CHATBOT FOR LICET ADMISSION ENQUIRY

A PROJECT REPORT

*Submitted by*

**ALAN JUDITH A**

**CYRIL MARTIN S**

**KANISHKUMAR K**

**SATYAVARSSHENI RA V**

*in partial fulfillment for the award of the degree of*

**BACHELOR OF ENGINEERING**

*in*

# COMPUTER SCIENCE AND ENGINEERING

**LOYOLA-ICAM COLLEGE OF ENGINEERING AND TECHNOLOGY**

# CHENNAI – 600034

**ANNA UNIVERSITY: 600025**

# MAY 2023

**ANNA UNIVERSITY CHENNAI: 600025 BONAFIDE CERTIFICATE**

Certified that this project report **“Chatbot for LICET Admission Enquiry”** is the bonafide work of **ALAN JUDITH A (311120104301), CYRIL MARTIN S (311120104014), KANISHKUMAR K (311120104027)** and **SATYAVARSSHENI RA V (311120104053)** who carried out the project work under my supervision.

# SIGNATURE SIGNATURE

Dr. K. Gopalakrishnan Ms. Daya Mary Mathew

# HEAD OF THE DEPARTMENT SUPERVISOR

Department of Computer Science Assistant Professor

and Engineering Department of Computer

Loyola-ICAM College of Science and Engineering

Engineering and Technology and Technology

Loyola Campus, Loyola Campus,

Nungambakkam, Nungambakkam,

Chennai-600034 Chennai-600034

Submitted for the project viva voce held on ……………………………….

# INTERNAL EXAMINER EXTERNAL EXAMINER

**ACKNOWLEDGEMENT**

Firstly, we are deeply obliged to the almighty for this opportunity and for leading us to the successful completion of the project. The experience has been very fruitful.

We express our sincere gratitude to the beloved Director **Rev Dr Sebastian Soosai J**, the Dean of Students **Dr D Caleb Chanthi Raj** for their continuous support and Encouragement.

We are deeply obliged to the Principal **Dr L Antony Michael Raj** for his inspiring guidance and invaluable advice during the project work. We also express our sincere thanks to the Head of the Department and Project Coordinator **Dr K Gopalakrishnan**, the teaching and non-teaching faculty of our department for their advice, support and guidance.

We extend our gratitude to our project guide **Ms Daya Mary Mathew** for providing valuable insights and resources. Without her supervision and constant help, this project would not have been possible.

Last but not the least, we are extremely grateful to our family and friends, who have been a constant source of support during the preparation of this project work.

**ABSTRACT**

This abstract presents a versatile chatbot designed to streamline LICET's (Loyola-ICAM College of Engineering and Technology) admission inquiries. The chatbot offers text and voice-based interactions, enabling users to engage through their preferred medium.

It provides a virtual campus tour, empowering prospective students to explore the campus remotely. Using the cosine similarity algorithm in natural language processing, the chatbot offers a cutoff calculator that analyzes and predicts admission scores. Additionally, it suggests courses based on students' interests and achievements, promoting informed decision-making.

Integration with SMS enables instant messaging for quick and convenient communication. The chatbot facilitates access to downloadable brochures and placement insights, empowering users with comprehensive information. It showcases a gallery of campus facilities, hosts details about hostels, sports, and offered courses. Student reviews further enrich the experience, while contact details ensure easy access to admission-related inquiries.

This chatbot combines user-friendly features with advanced technology, providing a personalized and efficient admission inquiry process for prospective students.

**Keywords:** *Cosine Similarity, NLP, Speech recognition, Chatbot, NLTK, Virtual campus tour***.**

# TABLE OF CONTENT

| **CHAPTER NO.**    **1** | **TITLE ABSTRACT**    **TABLE OF FIGURES**  **INTRODUCTION** | **PAGE NUMBER**  4  7 |
| --- | --- | --- |
|  | 1.1 Problem definition | 8 |
|  | 1.2 Existing applications | 8 |
|  | 1.3 Need for the system | 10 |
| **2** | **LITERATURE SURVEY**  2.1 Indonesian Chatbot of University Admission  2.2 NEU-chatbot: Chatbot for Admission  2.3 Voice-based University Information Chatbot  2.4 Implementation of a Chatbot System using AI  2.5 An Intelligent Chatbot for College Admission. | 12 |

1. **SYSTEM ANALYSIS**
   1. [System requirements](#_heading=h.30j0zll)  15
      1. Requirements 15
      2. [Packages](#_heading=h.1fob9te) 17
2. **SYSTEM DESIGN**
   1. [Object oriented design](#_heading=h.3znysh7) 20
   2. Structural diagram 21
   3. Behavioral diagram 21
   4. [Use case diagram 21](#_heading=h.2et92p0)
   5. Architectural diagram 22
3. **SYSTEM IMPLEMENTATION**
   1. Algorithm and proposed techniques 24
      1. Cosine similarity24
      2. Cosine similarity algorithm steps 26
      3. Flowchart28
4. **PERFORMANCE METRICS** 29
5. **CONCLUSION AND FUTURE WORK** 31
   1. [Conclusion](#_heading=h.17dp8vu) 31
   2. [Future work](#_heading=h.3rdcrjn) 31

**APPENDIX**

**APPENDIX A: CODE** 32

**APPENDIX B: SNAPSHOTS** 56

**REFERENCE** 60

# TABLE OF FIGURES

**FIGURE NO. FIGURE TITLE PAGE NO.**

4.1 Use Case diagram 22

4.2 Architecture diagram 23

5.1 Cosine Similarity algorithm visualization 26

5.2 Cosine similarity formula 27

5.3. Vector representation 27

5.4. Flowchart of chatbot 28

6.1 Accuracy Bar Graph 29

B.1 Snapshot of the Chatbot User Interface 56

B.2 Snapshot of the Chat 56

B.3 Snapshot of the Speaking avatar in ABOUT section 57

B.4 Snapshot of the Outside Campus Tour 57

B.5 Snapshot of the Virtual Campus Tour (Inside Campus) 58

B.6 Snapshot of the Alumni Review 58

B.7 Snapshot of the CutOff Insights 59

B.8 Snapshot of the Gallery 59

# CHAPTER 1 INTRODUCTION

* 1. **PROBLEM DEFINITION**

The LICET Campus Chatbot project is an innovative initiative aimed at revolutionizing the admission inquiry process for LICET (Loyola-ICAM College of Engineering and Technology). This intelligent chatbot combines advanced technologies, such as natural language processing and cosine similarity algorithm, to provide seamless and personalized interactions for prospective students. By offering text and voice-based interactions, virtual campus tours, and a cutoff calculator, the chatbot ensures a user-friendly experience. The project also envisions future enhancements, including integration with other platforms, incorporation of additional features, and the utilization of machine learning techniques for improved performance. Through this project, LICET aims to enhance accessibility, efficiency, and user satisfaction in the admission inquiry process, empowering prospective students to make informed decisions about their academic journey.

# EXISTING APPLICATIONS

1. **Chatbot for admission**

The Chatbot is an application designed specifically for the admission process at the National Economics University This chatbot serves as a virtual assistant, providing information and support to prospective students throughout their admission journey. It offers features such as answering frequently asked questions, providing information on admission requirements, guiding students through the application process, and offering updates on important dates and deadlines. While the NEU-Chatbot offers valuable assistance to prospective students, there are certain areas where it can be further improved. One aspect that could be enhanced is the chatbot's ability to provide more personalized recommendations and guidance based on individual student profiles, interests, and academic achievements.

1. **University Admission Using a Question Answering System Based on Sequence-to-Sequence Model.**

The chatbot is designed to understand and respond to inquiries related to university admission, such as admission requirements, application procedures, scholarship opportunities, and program offerings. By using a sequence-to-sequence model, the chatbot can generate accurate and contextually relevant responses.

This project aims to streamline the admission process by providing a virtual assistant that can handle a wide range of admission-related questions and provide timely and accurate information to prospective students. By leveraging the power of deep learning and natural language processing, the chatbot offers an interactive and user-friendly experience for users seeking information about university admissions in Indonesia.

**c) Chatbot Enquiry System**

The College Enquiry Chat-Bot System is an interactive platform designed to assist students and visitors with their inquiries related to colleges. This chatbot system leverages natural language processing and artificial intelligence techniques to provide accurate and prompt responses to various college-related questions.

The system enables users to engage in text-based conversations with the chatbot, allowing them to inquire about admission procedures, course offerings, faculty information, campus facilities, extracurricular activities, and more. The chatbot utilizes advanced algorithms to understand and interpret user queries, providing relevant and helpful answers.

# NEED FOR THE SYSTEM

In today's fast-paced and digitally-driven world, educational institutions are constantly seeking innovative solutions to streamline their admission processes and enhance the overall experience for prospective students. In this context, the need for a versatile chatbot system specifically designed to address the admission inquiries of LICET (Loyola-ICAM College of Engineering and Technology) becomes evident.

The chatbot serves as a virtual assistant, offering both text and voice-based interactions to cater to the diverse preferences of users. This flexibility ensures that prospective students can engage with the chatbot through their preferred medium, providing a seamless and personalized experience right from the initial contact.

One of the key features of the chatbot is its virtual campus tour capability. This feature allows prospective students to explore the college campus remotely, giving them a glimpse of the infrastructure, facilities, and ambiance of LICET. This virtual tour serves as a valuable tool, especially for students who may be unable to visit the campus physically or those residing in distant locations.

Another significant aspect of the chatbot is its cutoff calculator, which utilizes the cosine similarity algorithm in natural language processing. This algorithm analyzes and predicts admission scores, assisting students in understanding their chances of securing admission based on their academic performance. By providing this predictive tool, the chatbot empowers students to make informed decisions regarding their application, saving time and effort in the process.

Furthermore, the chatbot goes beyond just providing cutoff predictions by offering course suggestions tailored to students' interests and achievements. By leveraging the available data, including students' preferences and past accomplishments, the chatbot generates personalized recommendations, guiding students towards courses that align with their aspirations and strengths. This personalized approach enhances the overall satisfaction and likelihood of success for prospective students.

The integration of SMS functionality within the chatbot enables instant messaging, facilitating quick and convenient communication. This feature ensures that students can receive timely responses to their queries and concerns, eliminating the need for prolonged waiting periods and enhancing overall engagement.

Moreover, the chatbot showcases a gallery of campus facilities, allowing students to virtually explore the hostels, sports facilities, laboratories, libraries, and other amenities available at LICET. This visual representation creates a sense of connection and familiarity, helping students envision themselves as part of the college community.

**CHAPTER 2 LITERATURE SURVEY**

1. **Indonesian Chatbot of University Admission Using a Question Answering System Based on Sequence-to-Sequence Model**

The proposed work focuses on developing a chatbot system for the university admission process in Indonesia. The authors utilize a question-answering (QA) system based on a sequence-to-sequence (Seq2Seq) model. The methodology involves collecting questions and answers from various sources such as university admission websites and admission officers. The collected data is then preprocessed and used to train the Seq2Seq model.

However, one of the potential challenges of the study is the limited scope of evaluation. The authors only assess the system's performance using a small set of predefined questions, which may not fully represent the diverse range of queries that users might have during the university admission process. Additionally, the chatbot system described in the study does not incorporate voice-based recognition, implying that interactions with the system are text-based rather than voice-based.

**II. NEU-chatbot: Chatbot for admission of National Economics University**

The proposed work revolves around the development and evaluation of a chatbot system called NEU-chatbot, specifically designed for the admission process at the National Economics University. The authors utilize the Microsoft Bot Framework for the development of the chatbot system and employ natural language processing (NLP) techniques to comprehend and provide responses to user queries.

The efficiency of the NEU-chatbot system lies in its ability to effectively handle text-based inputs and generate appropriate responses. However, it is important to note that the scope of the study is limited, as it primarily focuses on addressing queries related to the admission process and does not incorporate other functionalities. The NEU-chatbot system may not support voice-based recognition or cater to a broader range of topics beyond the admission process at the National Economics University.

**III. Voice based University Information Chatbot System**

The proposed work focuses on implementing a voice-based chatbot system for universities to provide information to students through voice commands. The authors utilize Natural Language Processing (NLP) algorithms to enable seamless conversations without interruptions. The system also manages a database specific to the university and is designed to provide information 24/7 based on user queries.

The efficiency of this voice-based university information chatbot system lies in its ability to improve the university services by reducing response times and enhancing accessibility. By allowing students to interact with the chatbot using voice commands, it offers a convenient and efficient means of obtaining information. This can potentially lead to quicker access to relevant information, better user experience, and overall improved efficiency in the university's information dissemination process. The published study can be found in the International Research Journal of Engineering and Technology (IRJET).

**IV. Implementation of a Chatbot System using AI and NLP**

The proposed work outlined in the article focuses on implementing a chatbot system using artificial intelligence (AI) and natural language processing (NLP) techniques. The chatbot system is designed to interact with users through a natural language interface, comprehend their requests, and provide appropriate information or perform actions accordingly.

The methodology employed in the article involves utilizing the Python programming language and integrating various APIs, such as Dialogflow and IBM Watson, to facilitate the chatbot's understanding and response capabilities. The effectiveness of the system was evaluated through testing with different user inputs.

The efficiency of the implemented chatbot system lies in its ability to effectively understand and respond to user queries in a conversational manner, thanks to the integration of AI and NLP algorithms. The article primarily covers the architecture and implementation of the chatbot system, providing insights into the technologies and techniques used to enable its functionality.

**V. An Intelligent chatbot for College Admission Counseling**

The proposed work described in the article focuses on the development of an intelligent chatbot for college admission counseling. The authors present a modular architecture for the chatbot, consisting of a natural language processing module, a knowledge base, and a user interface. The performance of the chatbot is also evaluated.

The methodology employed in the study involves the use of the Python programming language and various natural language processing libraries like NLTK and TextBlob. Additionally, web scraping techniques are utilized to gather data from college websites and admission brochures.

The chatbot system proves to be efficient in providing accurate and relevant information to students. It can alleviate the workload of admission counselors by handling routine queries and offering students quick and easy access to information about colleges and admission requirements. However, it's important to note that the chatbot may not provide personalized advice on college selection or application processes.

**SYSTEM ANALYSIS**

# SYSTEM REQUIREMENTS

The following specifications were those required by the system for the software’s successful implementation and functioning

# REQUIREMENTS SOFTWARE REQUIREMENTS

* PYTHON 3.11
* VISUAL STUDIO CODE
* COSINE SIMILARITY ALGORITHM
* THINGLINK, RENDERSTUFF, IMGBB
* TWILIO API

# HARDWARE REQUIREMENTS

* SERVER
* PROCESSOR
* GPU

# PYTHON

# The python programming language is a high-level, object-oriented, and interpreted programming language. Python is of best use when it comes to Rapid Application Development and this due to the high-level built-in data structures which are combined with dynamic typing and dynamic binding. This programming language has a simple and easy-to-learn syntax which emphasizes readability and reduces the overall cost of program maintenance. This high-level programming language can also be used as a scripting or glue language to connect the existing components. It supports a lot of modules and packages and encourages program modularity and code reusability. Python has an interpreter and an extensive standard library that are made available in source or binary form without charge for all major platforms, and it is an open-source language.

# VISUAL STUDIO CODE

The Visual Studio Code which is a [dual-licensed](https://en.wikipedia.org/wiki/Dual-licensed) [source-code editor](https://en.wikipedia.org/wiki/Source-code_editor) is made by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) for [Windows](https://en.wikipedia.org/wiki/Windows), [Linux](https://en.wikipedia.org/wiki/Linux), and [macOS](https://en.wikipedia.org/wiki/MacOS). Features of visual studio code include support for [debugging](https://en.wikipedia.org/wiki/Debugging), [syntax highlighting](https://en.wikipedia.org/wiki/Syntax_highlighting), [intelligent code](https://en.wikipedia.org/wiki/Intelligent_code_completion) [completion](https://en.wikipedia.org/wiki/Intelligent_code_completion), [snippets](https://en.wikipedia.org/wiki/Snippet_(programming)), [code refactoring](https://en.wikipedia.org/wiki/Code_refactoring), and embedded [Git](https://en.wikipedia.org/wiki/Git). It is very user-friendly and the users can install [extensions](https://en.wikipedia.org/wiki/Plug-in_(computing)) that add additional functionality.

# COSINE SIMILARITY ALGORITHM

Cosine similarity is a widely used algorithm for natural language processing tasks such as text classification, information retrieval, and recommendation systems. It is a simple yet effective way to measure the similarity between two pieces of text by computing the cosine of the angle between their vector representations in a high-dimensional space. It is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

**THINGLINK, RENDERSTUFF, IMGBB**

**1. ThingLink:** ThingLink is an interactive media platform that allows users to create engaging visual experiences by adding interactive elements to images, videos, and 360-degree content. With ThingLink, you can enhance static visuals by adding tags, text, audio, and video overlays that provide additional information or create interactive storytelling experiences. These interactive elements can be accessed by viewers through hotspots or clickable areas within the media. ThingLink is commonly used in education, marketing, journalism, and e-commerce to provide immersive and interactive content experiences.

**2. RenderStuff:** RenderStuff is a website and online community that focuses on providing resources and tutorials for 3D artists and visualizers. It offers a wide range of content related to computer-generated imagery (CGI), including tutorials, downloadable 3D models, textures, plugins, and other resources that assist artists in creating realistic renders and visual effects. RenderStuff aims to support artists at various skill levels by sharing knowledge, techniques, and industry insights to improve their 3D rendering skills and workflow efficiency.

**3. ImgBB:** ImgBB is a free image hosting and sharing platform that allows users to upload, store, and share images online. It provides a simple and straightforward interface for uploading images from your device or pasting image URLs. ImgBB generates multiple image links, including direct links, HTML thumbnails, and BBCode for easy integration into websites, forums, or social media platforms. Additionally, ImgBB offers features such as image compression and resizing, making it convenient for users who need to optimize their images for web use or reduce file sizes without compromising image quality.

# **TWILIO API**

Twilio is a cloud communication platform that provides a set of APIs (Application Programming Interfaces) for developers to integrate voice, messaging, and video capabilities into their applications. The Twilio API allows developers to programmatically send and receive SMS messages, make and receive phone calls, and handle various other communication-related tasks.

# PACKAGES

These are the packages that were downloaded and utilized in our project.

* + - * Flask==2.2.3
      * matplotlib==3.7.1
      * nltk==3.8.1
      * pandas==1.5.3
      * PyQt5==5.15.9
      * scikit-learn==1.2.2
      * SpeechRecognition==3.10.0
      * twilio==8.1.0

**Flask**

Flask is a popular web framework for building web applications using Python. It is known for its simplicity and ease of use. Flask is considered a "micro" framework because it provides only the bare essentials for web development, allowing developers to choose and integrate additional libraries and tools as needed.

**NLTK**

NLTK (Natural Language Toolkit) is a Python library that provides a set of tools and resources for working with human language data. It is widely used for tasks such as text processing, tokenization, stemming, tagging, parsing, and semantic reasoning.

**Pandas**

Pandas is a powerful open-source library for data manipulation and analysis in Python. It provides easy-to-use data structures, such as DataFrame and Series, along with a variety of functions to perform operations on the data.

**Pyqt5**

PyQt is a set of Python bindings for the Qt application framework. Qt is a popular cross-platform framework for developing graphical user interfaces (GUIs) and applications.

**Scikit-learn**

Scikit-learn, also known as sklearn, is a popular machine learning library for Python. It provides a wide range of algorithms and tools for data preprocessing, feature extraction, model selection, and evaluation.

**Twilio**

Twilio is a cloud communication platform that provides APIs (Application Programming Interfaces) for developers to integrate voice, messaging, and video functionalities into their applications. It allows you to send SMS messages, make phone calls, and handle other communication tasks programmatically.

# Matplotlib

Matplotlib is a data visualization and graphics toolkit which is cross- platform. Python and its numerical extension NumPy also have a charting application named al. As a corollary, it offers an open-source alternative to MATLAB. Matplotlib's APIs (Application Programming Interfaces) could also be used to embed charts in graphical user interfaces.

# CHAPTER 4 SYSTEM DESIGN

# OBJECT ORIENTED DESIGN

Identifying the objects in a system would be what OO (Object Oriented) analysis and design have always been about identifying their relationships. Generating a design that could be transformed into executables utilising object- oriented programming languages.

UML (Unified Modelling Language) is a standard language that uses, designs, constructing, and documenting software components. The Unified Modelling Language (UML) is an effective instrument enabling Object- Oriented Analysis and Design. The Object Management Group (OMG) created it, and in January 1997, the OMG proposed UML 1.0 as a specification draught. It started as a way to capture the behaviour of vast software and non-software systems and has since grown into such an international standard. UML is created to be process generic, indicating it could be used in a variety of circumstances. It can be used for a spectrum of uses. Business analysts, software architects, and developers are all using UML as a common language. It can be used to describe, specify, build, and document the system's business processes, including its structural and behavioural artefacts.

UML is a modelling language used to create software blueprints. Diagrams are categorized into two divisions, which are further divided into subcategories:

# Structural Diagrams

* + - **Behavioral Diagrams**

# STRUCTURAL DIAGRAMS

# The static aspect of the system is portrayed by the structural diagrams. These static aspects are the components of a diagram that describes the main structure and are thus stable. Classes, interfaces, objects, components, and nodes are often used to represent them.

Some of the structural diagrams are:

* + - Class diagram
    - Object diagram
    - Component diagram
    - Deployment diagram

# BEHAVIORAL DIAGRAMS

Behavioral diagrams illustrate a system's complex nature. The changing/moving parts of a system are usually known for the dynamic aspect. The following five types of behavioral diagrams are supported in UML:

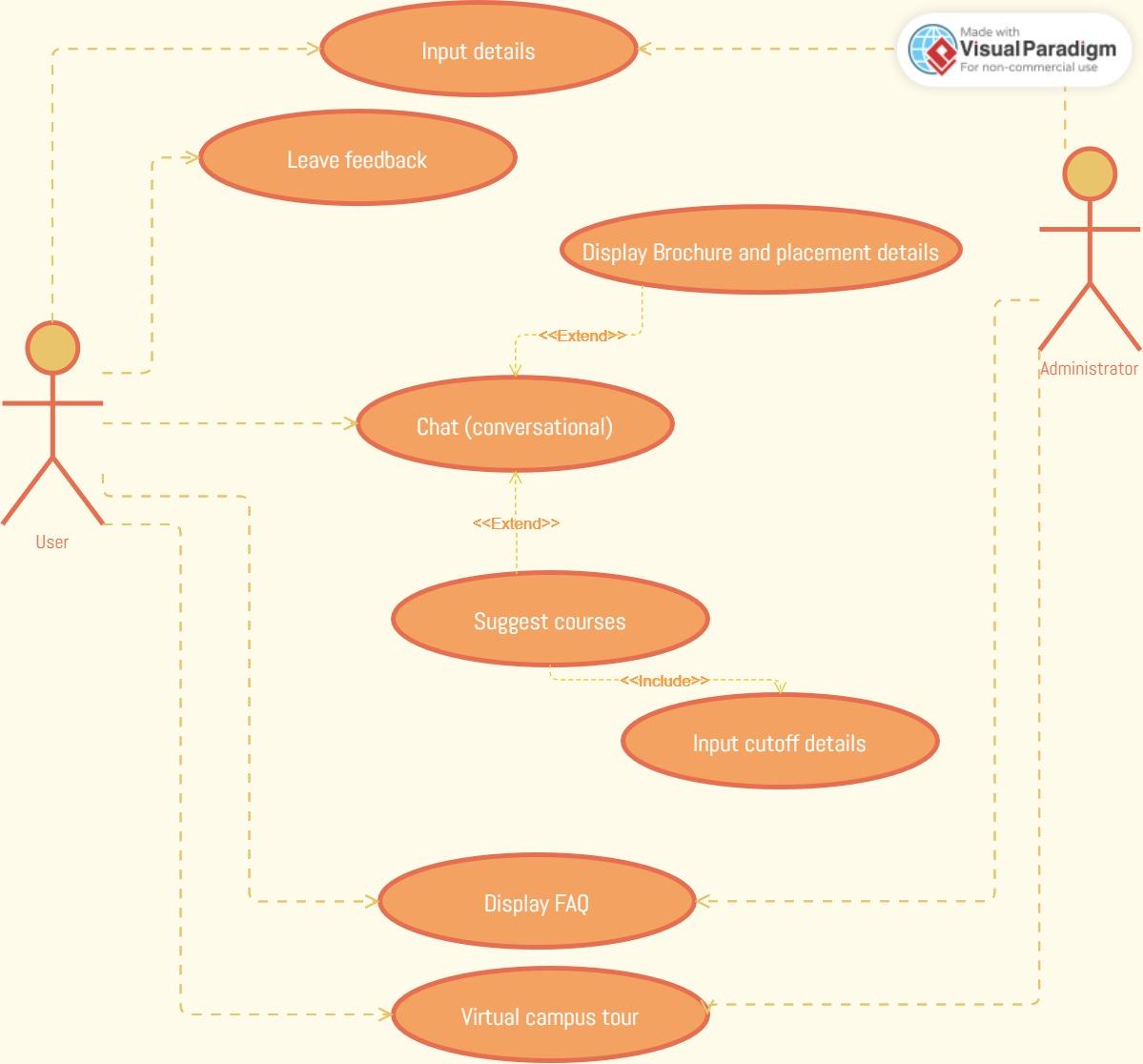
* + - Use case diagram
    - Sequence diagram
    - Collaboration diagram
    - Statechart diagram
    - Activity diagram

# USE CASE DIAGRAM

The use case diagrams show the system's behavior concerning to the deployment environment. It illustrates the proposed system's users. A use case is a system analysis methodology for finding, explaining, and monitoring programs needs.

A use case is a collection of conceivable sequences of interactions between systems and users in a specific environment, all of which are tied to a specific purpose. A use case is represented by an ellipse with the name of the use case. A stick figure with a name is used to represent an actor.

The use case diagram for the chatbot system incorporating cosine similarity for text and voice-based interactions involves several key components. The primary actor in this diagram is the user who interacts with the chatbot system. The user initiates the conversation by providing input in the form of text or voice. This input is then processed using NLP techniques to understand the user's intent and extract relevant information. The processed input is further used to calculate the cosine similarity between the user's input vector and the vectors representing the knowledge base.

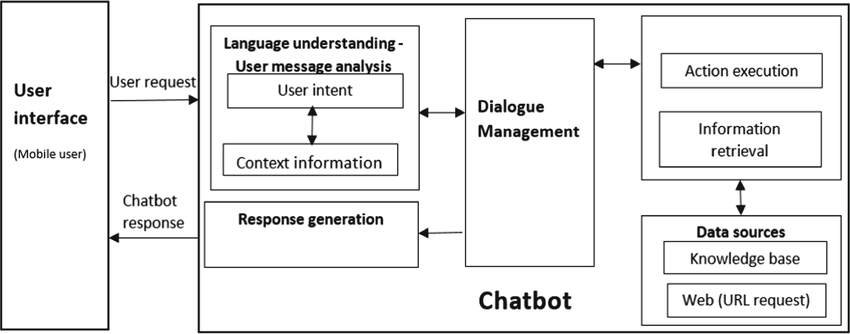


*Fig 4.1 Use Case diagram*

# ARCHITECTURE DIAGRAM

The architecture diagram for the chatbot system leveraging cosine similarity begins with the user providing input through text-based or voice-based interaction.

The input is then processed using Natural Language Processing (NLP) techniques to understand the user's intent and extract relevant information. Once the input is preprocessed and converted into a numerical representation, a knowledge base is established. The knowledge base consists of predefined responses or a collection of relevant information, where each entry is also transformed into vector representations.



*Fig 4.2 Architecture diagram*

# CHAPTER 5 SYSTEM IMPLEMENTATION

* 1. **ALGORITHMS AND PROPOSED TECHNIQUES**

# COSINE SIMILARITY ALGORITHM

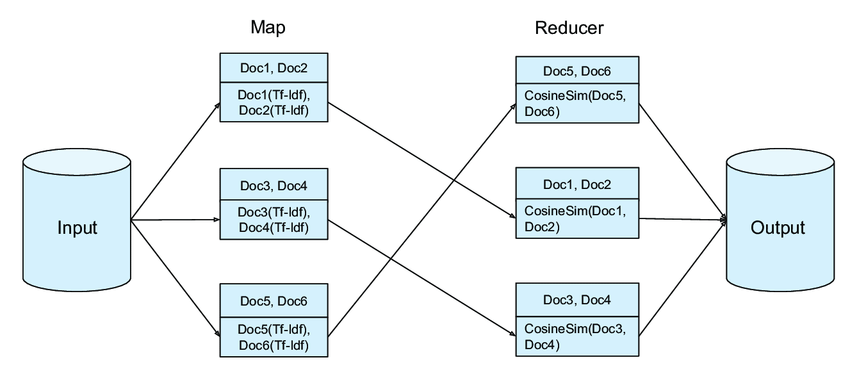
Cosine similarity is a mathematical measure used to determine the similarity between two vectors in a multi-dimensional space. It calculates the cosine of the angle between the vectors, which represents their similarity or dissimilarity.

In the context of natural language processing and information retrieval, cosine similarity is commonly employed to compare and measure the similarity between documents or text snippets. Each document or text snippet is represented as a vector, where each dimension represents a specific feature or term, and the value corresponds to the frequency or importance of that term in the document.

Cosine similarity has wide-ranging applications, including text classification, document clustering, recommendation systems, and information retrieval. It provides a useful and intuitive metric to compare and rank textual data based on their semantic similarity, aiding in tasks such as search engines, content recommendation, and question-answering systems.

There are several merits of using cosine similarity as a similarity metric:

* Intuitive and Interpretability: Cosine similarity provides an intuitive interpretation of similarity by measuring the angle between vectors. A value of 1 indicates perfect similarity, 0 denotes no similarity, and -1 represents complete dissimilarity. This makes it easy to understand and interpret the results.
* Scale-Invariant: Cosine similarity is scale-invariant, which means it is unaffected by the magnitude or length of the vectors being compared. This property is particularly advantageous when dealing with high-dimensional data or text documents of varying lengths. It allows for fair comparisons between documents regardless of their size.
* Effective for Text Comparison: Cosine similarity is widely used in natural language processing tasks, such as text classification, information retrieval, and document clustering. It works well with textual data represented as vectorized representations, such as bag-of-words or TF-IDF (Term Frequency-Inverse Document Frequency) vectors. It captures the semantic similarity between documents, enabling effective matching and retrieval of relevant information.
* Computationally Efficient: Computing cosine similarity is computationally efficient, especially for sparse high-dimensional data. It involves simple mathematical operations like dot products and vector magnitudes, which can be efficiently calculated. This makes it suitable for large-scale applications where real-time or near-real-time response is required.
* Robust to Noise and Irrelevant Features: Cosine similarity focuses on the orientation and direction of vectors rather than their magnitude. As a result, it is robust to noise and irrelevant features in the data. Irrelevant terms or features that may be present in the vectors do not significantly affect the similarity measurement.
* Widely Supported: Cosine similarity is a well-established and widely supported metric. It is implemented in various libraries and frameworks for machine learning, text mining, and information retrieval. This availability makes it easy to integrate and utilize in different applications and programming languages.



*Fig 5.1. Cosine similarity algorithm visualization*

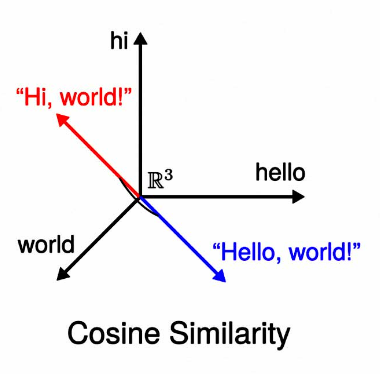
# COSINE SIMILARITY ALGORITHM STEPS

1. **Represent the Vectors:** Convert the data you want to compare into vector form. Each vector represents a document, sentence, or any set of features. The vectors can be represented as a set of numerical values or binary values.
2. **Normalize the Vectors:** Normalize the vectors to eliminate the influence of vector length on the cosine similarity. This step ensures that vectors with different lengths can still be compared accurately. Normalize each vector by dividing each component by its magnitude.
3. **Calculate the Dot Product:** Compute the dot product of the two normalized vectors. The dot product is the sum of the element-wise multiplication of corresponding elements in the vectors.
4. **Calculate Vector Magnitudes:** Calculate the magnitudes (lengths) of the two vectors. The magnitude of a vector can be calculated using the Euclidean distance formula or by taking the square root of the sum of the squared values of each element in the vector.
5. **Calculate Cosine Similarity:** Divide the dot product of the vectors by the product of their magnitudes. This division yields the cosine similarity value. The formula for cosine similarity is:
   1. cosine\_similarity = dot\_product / (magnitude\_vector1 \* magnitude\_vector2)



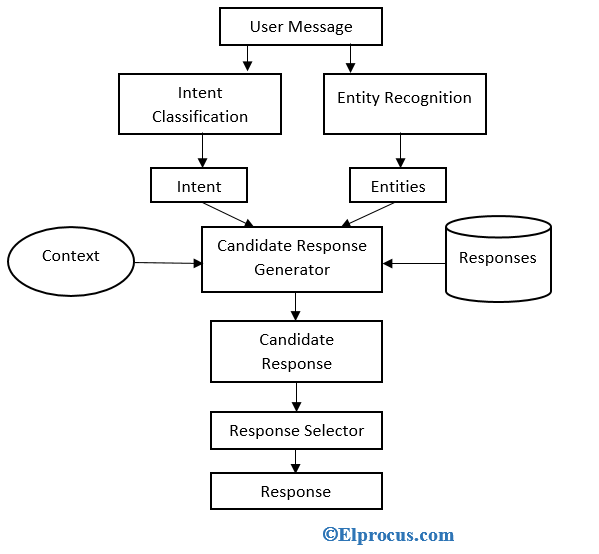
*Fig 5.2 Cosine similarity formula*

1. **Obtain the Similarity Score:** The resulting cosine similarity value ranges from -1 to 1. A cosine similarity of 1 indicates that the vectors are identical, 0 indicates no similarity, and -1 indicates complete dissimilarity.



*Fig 5.3. Vector representation*

# **F**LOWCHART

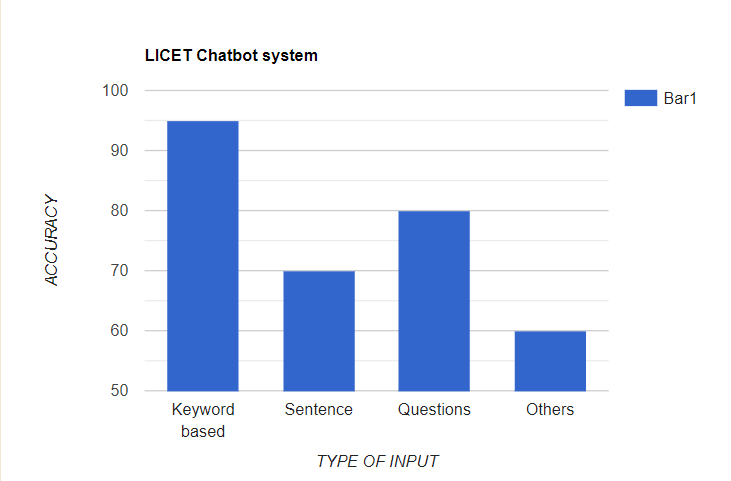


*Fig 5.4. flowchart of chatbot*

# CHAPTER 6 PERFORMANCE METRICS

The performance metrics of the LICET chatbot system provide valuable insights into its effectiveness and efficiency. Accuracy measures the correctness and relevance of responses, ensuring that users receive accurate information. Response time indicates the system's efficiency in providing prompt answers, enhancing the user experience.

The completion rate reflects the chatbot's ability to handle queries autonomously, reducing the need for human intervention. Additionally, the engagement level signifies the level of user interaction and involvement with the chatbot. These performance metrics collectively assess the chatbot's ability to deliver accurate, timely, and comprehensive information, resulting in an improved admission inquiry process for prospective students.



*Fig 6.1 Accuracy Bar Graph*

The bar graph displays the performance of the LICET chatbot system across different types of inputs: keyword-based, sentence, questions, and others. Each bar represents the accuracy of the chatbot's responses for a specific input type. The "Keyword-based" input type, represented by the tallest bar on the graph with a value of 95, indicates that the chatbot exhibits high accuracy in understanding and responding to user queries based on specific keywords. Users who input queries using relevant keywords can expect accurate and relevant responses from the chatbot. The "Sentence" input type, represented by the second tallest bar with a value of 70, signifies that the chatbot's accuracy is slightly lower when users input queries in the form of complete sentences.

The chatbot may encounter challenges in understanding and providing accurate responses to more complex sentence structures. The "Questions" input type, represented by the third bar with a value of 80, suggests that the chatbot performs relatively well in accurately interpreting and responding to user queries in the form of direct questions. Users can expect reasonably accurate answers when they ask specific questions to the chatbot. The "Others" input type, represented by the shortest bar with a value of 60, represents queries that do not fall into the previous categories. The chatbot's accuracy in handling these diverse and miscellaneous queries is relatively lower compared to the other input types. Overall, the bar graph provides a visual representation of the chatbot's performance across different types of inputs, showcasing the strengths and areas for improvement in accurately responding to user queries.

# CHAPTER 7 CONCLUSION AND FUTURE WORKS

# CONCLUSION

In conclusion, the development of the LICET campus chatbot utilizing cosine similarity and a virtual tour provides a valuable resource for students and prospective students who are seeking admission-related information. The chatbot is able to provide accurate and relevant information to users, such information about the admission process, admission requirements, and deadlines. Additionally, the virtual tour allows prospective students to explore different areas of the campus, such as the classrooms, library, and laboratories.

# FUTURE WORK

The LICET campus chatbot using cosine similarity and a virtual tour has several potential future enhancements, including, Integration with other platforms, such as social media or messaging apps, to make the chatbot more accessible to users. Incorporating additional features, such as a chatbot-based career guidance tool or an interactive campus map, to further enhance the user experience. Furthermore, incorporating machine learning techniques can enhance the chatbot's ability to understand and respond to complex user inquiries. Additionally, integrating a feedback mechanism can gather user suggestions for continuous improvement of the chatbot's performance.

# APPENDIX APPENDIX A - SOURCE CODE

**Final.py**

import sys, webbrowser

import subprocess

from PyQt5.QtWidgets import QApplication, QVBoxLayout, QHBoxLayout

from PyQt5.QtWidgets import QWidget, QLabel, QPushButton, QToolButton

from PyQt5.QtGui import QFont, QPalette, QColor, QIcon

from PyQt5 import QtCore

from PyQt5.QtCore import QSize

from PyQt5.QtGui import QPixmap

from PyQt5.QtWidgets import QLabel

from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QHBoxLayout, QVBoxLayout

from PyQt5.QtWidgets import QVBoxLayout, QLabel, QPushButton

from PyQt5.QtCore import Qt

from PyQt5.QtGui import QIcon, QPixmap

from PyQt5.QtWidgets import QToolButton

from PyQt5.QtGui import QDesktopServices

from PyQt5.QtCore import QUrl

from PyQt5.QtWidgets import QApplication, QLabel

from PyQt5.QtWidgets import QApplication, QPushButton

class ChatbotUI(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Set window title and dimensions

self.setWindowTitle('LICET Admission Enquiry Bot')

self.setFixedSize(500,800)

self.image\_label = QLabel(self)

self.image\_label.setGeometry(150, 120, 200,200)

self.image\_label.setStyleSheet("background-image: url('logo2.png'); background-repeat: no-repeat; background-position: center;")

# Display quote below image

self.quote\_label = QLabel(self)

self.quote\_label.setGeometry(0, 350, 500, 75)

self.quote\_label.setText("Let your light shine")

self.quote\_label.setAlignment(Qt.AlignCenter)

font = QFont('Arial', 14)

font.setBold(True)

self.quote\_label.setFont(font)

self.quote\_label.setStyleSheet("color: white; background-color: #293241;")

# Create functionality buttons

self.button1 = QPushButton('VIRTUAL CAMPUS TOUR')

self.button1.clicked.connect(self.open\_tour)

self.button2 = QPushButton('COURSE PREFERENCE FORM')

self.button2.clicked.connect(self.open\_course\_page)

self.button3 = QPushButton('DOWNLOAD PROSPECTUS')

self.button3.clicked.connect(self.execute\_other\_script)

self.button4 = QPushButton('CUTOFF INSIGHTS')

self.button4.clicked.connect(self.cutoff)

self.button5 = QPushButton('COURSE SUGGESTION')

self.button5.clicked.connect(self.sugg)

self.button6 = QPushButton('REVIEWS')

self.button6.clicked.connect(self.reviews)

self.button7 = QPushButton('ADMISSION PORTAL')

self.button7.clicked.connect(self.open\_adm\_page)

self.button8 = QPushButton('PLACEMENT INSIGHTS')

self.button8.clicked.connect(self.open\_html\_file)

self.button9 = QPushButton('ABOUT LICET')

self.button9.clicked.connect(self.open\_about\_page)

self.button14 = QPushButton('SPORTS')

self.button14.clicked.connect(self.open\_sports\_page)

self.button15 = QPushButton('COURSES OFFERED')

self.button15.clicked.connect(self.open\_courses\_page)

self.button16 = QPushButton('GALLERY')

self.button16.clicked.connect(self.open\_gallery)

self.button10 = QToolButton()

icon = QIcon()

icon.addPixmap(QPixmap("send\_icon.png"), QIcon.Normal, QIcon.Off)

self.button10.setIcon(icon)

self.button10.setText('')

self.button10.clicked.connect(self.chat)

icon\_size = QSize(60, 60)

self.button10.setIconSize(icon\_size)

self.button10.setToolButtonStyle(Qt.ToolButtonTextBesideIcon)

self.button11= QToolButton()

icon = QIcon()

icon.addPixmap(QPixmap("voice\_icon.png"), QIcon.Normal, QIcon.Off)

self.button11.setIcon(icon)

self.button11.setText('')

self.button11.clicked.connect(self.voice)

icon\_size = QSize(60, 60)

self.button11.setIconSize(icon\_size)

self.button11.setToolButtonStyle(Qt.ToolButtonTextBesideIcon)

self.button12= QToolButton()

icon = QIcon()

icon.addPixmap(QPixmap("call.png"), QIcon.Normal, QIcon.Off)

self.button12.setIcon(icon)

self.button12.setText('')

self.button12.clicked.connect(self.call)

icon\_size = QSize(60,60)

self.button12.setIconSize(icon\_size)

self.button12.setToolButtonStyle(Qt.ToolButtonTextBesideIcon)

self.button13 = QPushButton('ABOUT LICET')

self.button13.clicked.connect(self.open\_about\_page)

button\_style = """

QPushButton {

border: 2px solid #555555;

border-radius: 10px;

padding: 10px;

font-size: 16px;

}

QPushButton#special\_button3:hover {

border: 3.5px solid #3B3A39;

}

QPushButton#special\_button1:hover {

border: 3.5px solid #3B3A39;

}

QPushButton#special\_button3 {

background-color: #ffc315;

color: black;

}

QPushButton#special\_button1 {

background-color: #137ec2;

color: white;

}"""

self.button1.setStyleSheet(button\_style)

self.button2.setStyleSheet(button\_style)

self.button3.setStyleSheet(button\_style)

self.button4.setStyleSheet(button\_style)

self.button5.setStyleSheet(button\_style)

self.button6.setStyleSheet(button\_style)

self.button7.setStyleSheet(button\_style)

self.button8.setStyleSheet(button\_style)

self.button9.setStyleSheet(button\_style)

self.button10.setStyleSheet(button\_style)

self.button11.setStyleSheet(button\_style)

self.button12.setStyleSheet(button\_style)

self.button14.setStyleSheet(button\_style)

self.button15.setStyleSheet(button\_style)

self.button16.setStyleSheet(button\_style)

self.button1.setObjectName('special\_button3')

self.button2.setObjectName('special\_button3')

self.button3.setObjectName('special\_button3')

self.button4.setObjectName('special\_button3')

self.button5.setObjectName('special\_button1')

self.button6.setObjectName('special\_button1')

self.button7.setObjectName('special\_button1')

self.button8.setObjectName('special\_button3')

self.button9.setObjectName('special\_button3')

self.button14.setObjectName('special\_button1')

self.button15.setObjectName('special\_button1')

self.button16.setObjectName('special\_button1')

self.button1.setStyleSheet(button\_style)

self.button2.setStyleSheet(button\_style)

# Create layouts and add UI elements

messages\_layout = QVBoxLayout()

input\_layout = QHBoxLayout()

button\_layout = QVBoxLayout()

row1\_layout = QHBoxLayout()

row1\_layout.addWidget(self.button1)

row1\_layout.addWidget(self.button2)

row3\_layout = QHBoxLayout()

row3\_layout.addWidget(self.button3)

row3\_layout.addWidget(self.button4)

row2\_layout = QHBoxLayout()

row2\_layout.addWidget(self.button5)

row2\_layout.addWidget(self.button6)

row2\_layout.addWidget(self.button7)

row6\_layout=QHBoxLayout()

row6\_layout.addWidget(self.button8)

row6\_layout.addWidget(self.button9)

row4\_layout = QHBoxLayout()

row4\_layout.addWidget(self.button14)

row4\_layout.addWidget(self.button15)

row4\_layout.addWidget(self.button16)

row5\_layout = QHBoxLayout()

row5\_layout.addWidget(self.button10)

row5\_layout.addWidget(self.button11)

row5\_layout.addWidget(self.button12)

button\_layout.addLayout(row1\_layout)

button\_layout.addLayout(row2\_layout)

button\_layout.addLayout(row3\_layout)

button\_layout.addLayout(row4\_layout)

button\_layout.addLayout(row6\_layout)

button\_layout.addLayout(row5\_layout)

main\_layout = QVBoxLayout()

welcome\_label = QLabel('\nWELCOME TO LICET \n ADMISSION ENQUIRY CHATBOT\n')

welcome\_label.setAlignment(QtCore.Qt.AlignHCenter)

font2 = QFont('Arial', 8)

welcome\_label.setFont(font2)

main\_layout.addWidget(welcome\_label)

main\_layout.addWidget(QLabel(''))

main\_layout.addLayout(button\_layout)

main\_layout.addLayout(messages\_layout)

main\_layout.addLayout(input\_layout)

self.setLayout(main\_layout)

# Set background color and font for welcome message

palette = QPalette()

palette.setColor(QPalette.WindowText, QColor('blue'))

self.layout().itemAt(0).widget().setPalette(palette)

font = QFont()

font.setBold(True)

font.setPointSize(16)

self.layout().itemAt(0).widget().setFont(font)

def open\_adm\_page(self):

webbrowser.open('https://admission.licet.ac.in/login/index.php?\_gl=1\*w18m7r\*\_ga\*NDA4NjI2NjU2LjE2ODA5MjcxMTE.\*\_ga\_762BPEDVM4\*MTY4MjE1MDY2OC4yLjEuMTY4MjE1NjQzNS4wLjAuMA..&\_ga=2.268898545.1072618897.1682143494-408626656.1680927111')

def open\_tour(self):

file\_path = 'index.html'

url = QUrl.fromLocalFile(file\_path)

QDesktopServices.openUrl(url)

def cutoff(self):

subprocess.run(['python', 'cutoff.py'])

def open\_about\_page(self):

subprocess.run(['python', 'avatar.py'])

def reviews(self):

webbrowser.open('https://kanishkumar-k.github.io/Licet\_reviews/')

def execute\_other\_script(self):

subprocess.run(['python', 'prospectus.py'])

def open\_html\_file(self):

file\_path = 'https://kanishkumar-k.github.io/Placement-Details/Placement.html'

url = QUrl.fromLocalFile(file\_path)

QDesktopServices.openUrl(url)

def chat(self):

subprocess.run(['python', 'gichat.py'])

def voice(self):

subprocess.run(['python', 'gispeech.py'])

def call(self):

subprocess.run(['python', 'call.py'])

def sugg(self):

subprocess.run(['python', 'suggestion.py'])

def open\_sports\_page(self):

subprocess.run(['python', 'sports.py'])

def open\_course\_page(self):

subprocess.run(['python', 'course.py'])

def open\_courses\_page(self):

subprocess.run(['python', 'courses.py'])

def open\_gallery(self):

subprocess.run(['python', 'testgallery.py'])

def run\_ui():

# Create application and UI

app = QApplication(sys.argv)

ui = ChatbotUI()

ui.show()

# Run application

sys.exit(app.exec\_())

if \_\_name\_\_ == '\_\_main\_\_':

run\_ui()

index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

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

<title>Virtual Tour</title>

<style>

#left-pane {

float: left;

width: 200px;

height: 900px;

background-color: #f2f2f2;

border-right: 1px solid #ccc;

}

#left-pane ul {

list-style-type: none;

padding: 0;

margin: 0;

}

#left-pane li {

margin: 10px 0;

}

#left-pane a {

display: block;

padding: 10px;

color: #333;

text-decoration: none;

border-radius: 5px;

}

#left-pane a:hover {

background-color: #333;

color: #fff;

}

#central-screen {

float: left;

margin-left: 50px;

}

#map {

position: fixed;

bottom: 10px;

right: 10px;

width: 100px;

height: 100px;

border-radius: 50%;

overflow: hidden;

transition: all 0.3s ease-in-out;

}

#map:hover {

width: 300px;

height: 200px;

}

#map iframe {

width: 100%;

height: 100%;

border: none;

}

#map-icon {

position: fixed;

bottom: 10px;

right: 10px;

width: 50px;

height: 50px;

background-color: #fff;

border-radius: 25px;

box-shadow: 0px 2px 4px rgba(0, 0, 0, 0.4);

z-index: 1;

}

#map-icon:hover {

cursor: pointer;

box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.4);

}

#map-icon img {

width: 100%;

height: 100%;

border-radius: 25px;

}

</style>

</head>

<body bgcolor="lightgrey">

<div id="left-pane">

<ul>

<li><H2><center>Virtual Campus Tour</center></H2></li><br />

<li><a href="#" onclick="changeImage('https://i.ibb.co/jTcxsV0/G01.jpg')">AUDITORIUM GO1</a></li>

<li><a href="#" onclick="changeImage('https://i.ibb.co/3BxZ7MN/DSC00340-1.jpg')">LIBRARY</a></li>

<li><a href="#" onclick="changeImage('https://i.ibb.co/jwx28HQ/DSC00305.jpg')">FABLAB</a></li>

<li><a href="#" onclick="changeImage('https://i.ibb.co/g7V6h77/DSC00426.jpg')">MECHANICAL DEPARTMENT LAB</a></li>

<li><a href="#" onclick="changeImage('https://i.ibb.co/mvn5Gwk/DSC00454.jpg')">ENTRANCE</a></li>

<li><a href="#" onclick="changeImage('https://i.ibb.co/cCXPbRx/DSC00321.jpg')">CSE DEPARTMENT LAB</a></li>

<li><a href="#" onclick="changeImage('https://i.ibb.co/bXbVM3r/clr.jpg')">CLASSROOM</a></li>

<li><a href="#" onclick="changeImage('')">SEMINAR HALL</a></li>

<li><a href="#" onclick="changeImage('')">SMART ROOM</a></li>

</ul>

</div>

<div id="central-screen">

<iframe id="panorama-viewer" width="1200" height="700" title="Panorama Viewer" scrolling="no" src="https://renderstuff.com/tools/360-panorama-web-viewer-embed/?image=https://i.ibb.co/jTcxsV0/G01.jpg"></iframe>

</div>

<div id="map-icon">

<a href="https://goo.gl/maps/Pcb6mv6icwgX8K7GA" target="\_blank">

<img src="https://www.gstatic.com/images/branding/product/1x/maps\_32dp.png" alt="Map Icon">

</a>

</div>

<script>

function changeImage(imageURL) {

var viewer = document.getElementById("panorama-viewer");

viewer.src = "https://renderstuff.com/tools/360-panorama-web-viewer-embed/?image=" + imageURL;

}

</script>

</body>

</html>

Prospectus.py

import sys

import urllib.request

import os

import webbrowser

import os.path

from pathlib import Path

from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel, QPushButton

from PyQt5.QtGui import QLinearGradient, QColor, QPalette, QBrush

from PyQt5.QtGui import QLinearGradient, QColor, QBrush

from PyQt5.QtWidgets import QApplication, QMainWindow

from PyQt5.QtGui import QPixmap

class BrochureViewer(QMainWindow):

def \_\_init\_\_(self, pdf\_url, college\_name, description):

super().\_\_init\_\_()

self.pdf\_url = pdf\_url

self.college\_name = college\_name

self.description = description

self.setWindowTitle(college\_name + " Brochure Viewer")

self.setFixedSize(500, 800)

self.setStyleSheet("background-color: #F9F9F9;")

# Add image above the buttons

image\_label = QLabel(self)

pixmap = QPixmap("logo.jpg")

pixmap = pixmap.scaledToWidth(200)

image\_label.setPixmap(pixmap)

image\_label.setGeometry(150, 20, 250, 250)

self.college\_label = QLabel(self)

self.college\_label.setText(college\_name)

self.college\_label.setGeometry(20, 290, 460, 40)

self.college\_label.setStyleSheet("font: bold 24px Arial; color: #007FFF;")

self.desc\_label = QLabel(self)

self.desc\_label.setText(description)

self.desc\_label.setWordWrap(True)

self.desc\_label.setGeometry(20, 340, 460, 100)

self.desc\_label.setStyleSheet("font: bold 16px Arial;")

self.view\_button = QPushButton(self)

self.view\_button.setText("VIEW PROSPECTUS")

self.view\_button.setGeometry(20, 470, 460, 60)

self.view\_button.clicked.connect(self.view\_brochure)

self.view\_button.setStyleSheet("font: bold 17px Arial; background-color: #007FFF; color: #FFFFFF;")

self.download\_button = QPushButton(self)

self.download\_button.setText("DOWNLOAD PROSPECTUS")

self.download\_button.setGeometry(20, 550, 460, 60)

self.download\_button.clicked.connect(self.download\_brochure)

self.download\_button.setStyleSheet("font: bold 17px Arial; background-color: #ffc315; color: #000000;")

self.close\_button = QPushButton(self)

self.close\_button.setText("EXIT")

self.close\_button.setGeometry(20, 630, 460, 60)

self.close\_button.clicked.connect(self.close)

self.close\_button.setStyleSheet("font: bold 17px Arial; background-color: red; color: #FFFFFF;")

def view\_brochure(self):

webbrowser.open\_new(self.pdf\_url)

def download\_brochure(self):

pdf\_filename = self.college\_name.replace(" ", "\_").lower() + "\_brochure.pdf"

home = str(Path.home())

downloads\_path = os.path.join(home, "Downloads")

pdf\_file\_path = os.path.join(downloads\_path, pdf\_filename)

if os.path.exists(pdf\_file\_path):

print("File already exists in Downloads folder.")

else:

urllib.request.urlretrieve(self.pdf\_url, pdf\_file\_path)

print("Brochure downloaded successfully in Downloads folder.")

os.startfile(downloads\_path)

if \_\_name\_\_ == '\_\_main\_\_':

# Set the URL of the PDF file to open

pdf\_url = "https://licet.ac.in/wp-content/uploads/2021/05/Prospectus-2023-1.pdf"

# Set the name of the college and a brief description of the brochure

college\_name = "\t LICET PROSPECTUS"

description = "Learn more about the application process, campus life, and more at LICET by downloading Prospectus '23!"

app = QApplication(sys.argv)

brochure\_viewer = BrochureViewer(pdf\_url, college\_name, description)

brochure\_viewer.show()

sys.exit(app.exec\_())

text.py

from PyQt5.QtCore import Qt

from PyQt5.QtGui import QFont

from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QPushButton, QScrollArea, QListWidgetItem

import random

import nltk

from PyQt5.QtCore import QPropertyAnimation

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

from PyQt5.QtGui import QLinearGradient, QColor

from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QPushButton, QScrollArea

from PyQt5.QtGui import QBrush, QColor

from PyQt5.QtCore import Qt

from PyQt5.QtCore import QEasingCurve,QAbstractAnimation

from PyQt5.QtGui import QBrush, QColor, QPalette

class ChatbotGUI(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Set up the GUI

self.setWindowTitle('LICET Admission Enquiry Bot')

self.setFixedSize(500,800)

# Set up the chat history label

self.setStyleSheet("border: 1px solid black; padding: 5px; color:#000000; font-weight:italics; background-image: url('C:/Users/karun/OneDrive/Desktop/py/image12.png'); background-repeat: no-repeat; background-position: center; background-attachment: fixed;")

self.chat\_history = QLabel(self)

font = QFont("Arial", 10)

palette = QPalette()

palette.setColor(QPalette.WindowText, Qt.black)

style\_sheet = "background-color: #f2f2f2; border: 1px solid #ccc;"

self.chat\_history.setFont(font)

self.chat\_history.setPalette(palette)

self.chat\_history.setStyleSheet(style\_sheet)

self.chat\_history.setText("\nWelcome to LICET Chatbot! How can i assist you? \n\n")

self.chat\_history.setAlignment(Qt.AlignTop)

self.chat\_history.setWordWrap(True)

self.chat\_history.setFont(QFont("Open Sans", 12))

self.chat\_history.setStyleSheet("""

background-color: #fff;

border-radius: 10px;

padding: 10px;

border: 1px solid #ccc;

""")

# Add a scroll area to the chat history label

scroll = QScrollArea(self)

scroll.setWidgetResizable(True)

scroll.setWidget(self.chat\_history)

scroll.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOn)

scroll.setStyleSheet("""

QScrollBar:vertical {

border: none;

background-color: #f2f2f2;

width: 10px;

margin: 0px 0 0px 0;

}

QScrollBar::handle:vertical {

background-color: #888;

min-height: 50px;

}

QScrollBar::add-line:vertical {

background: none;

}

QScrollBar::sub-line:vertical {

background: none;

}

QScrollBar {

background: #FFFFFF;

}

""")

# Set up the user input box and submit button

self.user\_input = QLineEdit(self)

self.user\_input.setStyleSheet('''

QLineEdit {

background-image: url();

background-color: #f2f2f2;

border: 2px solid #ccc;

border-radius: 10px;

padding: 5px;

font-size: 12px;

}

QLineEdit:hover {

background-color: #c2c4c0;

border: 2px solid #aaa;

}

''')

self.user\_input.setFont(QFont("Open Sans", 11, weight=QFont.Normal))

self.user\_input.setPlaceholderText('Type here:') # Set the placeholder text for the input box

self.submit\_button = QPushButton('Submit', self)

self.submit\_button.clicked.connect(self.submit\_message)

self.submit\_button.setStyleSheet("""

QPushButton {

background-color: #006600;

color: white;

padding: 10px;

border: none;

border-radius: 10px;

font-size: 14px;

}

QPushButton:hover {

background-color: #004d00;

}

""")

self.close\_button = QPushButton('Close', self)

self.close\_button.setFont(QFont("Open Sans", 10, weight=QFont.Normal))

self.close\_button.setStyleSheet("""

QPushButton {

background-color: #f44336;

color: white;

padding: 10px;

border: none;

border-radius: 10px;

font-size: 14px;

}

QPushButton:hover {

background-color: #d32f2f;

}

""")

self.close\_button.clicked.connect(self.close)

# Set up the layout

vbox = QVBoxLayout()

vbox.addWidget(scroll)

hbox = QHBoxLayout()

hbox.addWidget(self.user\_input)

hbox.addWidget(self.submit\_button)

hbox.addWidget(self.close\_button)

vbox.addLayout(hbox)

self.setLayout(vbox)

self.vectorizer = TfidfVectorizer()

self.vectorizer.fit(list(self.responses.keys()))

def submit\_message(self):

user\_input = self.user\_input.text()

response = self.process\_input(user\_input)

self.add\_chat\_entry("User: " + user\_input, "Bot: " + response)

self.user\_input.clear()

pulse\_animation = QPropertyAnimation(self.submit\_button, b"opacity")

pulse\_animation.setDuration(400)

pulse\_animation.setStartValue(0.2)

pulse\_animation.setEndValue(1)

pulse\_animation.setEasingCurve(QEasingCurve.OutQuad)

pulse\_animation.start(QAbstractAnimation.DeleteWhenStopped)

def process\_input(self, input\_text):

# Tokenize the input text

input\_tokens = nltk.word\_tokenize(input\_text.lower())

# Get the TF-IDF vectors for the possible user inputs and the user input

input\_vector = self.vectorizer.transform(input\_tokens)

response\_vectors = self.vectorizer.transform(self.responses.keys())

# Calculate the cosine similarities between the input vector and each response vector

similarities = cosine\_similarity(input\_vector, response\_vectors)

# Get the index of the most similar response

response\_index = similarities.argmax()

# Return the response

return self.responses[list(self.responses.keys())[response\_index]][0]

def add\_chat\_entry(self, user\_message, bot\_message):

# Append the new chat entry to the chat history label

self.chat\_history.setText(self.chat\_history.text() + "\n" + user\_message + "\n\n" + bot\_message + "\n")

def add\_bot\_message(self, text):

item= QListWidgetItem(text)

item.setTextAlignment(Qt.AlignRight)

item.setData(Qt.UserRole, "bot")

item.setData(Qt.UserRole + 1, True) # Set is\_bot property to True

self.chat\_history.addItem(item)

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication([])

gui = ChatbotGUI()

gui.show()

app.exec\_()

speech.py

import random

import nltk

import speech\_recognition as sr

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

# Define the chatbot function

def chatbot():

print("Welcome to LICET college admission chatbot. How can I help you today?")

# Define the NLP function

def process\_input(input\_text):

# Tokenize the input text

input\_tokens = nltk.word\_tokenize(input\_text.lower())

# Get the TF-IDF vectors for the possible user inputs and the user input

vectorizer = TfidfVectorizer()

vectorizer.fit(list(responses.keys()) + input\_tokens)

input\_vector = vectorizer.transform(input\_tokens)

response\_vectors = vectorizer.transform(responses.keys())

# Calculate the cosine similarities between the input vector and each response vector

similarities = cosine\_similarity(input\_vector, response\_vectors)

# Get the index of the most similar response

response\_index = similarities.argmax()

# Return the response

return random.choice(responses[list(responses.keys())[response\_index]])

# Define the speech recognition function

def recognize\_speech():

# Initialize the recognizer and microphone

r = sr.Recognizer()

mic = sr.Microphone()

# Adjust for ambient noise

with mic as source:

r.adjust\_for\_ambient\_noise(source)

# Prompt the user to speak and recognize their speech

with mic as source:

print("Speak now!")

audio = r.listen(source)

try:

# Convert speech to text

text = r.recognize\_google(audio)

print("You: " + text)

return text

except sr.UnknownValueError:

# If the speech cannot be recognized, ask the user to repeat

print("Sorry, I didn't catch that. Please try again.")

return recognize\_speech()

except sr.RequestError as e:

# If there is an error with the speech recognition API, notify the user

print("Could not request results from speech recognition service; {0}".format(e))

return ""

# Loop until the user exits

while True:

# Get the user input

user\_input = recognize\_speech()

# Check if the user wants to exit

if "exit" in user\_input:

response = random.choice(responses["exit"])

print("Bot: " + response)

break

# Get the bot response

response = process\_input(user\_input)

print("Bot: " + response)

# Call the chatbot function

chatbot()

suggestion.py

import tkinter as tk

from tkinter import ttk

# Create a dictionary of courses with their associated interests, hobbies, and achievements

courses = {

'Computer Science Engineering': {

'interests': ['programming', 'gaming', 'mathematics', 'web development', 'Artificial intelligence', 'Machine Learning', 'software development'],

'hobbies': ['coding projects', 'robotics'],

'achievements': ['programming competitions', 'hackathons']

},

'Mechanical Engineering': {

'interests': ['cars', 'machines'],

'hobbies': ['building machines', 'mechanical puzzles'],

'achievements': ['robotics competitions', 'engineering projects']

},

'Electrical and Electronics Engineering': {

'interests': ['circuits', 'robotics', 'Internet of Things', 'VLSI Design', 'Renewable energy resources'],

'hobbies': [],

'achievements': []

},

'Information technology': {

'interests': ['programming', 'Data analysis', 'cybersecurity', 'cloud computing', 'computer networks', 'Blockchain'],

'hobbies': [],

'achievements': []

},

'Electronics and communication Engineering': {

'interests': ['circuits', 'embedded systems', 'photonics', 'VLSI Design', 'wireless communications'],

'hobbies': [],

'achievements': []

}

}

# Define a function to suggest courses based on interests, hobbies, and achievements

def suggest\_course(interests, hobbies, achievements):

suggested\_courses = []

for course, course\_data in courses.items():

if any(interest in course\_data['interests'] for interest in interests):

suggested\_courses.append(course)

elif any(hobby in course\_data['hobbies'] for hobby in hobbies):

suggested\_courses.append(course)

elif any(achievement in course\_data['achievements'] for achievement in achievements):

suggested\_courses.append(course)

suggested\_courses = list(set(suggested\_courses))

if suggested\_courses:

return f"You could consider these courses: {', '.join(suggested\_courses)}"

else:

return "Sorry, there are no courses available for your interests, hobbies, and achievements."

# Define a function to interact with the user

def chat():

def on\_submit():

interests = [interests\_combo.get()]

hobbies = [hobbies\_combo.get()]

achievements = [achievements\_combo.get()]

result = suggest\_course(interests, hobbies, achievements)

result\_label.config(text=result)

# Create GUI

window = tk.Tk()

window.title("Course Suggestion Chatbot")

window.geometry("500x400")

title\_label = ttk.Label(window, text="LICET - Course Suggestion Bot ", font=("TkDefaultFont", 16), justify="center")

title\_label.grid(column=0, row=0, columnspan=2, padx=130, pady=20)

# Create input fields

interests\_label = ttk.Label(window, text="Select your interests:")

interests\_label.grid(column=0, row=1, padx=40, pady=10, sticky=tk.W)

interests\_values = ["programming", 'cars', 'machines', 'circuits', 'robotics', 'Internet of Things', 'VLSI Design', 'circuits', 'embedded systems', 'photonics', 'VLSI Design', 'wireless communications', 'Renewable energy resources', "gaming", "mathematics", 'Data analysis', 'cybersecurity', 'cloud computing', 'computer networks', 'Blockchain', "web development", "Artificial intelligence", "Machine Learning", "software development"]

interests\_combo = ttk.Combobox(window, values=interests\_values, state='readonly')

interests\_combo.current(0)

interests\_combo.grid(column=1, row=1, padx=0, pady=10, sticky=tk.W)

hobbies\_label = ttk.Label(window, text="Select your hobbies:")

hobbies\_label.grid(column=0, row=2, padx=40, pady=10, sticky=tk.W)

hobbies\_values = ["reading", "writing", "photography", "painting", "traveling", "cooking", "music", "dance", "sports", "gardening", "fishing", "camping", "hiking"]

hobbies\_combo = ttk.Combobox(window, values=hobbies\_values, state='readonly')

hobbies\_combo.current(0)

hobbies\_combo.grid(column=1, row=2, padx=0, pady=10, sticky=tk.W)

achievements\_label = ttk.Label(window, text="Select your achievements:")

achievements\_label.grid(column=0, row=3, padx=40, pady=10, sticky=tk.W)

achievements\_values = ["programming competitions", "hackathons"]

achievements\_combo = ttk.Combobox(window, values=achievements\_values, state='readonly')

achievements\_combo.current(0)

achievements\_combo.grid(column=1, row=3, padx=0, pady=10, sticky=tk.W)

# Create submit button

submit\_button = ttk.Button(window, text="Get Course Suggestions", command=on\_submit, style='Submit.TButton')

submit\_button.grid(column=0, row=4, columnspan=2, padx=0, pady=10)

# Create result label

result\_label = ttk.Label(window, text="", font=("Arial", 11), wraplength=350, justify="center")

result\_label.grid(column=0, row=5, columnspan=2, padx=10, pady=20)

window.mainloop()

# Start the chatbot

chat()

cutoff.py

import sys

from PyQt5 import QtWidgets, QtGui, QtCore

from cutoff\_calculator import cutoff\_calculator

class CollegeAdmissionChatbot(QtWidgets.QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Create the UI elements

self.setWindowTitle('LICET Admission Chatbot')

self.setWindowIcon(QtGui.QIcon('icon.png'))

self.setStyleSheet('background-color: #f5f5f5;')

self.setFixedSize(500, 800)

self.heading\_label = QtWidgets.QLabel('LICET Admission Chatbot - CUTOFF Insights')

self.heading\_label.setStyleSheet('font-size: 24px; font-family: Arial; font-weight: bold; color: #333333;')

self.heading\_label.setWordWrap(True)

self.marks\_label = QtWidgets.QLabel('Please enter your marks in Maths, Physics, and Chemistry (Seperated by comma\'s)')

self.marks\_label.setStyleSheet('font-size: 20px; font-family: Arial; color: #333333;')

self.marks\_label.setWordWrap(True)

self.cast\_label = QtWidgets.QLabel('Please select your category:')

self.cast\_label.setStyleSheet('font-size: 20px; font-family: Arial; color: #333333;')

self.cast\_label.setWordWrap(True)

self.marks\_input = QtWidgets.QLineEdit()

self.marks\_input.setStyleSheet('font-size: 16px; font-family: Arial; color: #333333; margin-top: 0px;')

self.cast\_combo = QtWidgets.QComboBox()

self.cast\_combo.setStyleSheet('font-size: 16px; font-family: Arial; color: #333333; margin-top: 0px;')

self.cast\_combo.addItem("OC")

self.cast\_combo.addItem("BCM")

self.cast\_combo.addItem("BC")

self.cast\_combo.addItem("MBC")

self.cast\_combo.addItem("SC")

self.cast\_combo.addItem("SCA")

self.cast\_combo.addItem("ST")

self.cutoff\_label = QtWidgets.QLabel('Your cutoff mark is: ')

self.cutoff\_label.setWordWrap(True)

self.cutoff\_label.setStyleSheet('font-size: 16px; font-family: Arial; color: #333333; margin-top: 0px;')

self.cutoff\_output = QtWidgets.QLabel()

self.cutoff\_output.setStyleSheet('font-size: 46px; font-family: Arial; color: #008080;')

self.result\_label = QtWidgets.QLabel()

self.result\_label.setWordWrap(True)

self.result\_label.setStyleSheet('font-size: 16px; font-family: Arial; color: #333333;')

self.submit\_button = QtWidgets.QPushButton('GET CUTOFF INSIGHTS')

self.submit\_button.setStyleSheet('font-size: 16px; font-family: Arial; color: #ffffff; background-color: #008080; padding:20px')

self.submit\_button.clicked.connect(self.calculate\_cutoff)

self.result\_label.setWordWrap(True)

# Create a layout and add the UI elements to it

layout = QtWidgets.QVBoxLayout()

layout.addWidget(self.heading\_label)

layout.setContentsMargins(50, 50, 50, 50)

layout.addWidget(self.marks\_label)

layout.addWidget(self.marks\_input)

layout.addWidget(self.cast\_label)

layout.addWidget(self.cast\_combo)

layout.addWidget(self.cutoff\_label)

layout.addWidget(self.cutoff\_output)

layout.addWidget(self.result\_label)

layout.addWidget(self.submit\_button)

# Set the layout for the widget

self.setLayout(layout)

def calculate\_cutoff(self):

# Get the user input

user\_input = self.marks\_input.text()

# Get the selected caste

selected\_caste = self.cast\_combo.currentText()

# Calculate the cutoff mark

try:

maths\_marks, physics\_marks, chemistry\_marks = map(int, user\_input.split(","))

except ValueError:

self.result\_label.setText('Invalid input! Please enter your marks in the format "maths,physics,chemistry".')

return

cutoff\_marks = cutoff\_calculator(maths\_marks, physics\_marks, chemistry\_marks)

# Display the result

self.cutoff\_output.setText(str(cutoff\_marks))

if selected\_caste == 'OC':

if cutoff\_marks >= 185:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for CSE, IT, ECE, EE, Mech departments.')

elif cutoff\_marks >= 180:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for IT, ECE, EEE, Mech departments.')

elif cutoff\_marks >= 178:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for ECE, EEE, Mech departments.')

elif cutoff\_marks >= 171:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for EEE, Mech departments.')

elif cutoff\_marks >= 161:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for Mech department.')

else:

self.result\_label.setText('Sorry, your marks do not meet the cutoff criteria for any department. .')

if cutoff\_marks >= 140 and cutoff\_marks < 190:

self.result\_label.setText('\nHowever, you may be eligible for admission for other departments through management quota.')

elif selected\_caste == 'BC':

if cutoff\_marks >= 178:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for CSE, IT, ECE, EE, Mech departments.')

elif cutoff\_marks >= 175:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for ECE, IT, EEE, Mech departments.')

elif cutoff\_marks >= 169:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for IT, EEE, Mech departments.')

elif cutoff\_marks >= 166:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for EEE, Mech departments.')

elif cutoff\_marks >= 150:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for Mech department.')

elif cutoff\_marks >= 140:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for Civil, Chemical department but unfortunately we dont provide those departments.')

else:

self.result\_label.setText('Sorry, your marks do not meet the cutoff criteria for any department. .')

if cutoff\_marks >= 130 and cutoff\_marks < 180:

self.result\_label.setText('\nHowever, you may be eligible for admission for other departments through management quota.')

elif selected\_caste == 'BCM':

if cutoff\_marks >= 174:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for CSE, IT, ECE, EE, Mech departments.')

elif cutoff\_marks >= 170:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for ECE, IT, EEE, Mech departments.')

elif cutoff\_marks >= 165:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for IT, EEE, Mech departments.')

elif cutoff\_marks >= 160:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for EEE, Mech departments.')

elif cutoff\_marks >= 150:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for Mech department.')

elif cutoff\_marks >= 140:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for Civil, Chemical department but unfortunately we dont provide those departments.')

else:

self.result\_label.setText('Sorry, your marks do not meet the cutoff criteria for any department. .')

if cutoff\_marks >= 130 and cutoff\_marks < 180:

self.result\_label.setText('\nHowever, you may be eligible for admission for other departments through management quota.')

elif selected\_caste == 'MBC':

if cutoff\_marks >= 175:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for CSE, IT, ECE, EE, Mech departments.')

elif cutoff\_marks >= 170:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for ECE, IT, EEE, Mech departments.')

elif cutoff\_marks >= 167:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for IT, EEE, Mech departments.')

elif cutoff\_marks >= 163:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for EEE, Mech departments.')

elif cutoff\_marks >= 154:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for Mech department.')

elif cutoff\_marks >= 140:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for Civil, Chemical department but unfortunately we dont provide those departments.')

else:

self.result\_label.setText('Sorry, your marks do not meet the cutoff criteria for any department. .')

if cutoff\_marks >= 130 and cutoff\_marks < 180:

self.result\_label.setText('\nHowever, you may be eligible for admission for other departments through management quota.')

elif selected\_caste == 'SC':

if cutoff\_marks >= 148:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for CSE, IT, ECE, EE, Mech departments.')

elif cutoff\_marks >= 146:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for ECE, IT, EEE, Mech departments.')

elif cutoff\_marks >= 144:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for IT, EEE, Mech departments.')

elif cutoff\_marks >= 127:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for EEE, Mech departments.')

elif cutoff\_marks >= 116:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for Mech department.')

elif cutoff\_marks >= 140:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for Civil, Chemical department but unfortunately we dont provide those departments.')

else:

self.result\_label.setText('Sorry, your marks do not meet the cutoff criteria for any department. .')

if cutoff\_marks >= 130 and cutoff\_marks < 180:

self.result\_label.setText('\nHowever, you may be eligible for admission for other departments through management quota.')

elif selected\_caste == 'SCA':

if cutoff\_marks >= 174:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for CSE, IT, ECE, EE, Mech departments.')

elif cutoff\_marks >= 167:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for IT, ECE, EEE, Mech departments.')

elif cutoff\_marks >= 0:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for ECE, EEE, Mech departments.')

elif cutoff\_marks >= 0:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for EEE, Mech departments.')

elif cutoff\_marks >= 150:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for Mech department.')

elif cutoff\_marks >= 140:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for Civil, Chemical department but unfortunately we dont provide those departments.')

else:

self.result\_label.setText('Sorry, your marks do not meet the cutoff criteria for any department. .')

if cutoff\_marks >= 130 and cutoff\_marks < 180:

self.result\_label.setText('\nHowever, you may be eligible for admission for other departments through management quota.')

elif selected\_caste == 'ST':

if cutoff\_marks >= 0:

self.result\_label.setText('Congratulations! You have a great chance of getting admission through AU counselling for CSE, IT, ECE, EE, Mech departments.')

else:

self.result\_label.setText('\nHowever, you may be eligible for admission for other departments through management quota.')

else:

result\_text = 'Invalid caste selection. Please select from the available options: OC, BCM, BC, MBC, SC, SCA, ST'

def run(self):

self.show()

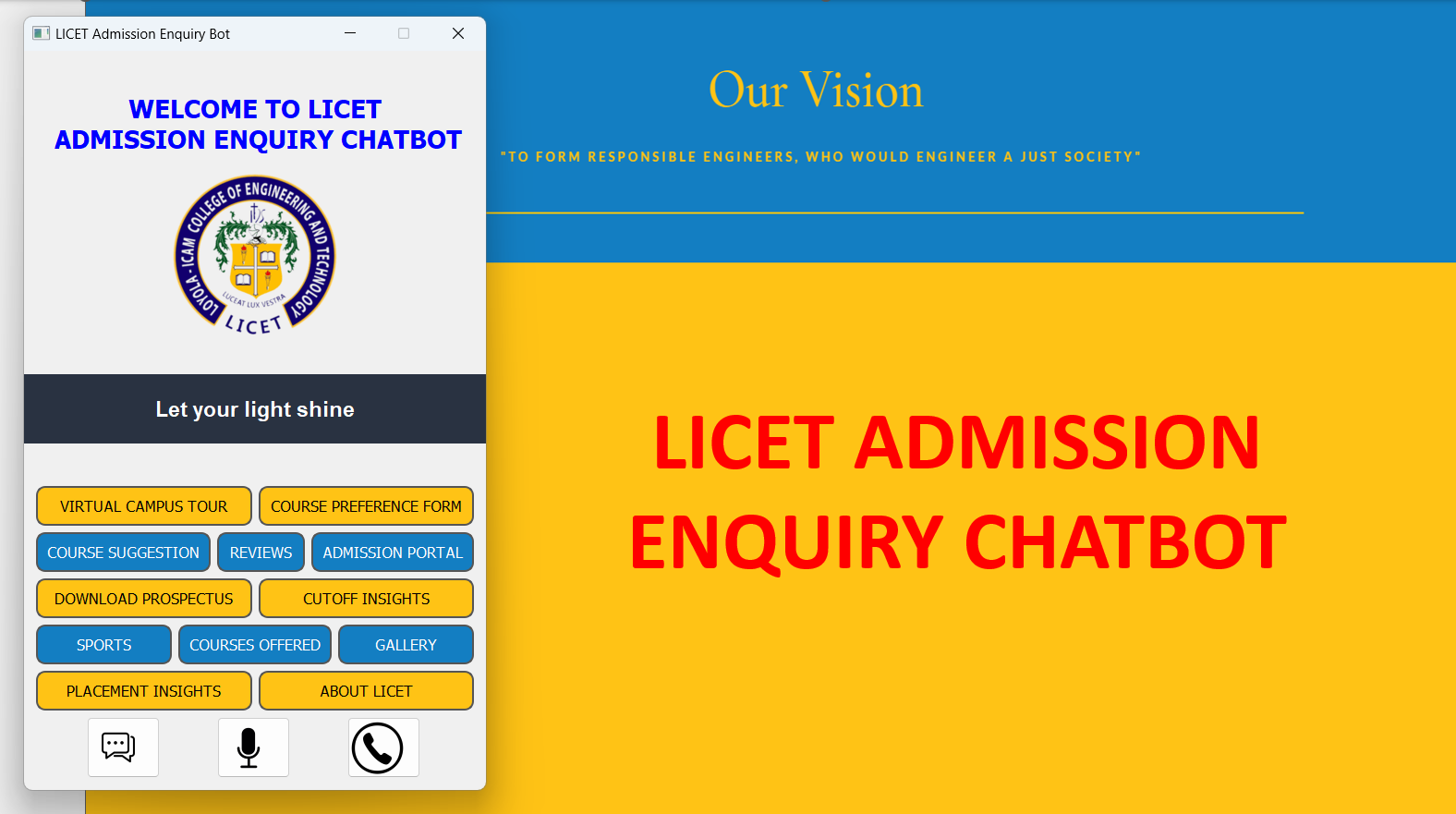
app = QtWidgets.QApplication([])

widget = CollegeAdmissionChatbot()

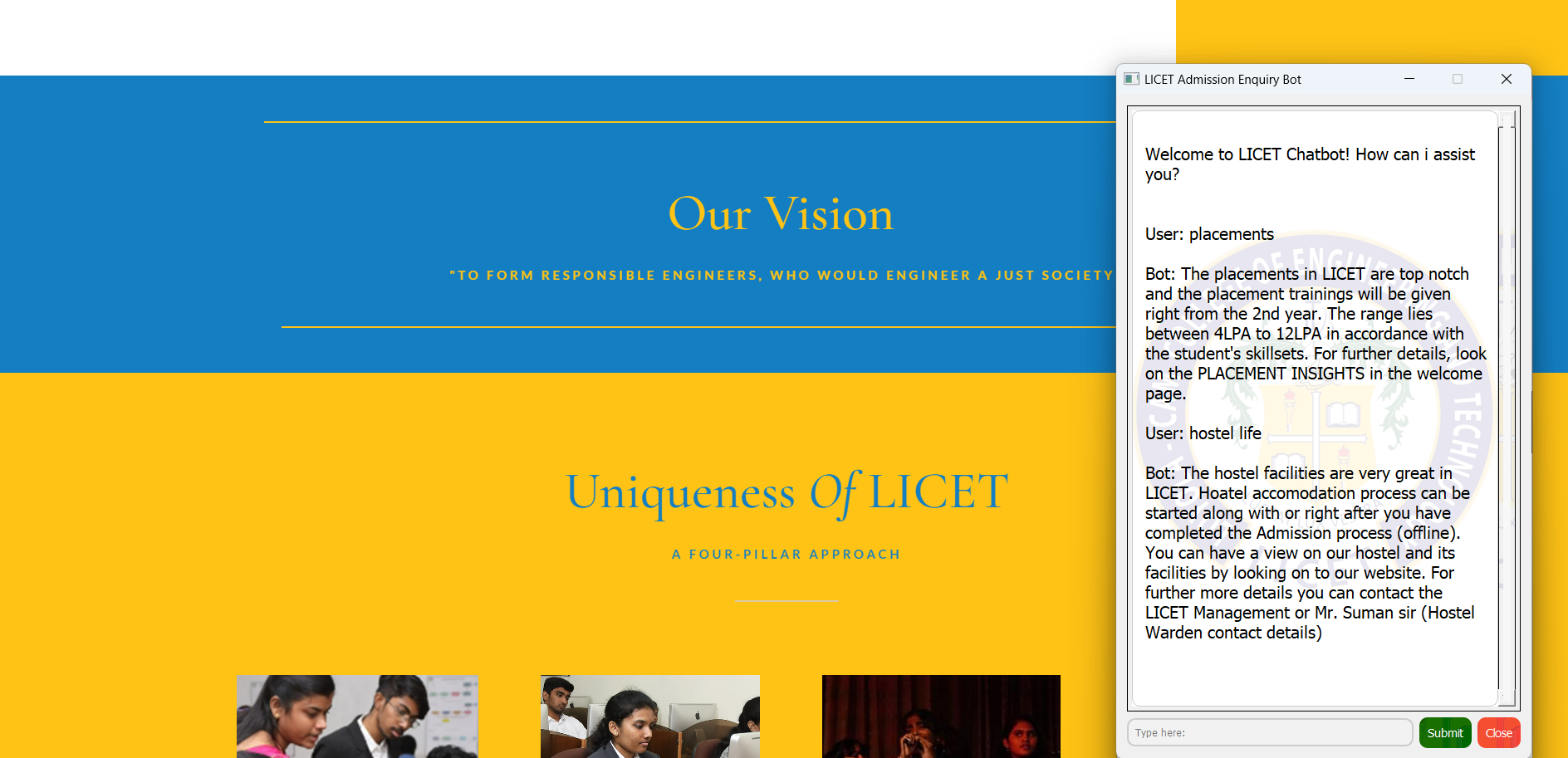
widget.run()

app.exec\_()

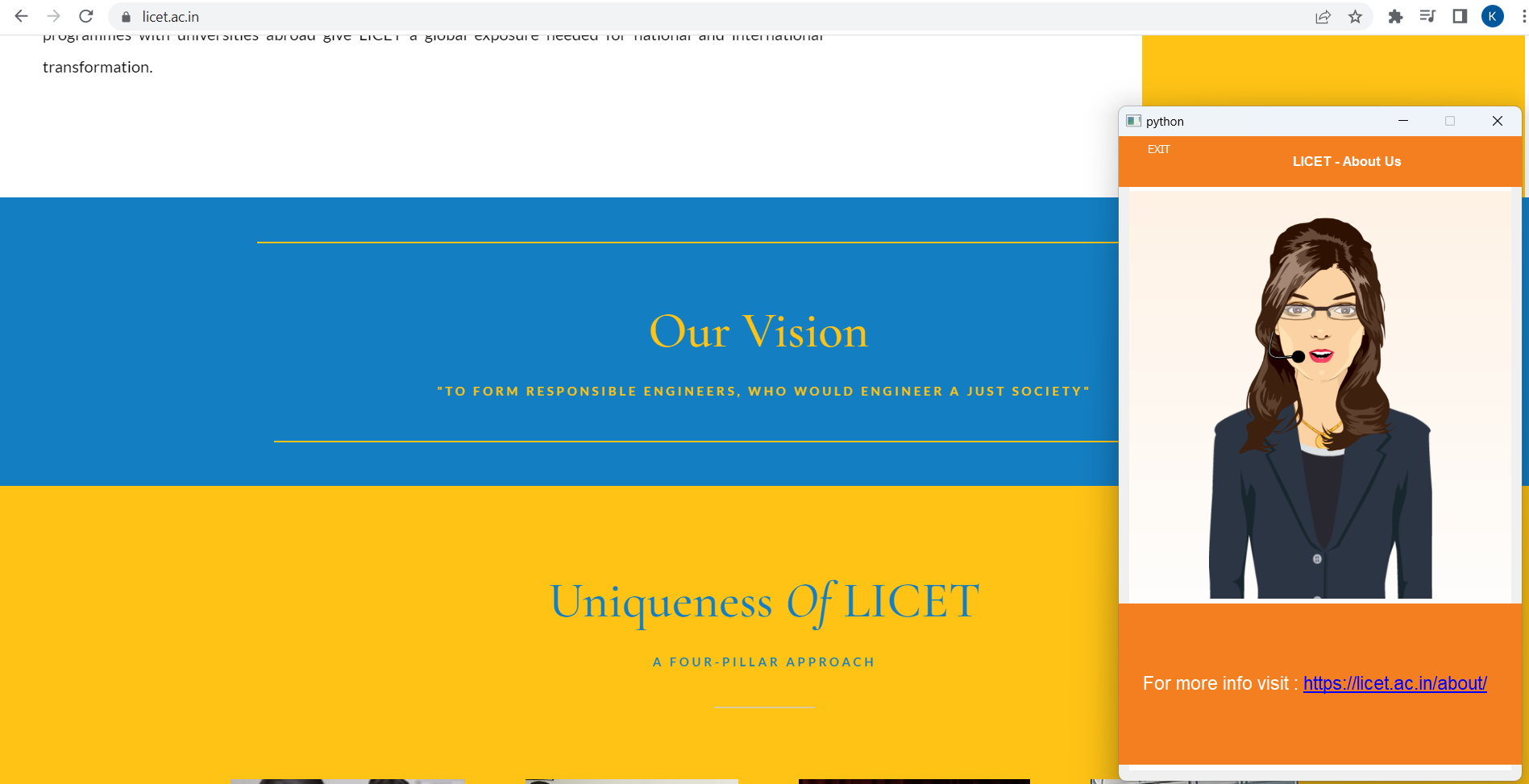
**APPENDIX B**

**

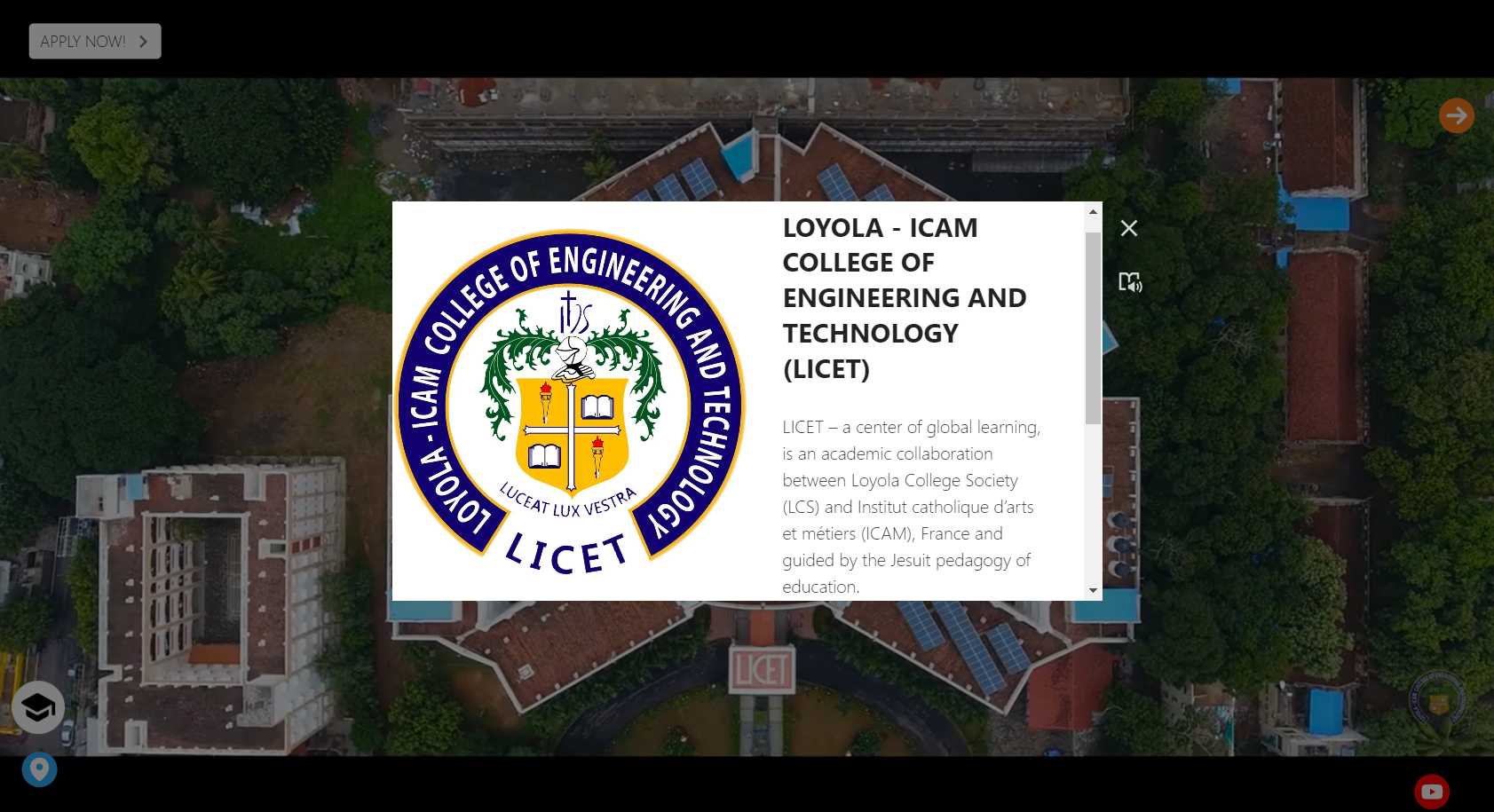
*Fig:B.1 Snapshot of the Chatbot User Interface*

**

*Fig:B.2 Snapshot of the Chat*

**

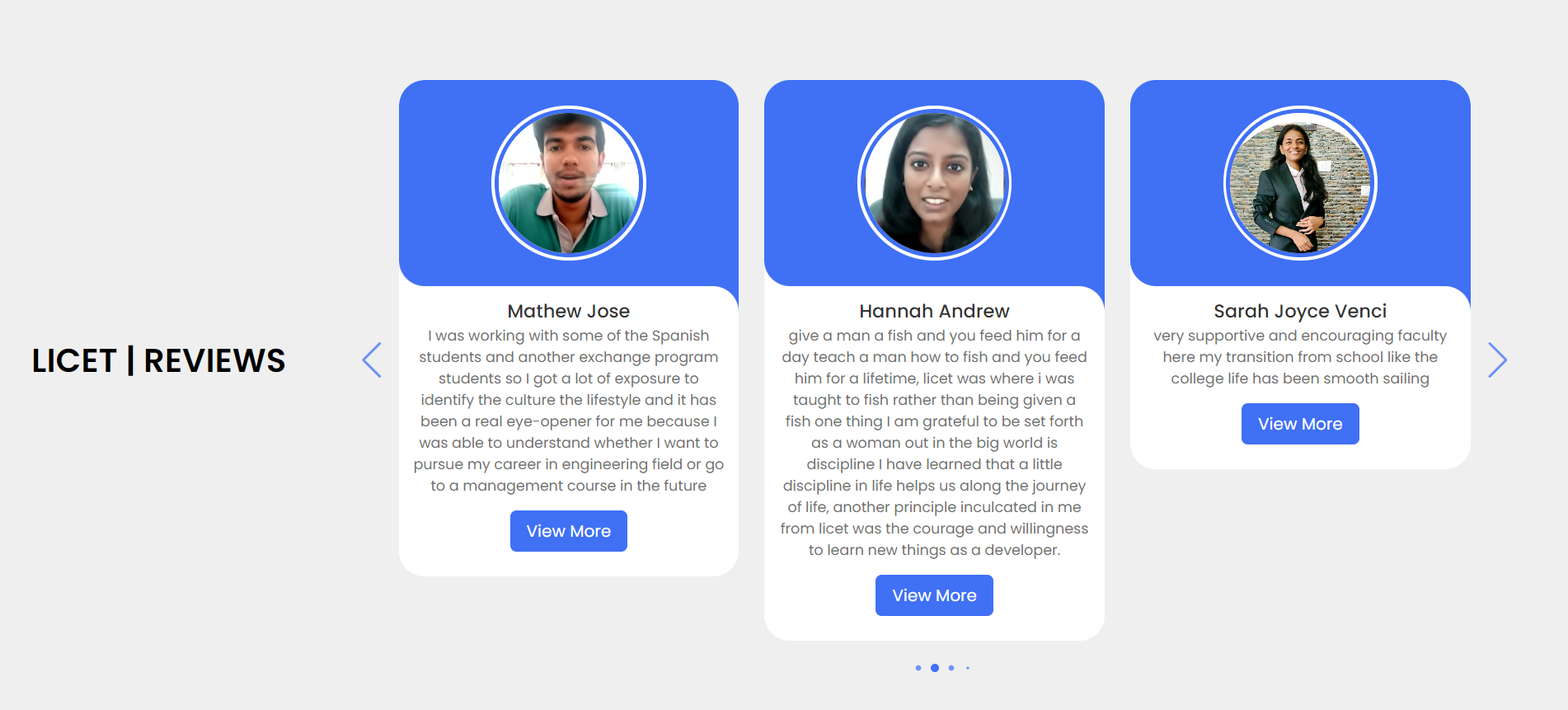
*Fig: B.3 Snapshot of the Speaking avatar in ABOUT section*

**

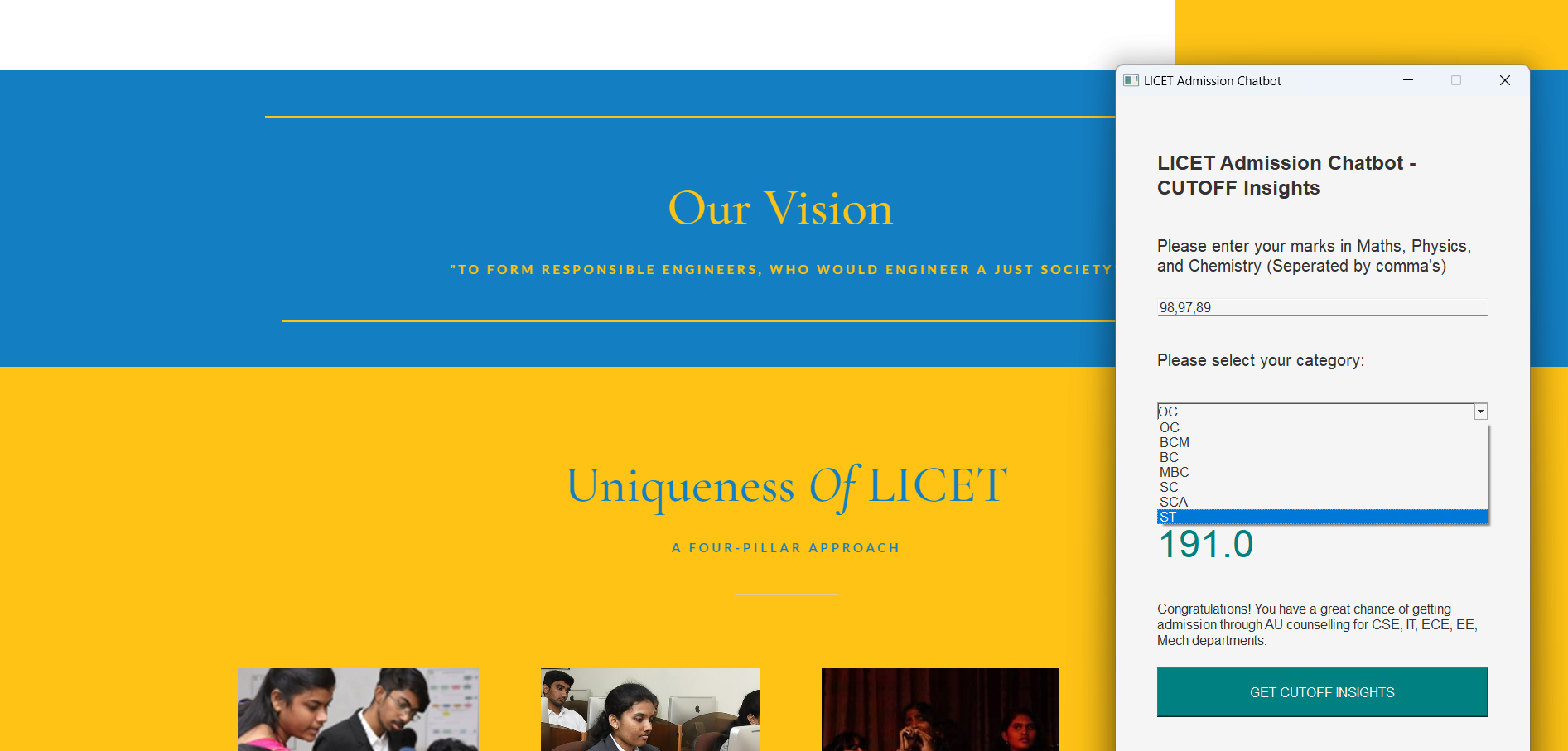
*Fig: B.4 Snapshot of the Outside Campus Tour*

**

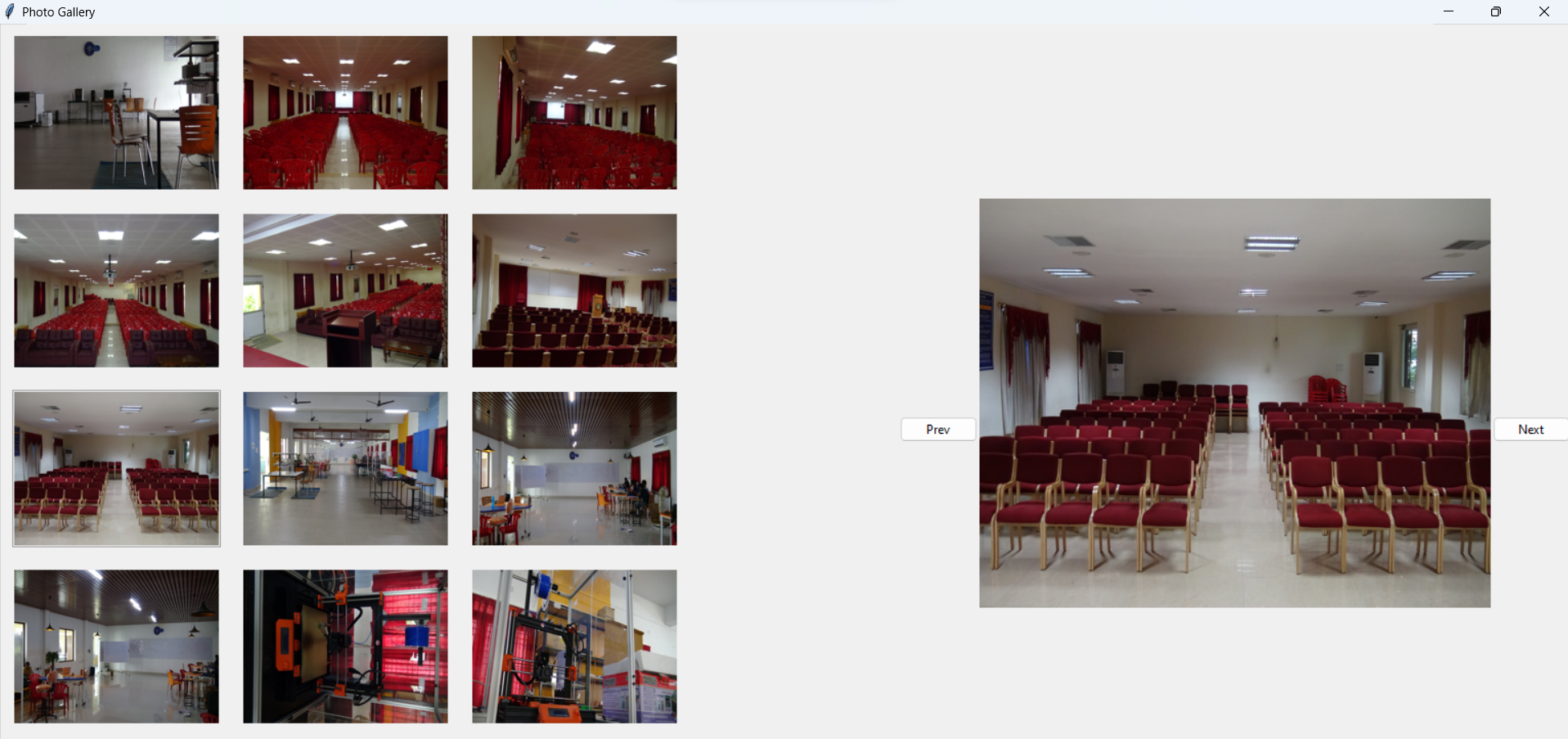
*Fig: B.5 Snapshot of the Virtual Campus Tour (Inside Campus)*

**

*Fig: B.6 Snapshot of the Alumni Review*

**

*Fig: B.7 Snapshot of the CutOff Insights*

**

*Fig: B.8 Snapshot of the Gallery*

# REFERENCES

[1] Yogi Wisesa Chandraa, Suyanto Suyantoa, "Indonesian Chatbot of University Admission Using a Question Answering System Based on Sequence-to-Sequence Model," ELSEVIER, 2021.

[2] CK. Gomathy, P.L.S.K Meghana, P. Vishnu Vardhan Reddy, "Voice based University Information Chatbot System," INTERNATIONAL RESEARCH JOURNAL OF ENGINEERING AND TECHNOLOGY (IRJET), 2023.

[3] Tarun Lalwani, Shashank Bhalotia, Ashish Pal, Shreya Bisen, Vasundhara Rathod, "Implementation of a Chatbot System using AI and NLP," International Journal of Advanced Computer Science and Applications (IJACSA), 2022.

[4] H.N.Khan, N. A. B. Mat Salleh, and M. R. Mahmud, "An Intelligent Chatbot for College Admission Counseling," International Journal of Advanced Computer Science and Applications (IJACSA), 2020.

[5] K. Jwala, G.N.V.G Sirisha, G.V. Padma Raju, "Developing a Chatbot using Machine Learning," International Journal of Recent Technology and Engineering (IJRTE), 2019.

[6] R. Saranya and S. Jaya, "A Review on Chatbot Systems using Natural Language Processing Techniques," International Journal of Computer Applications, vol. 179, no. 42, pp. 1-5, 2018.

[7] A. N. El-Sisi, E. A. Salama, and M. A. Salama, "A Chatbot-based Intelligent Tutoring System for e-Learning Environments," International Journal of Advanced Computer Science and Applications, vol. 10, no. 2, pp. 379-386, 2019.

[8] S. K. Dubey, M. N. Raju, and K. M. P. Kumari, "Development of a Chatbot for Higher Education Admission Process," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 9, no. 8, pp. 203-208, 2019.

[9] M. Rathore and S. Gupta, "Development of an Intelligent Chatbot for University Admission Guidance," International Journal of Computer Science and Mobile Computing, vol. 8, no. 6, pp. 1-7, 2019.

[10] T. S. Dixit and D. M. Thakore, "An AI-based Chatbot for College Admission Process," International Journal of Advanced Research in Computer Science, vol. 9, no. 1, pp. 123-128, 2018.