],
"context_set": ""
},
{
"tag": "swear",
"patterns": [
"fuck",
"bitch",
"shut up",
"hell",
"stupid",
"idiot",
"dumb ass",
"asshole",
"fucker"
],
"responses": [
"please use appropriate language",
"Maintaining decency would be appreciated"
],
"context_set": ""
},
{
"tag": "vacation",
"patterns": [
"holidays",
"when will semester starts",
"when will semester end",
"when is the holidays",
"list of holidays",
"Holiday in these year",
"holiday list",
"about vacations",
"about holidays",
"When is vacation",
"When is holidays",
"how long will be the vacation"
],
"responses": [
"Check your academic calender for holidays details. If you dont have it you can find it <a target=\"_blank\" href=\"https://gcet.edu.in/academic-calendar\">here</a>."
],
"context_set": ""
},
{
"tag": "sports",
"patterns": [
"sports and games",
"give sports details",
"sports infrastructure",
"sports facilities",
"information about sports",
"Sports activities",
"please provide sports and games information"
],
"responses": [
"Our university encourages all-round development of students and hence provides sports
facilities in the campus. For more details visit <a target=\"_blank\"
href=\"https://gcet.edu.in/sports\">here</a>"
],
"context_set": ""
},
{
"tag": "salutaion",
"patterns": [
"okk",
"okie",
"nice work",
"well done",
"good job",
"thanks for the help",
"Thank You",
"its ok",
"Thanks",
"Good work",
"k",
"ok",
"okay"
],
"responses": [
"I am glad I helped you",
"welcome, anything else i can assist you with?"
],
"context_set": ""
},
{
```

```
"tag": "task",
"patterns": [
"what can you do",
"what are the thing you can do",
"things you can do",
"what can u do for me",
"how u can help me",
"why i should use you"
],
"responses": [
"I can answer to low-intermediate questions regarding college",
"You can ask me questions regarding college, and i will try to answer them"
],
"context_set": ""
},
{
"tag": "ragging",
"patterns": [
"ragging",
"is ragging practice active in college",
"does college have any antiragging facility",
"is there any ragging cases",
"is ragging done here",
"ragging against",
"antiragging facility",
"ragging juniors",
"ragging history",
"ragging incidents"
],
"responses": [
"We are Proud to tell you that our college provides ragging free environment, and we have
strict rules against ragging."
],
"context_set": ""
},
{
"tag": "hod",
"patterns": [
"hod",
"hod name",
"who is the hod"
],
"responses": [
"HODs differ for each branch."
],
"context_set": ""
},
{
"tag": "labs",
"patterns": [
"how are the labs in college",
"are there any labs in college",
"labs?",
"computer labs?",
"electrical labs?"
],
"responses": [
"The labs are clean with excellent equipments, and also they are very spacious."
],
"context_set": ""
},{
"tag": "transport",
"patterns": [
"college transport",
"bus facility",
"bus timings",
"transport service",
"how to reach college",
"college buses"
],
"responses": [
"The college provides bus facilities from major points in the city. Bus timings and routes are
available here: <a href='https://gcet.edu.in/transport' target='_blank'>Transport info</a>."
],
"context_set": ""
}
]
}
```

3.4 Output



hello

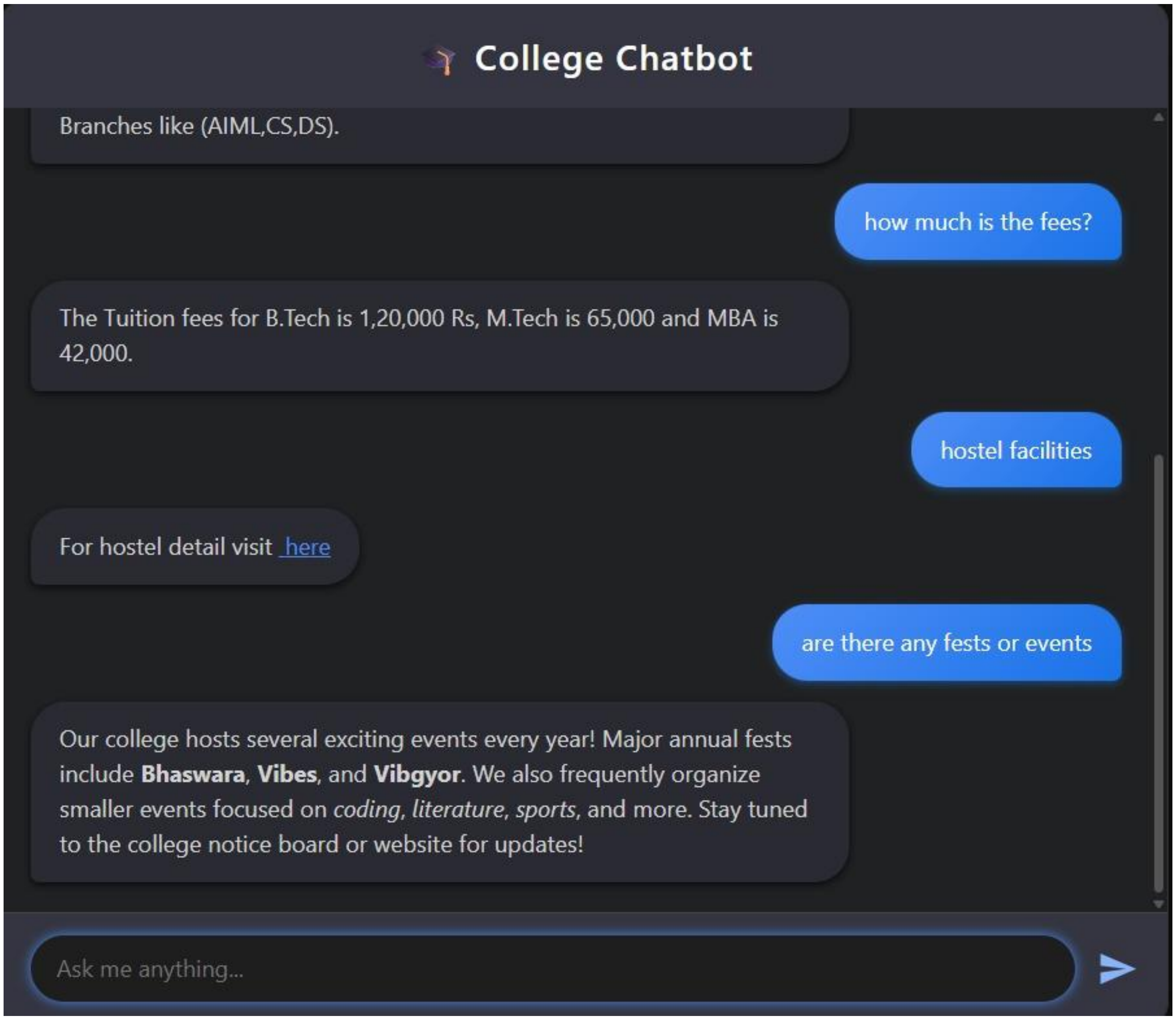
Hi there, how can I help?

when is the college open

College is open from 9am-3:40pm Monday-Friday!

what are the list of courses offered

Our university offers Information Technology, Computer Science Engineering, Mechanical engineering,Civil engineering and Emerging Branches like (AIML,CS,DS).



College Chatbot

Branches like (AIML,CS,DS).

how much is the fees?

The Tuition fees for B.Tech is 1,20,000 Rs, M.Tech is 65,000 and MBA is 42,000.

hostel facilities

For hostel detail visit here

are there any fests or events

Our college hosts several exciting events every year! Major annual fests include **Bhaswara**, **Vibes**, and **Vibgyor**. We also frequently organize smaller events focused on *coding*, *literature*, *sports*, and more. Stay tuned to the college notice board or website for updates!

4.Conclusion

The AI-Powered College Chatbot is a robust and intelligent system designed to streamline communication between users and institutional information resources. By leveraging machine learning and natural language processing techniques, the chatbot efficiently interprets user queries and delivers accurate, context-aware responses in real time.

The modular architecture—comprising a Flask-based backend, a trained intent classification model, and a dynamic JavaScript-driven frontend—ensures the system is scalable, maintainable, and adaptable to future enhancements. Whether it's answering questions about admissions, hostel facilities, campus life, or events, the chatbot demonstrates how modern AI solutions can simplify access to information and improve user experience.

Beyond its current implementation, the project lays the groundwork for potential integrations with voice input, multilingual support, real-time data from college databases, and even personalized user sessions. It serves as a practical example of how machine learning can be applied to real-world challenges in education and public service.

The simplicity of its training workflow and intent-based logic makes it easy to update as new student queries or college processes emerge. Its intuitive design ensures that even first-time users can navigate and interact with ease. With minimal infrastructure requirements and an open-source foundation, the chatbot is highly deployable in institutions of varying sizes and technical capacities.

Furthermore, the chatbot not only reduces the workload on administrative staff but also boosts the digital reputation of the institution by providing 24/7 support. It exemplifies the growing role of conversational AI in transforming how information is accessed and delivered in academic environments.

In summary, the College Chatbot offers a smart, scalable, and user-friendly approach to modern student engagement—paving the way for intelligent virtual assistants in the education sector.

5. References

. https://github.com/keras-team/keras

https://flask.palletsprojects.com/

https://web.stanford.edu/~jurafsky/slp3/

****