**Educational Counseling Chatbot**

Technical Documentation

By:

E223739 - Akila Dilan Fernando

EEEEEEE - Sheleena

Table of Contents

[**Project Summary 3**](#_heading=)

[**Research 3**](#_heading=h.leqqcfje1ugh)

[Performance 3](#_heading=h.o38cd3i7bdeh)

[Environment 4](#_heading=h.2pkvdmt46ds9)

[Actuators 4](#_heading=h.9zumh2g1v6kz)

[Sensors 5](#_heading=h.dkc73j4574ts)

[Summary 5](#_heading=h.ovdm2u66b5q)

[Architecture 5](#_heading=h.gjdgxs)

[Implementation Details 7](#_heading=h.9gc79ie9z7go)

[Example Workflow 8](#_heading=h.okx6zw4qdwhd)

# Project Summary

*Clear description of the product*

AI Powered Virtual Educational Consultant Chatbot for educational institutions.

*include following:*

1. *problem description*

From schools to universities, educational establishments have many difficulties in giving prompt, individualized advice to students, staff, and potential applicants. Managing questions regarding classes, admissions, campus amenities, and academic support, as well as providing individualized career counseling and academic planning, are some of these obstacles. The sheer volume of requests and the demand for individualized guidance frequently put a strain on educational institutions' resources, resulting in delays, misunderstandings, and an unsatisfactory user experience.

1. *a brief description*
2. *research and challenges faced*
3. *UML Diagrams (e.g. state transition diagrams, class diagrams), a description of how the Artifact works (possibly aided by diagram/screen shots/photos)*

[space for Running Application Screenshot x 3 - 1and half pages.]

# Research

Selecting programming language : python

Natural Language Interface Engine : considering NLIs such as TextBlob, spaCy and NLTK i found that NLTK is easy to use and has the most capability for python.

Decided to use scikit-learn as it’s go well with NLTK and python again and it’s ease of use.

Using Command line interface for the PoC and anyways there is capability to implement Graphical User Interface

P.E.A.S

Description of **P.E.A.S (Performance, Environment, Actuators, Sensors)** of the Bot

### **Performance**

**Performance** refers to how well the chatbot accomplishes its goal of providing educational counseling.

* **Accuracy**: The bot should provide correct and relevant answers to user queries about courses, admissions, schedules, etc.
* **Learning Capability**: The bot should improve its responses over time by learning from user interactions through the machine learning component.
* **Responsiveness**: The bot should provide responses quickly to ensure a smooth user experience.
* **User Satisfaction**: The overall user experience should be positive, with users finding the bot helpful and easy to interact with.

### **Environment**

**Environment** refers to the context in which the chatbot operates.

* **User Interface**: Currently, the bot operates via a Command Line Interface (CLI), with potential for future expansion to a Graphical User Interface (GUI) or web interface.
* **User Base**: The bot is used by students, prospective students, and other individuals seeking information about educational programs and courses.
* **Operational Environment**: The bot runs on a server or local machine with access to the necessary libraries (NLTK, scikit-learn) and the SQLite database.
* **Integration**: The bot may need to integrate with other systems or databases in the future to fetch or update information dynamically.

### **Actuators**

**Actuators** refer to the mechanisms the bot uses to perform actions in its environment.

* **Text Output**: The primary actuator is the text-based response system that outputs responses to the user's queries.
* **Database Updates**: The bot can update its knowledge base by adding new information to the SQLite database when new data is received.
* **Logging**: The bot logs user interactions in the database, which can be used to retrain the machine learning model.

### **Sensors**

**Sensors** refer to the mechanisms the bot uses to receive input from its environment.

* **User Input**: The bot receives text input from users through the CLI.
* **Tokenization and Parsing**: Using NLTK, the bot processes and understands the text input by tokenizing and parsing it.
* **Database Queries**: The bot retrieves information from the SQLite database based on the user input.
* **Machine Learning Feedback**: The bot uses the feedback loop of logged user interactions to improve its machine learning model.

### **Summary**

The Educational Counseling Chatbot's P.E.A.S. framework can be summarized as follows:

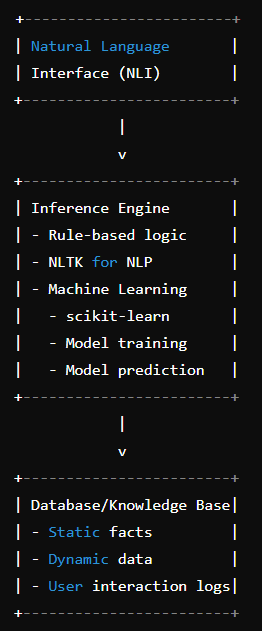
* **Performance**: Accuracy, learning capability, responsiveness, user satisfaction.
* **Environment**: Command Line Interface (CLI), potential for GUI, user base (students and prospective students), server/local machine, integration with other systems.
* **Actuators**: Text output, database updates, logging interactions.
* **Sensors**: User text input, tokenization and parsing with NLTK, database queries, machine learning feedback.

This P.E.A.S. framework helps in understanding how the chatbot interacts with its environment, processes input, and performs actions to achieve its goal of providing educational counseling.

# Architecture

High Level Architecture

(draw a flowchart)



+------------------------+

| Natural Language |

| Interface (NLI) |

+------------------------+

|

v

+------------------------+

| Inference Engine |

| - Rule-based logic |

| - NLTK for NLP |

| - Machine Learning |

| - scikit-learn |

| - Model training |

| - Model prediction |

+------------------------+

|

v

+------------------------+

| Database/Knowledge Base|

| - Static facts |

| - Dynamic data |

| - User interaction logs|

+------------------------+

### **Implementation Details**

#### **Natural Language Interface (NLI)**

We are using a command-line interface for initial prototyping.

#### **Inference Engine**

1. **Rule-based Logic**: Handle straightforward mappings.
2. **NLTK for NLP**: Tokenize, parse, and analyze user input.
3. **Machine Learning with scikit-learn**:
   * **Model Training**: Train the model using logged user interactions.
   * **Model Prediction**: Predict responses based on user input.
4. **Database Queries**: Retrieve and update information in the database.

#### **Database/Knowledge Base**

We'll use SQLite for simplicity, storing information about courses, admissions, schedules, and user interaction logs.

### **Example Workflow**

#### **Step 1: User Input**

User: "What courses do you offer?"

#### **Step 2: Inference Engine**

* **NLTK for NLP**: Tokenize, parse, and analyze user input.
* **Rule-based logic**: Recognize keywords like "courses" and "offer".
* **Machine Learning**: Predict responses using a trained model.
* **Database Queries**: Retrieve information from the database.

#### **Step 3: Database/Knowledge Base Query**

Query the database for available courses.

#### **Step 4: Response**

Respond to the user with the list of courses.

Source Code

<https://github.com/akdev96/Educational-Counselling-ChatBot>

Test Plan

#### **1. Introduction**

The test plan outlines the approach, scope, and objectives for testing the Educational Counseling Chatbot. The purpose is to ensure that the chatbot functions correctly, provides accurate information, and delivers a satisfactory user experience.

#### **2. Scope**

The test plan covers functional and non-functional testing of the Educational Counseling Chatbot. It includes testing of features such as natural language processing (NLP), machine learning (ML), database interactions, and user interface (CLI).

#### **3. Objectives**

* Verify the accuracy of responses provided by the chatbot.
* Ensure the chatbot handles various types of user queries effectively.
* Validate the performance and responsiveness of the chatbot.
* Confirm that the chatbot updates its knowledge base appropriately based on user interactions.
* Assess the usability and user experience of the Command Line Interface (CLI).

#### **4. Testing Approach**

The testing approach includes:

* **Unit Testing**: Test individual components such as NLP functions, database queries, and ML model predictions. Using python [unittest](https://docs.python.org/3/library/unittest.html)
* **Integration Testing**: Test interactions between different components (NLP, ML, database).
* **System Testing**: Test the chatbot as a whole to ensure it meets functional and non-functional requirements.
* **User Acceptance Testing (UAT)**: Gather feedback from real users to evaluate usability and effectiveness this step has been skipped due to no real users.

#### **5. Test Scenarios**

##### **5.1 Functional Testing**

**Scenario 1: Basic Functionality**

* **Test Case 1.1**: Input: "Hello" - Expected Output: "Hi there! How can I assist you today?"
* **Test Case 1.2**: Input: "What courses do you offer?" - Expected Output: List of courses retrieved from the database.

**Scenario 2: Natural Language Processing**

* **Test Case 2.1**: Input: "I want to study computer science" - Expected Output: Relevant information about computer science programs.
* **Test Case 2.2**: Input: "Tell me about the admission process" - Expected Output: Explanation of the admission process retrieved from the database.

**Scenario 3: Machine Learning**

* **Test Case 3.1**: Input: Frequently asked questions - Expected Output: Responses learned and adapted by the ML model based on historical data.
* **Test Case 3.2**: Input: Queries with nuanced language - Expected Output: Correct interpretation and relevant responses generated by the ML model.

##### **5.2 Non-Functional Testing**

**Scenario 4: Performance Testing**

* **Test Case 4.1**: Measure response time for typical queries to ensure responses are prompt (within acceptable limits).

**Scenario 5: Usability Testing**

* **Test Case 5.1**: Evaluate the ease of use of the CLI interface.
* **Test Case 5.2**: Assess the clarity and helpfulness of responses provided by the chatbot.

#### **6. Tools and Resources**

* **NLTK**: Used for natural language processing tasks.
* **scikit-learn**: Utilized for machine learning model training and prediction.
* **SQLite**: Database for storing course information and user interactions.
* **Python unittest framework**: For writing and executing unit tests.

#### **7. Test Environment**

* Operating System: Windows 10 / macOS / Linux
* Python Environment: Python 3.x
* Dependencies: NLTK, scikit-learn, SQLite

#### **8. Test Execution**

* **Unit Tests**: Automated tests using unittest framework to validate individual functions and components.
* **Integration Tests**: Conducted manually to verify interactions between NLP, ML, and database components.
* **System Tests**: Executed in a controlled environment to ensure the chatbot meets functional and non-functional requirements.
* **User Acceptance Tests**: Real-world usage scenarios tested with actual users to gather feedback on usability and effectiveness.

#### **9. Test Deliverables**

* Test Plan Document
* Test Cases Document
* Test Execution Report

#### **10. Risks and Mitigation**

* **Risk**: Unpredictable user queries that the chatbot may not handle correctly.
  + **Mitigation**: Continuous improvement through user feedback and iterative updates to the ML model.
* **Risk**: Performance degradation under heavy load or with large datasets.
  + **Mitigation**: Monitor and optimize performance metrics; scale resources if necessary.

The test plan outlines comprehensive testing strategies to ensure the Educational Counseling Chatbot functions accurately, responsively, and effectively meets user expectations. By rigorously testing functional, non-functional, and usability aspects, we aim to deliver a reliable and user-friendly chatbot for educational counseling purposes.

**Test data,**

Test data to be created manually based on common question and answers based on database

Conclusion

The Educational Counseling Chatbot is a sophisticated application designed to assist students and prospective students by providing accurate and timely information about courses, admissions, and other educational queries. By leveraging natural language processing (NLP) with NLTK and machine learning with scikit-learn, the chatbot can understand user input and improve its responses over time through continuous learning.

The architecture of the chatbot ensures a smooth and efficient user experience. The Command Line Interface (CLI) serves as a practical proof of concept, demonstrating the bot's core functionalities and its potential for future expansion to a more user-friendly Graphical User Interface (GUI). The use of SQLite as the database allows for easy storage and retrieval of dynamic and static information, while also maintaining a log of user interactions to train the machine learning model.

The P.E.A.S. (Performance, Environment, Actuators, Sensors) framework helps in understanding the bot's capabilities and interactions within its operational environment. The chatbot is designed to be accurate, responsive, and capable of learning from user interactions, ensuring a high level of user satisfaction.

Overall, this chatbot represents a significant step forward in utilizing artificial intelligence to provide educational counseling. It demonstrates how modern technologies can be harnessed to create intelligent systems that enhance the user experience by offering relevant and personalized information. As the chatbot continues to learn and evolve, it promises to become an even more valuable tool for educational institutions and their students.

References

Python 3.8 - <https://www.python.org/downloads/release/python-380/>

NLTK - (Natural Languate ToolKit) - <https://www.nltk.org/>

scikit-learn - <https://scikit-learn.org/stable/>

NumPy - <https://numpy.org/>

How to build a Simple Chatbot with Python and NLTK - <https://medium.com/@lurkinguard30/how-to-build-a-simple-chatbot-with-python-and-nltk-c1b9fff5f3>

ryuzaki\_bot - <https://github.com/LuciaLlavero/ryuzaki_bot>

NLP - Natural Language Processing - <https://en.wikipedia.org/wiki/Natural_language_processing>

From Requirement

**MAINSIT COURSEWORK 2**

**Introduction to AI – Assessment 2023/24**

1. **Chat Bot**

**Marking Schmes for Assignment 2 (75%)**

**MARKING SCHEME FOR CHAT BOT**

| **Professionalism (50%)** | Choice of a suitable application (e.g. banking, travel & tourism, COVID 19 Support Centre) | 2 |
| --- | --- | --- |
|  | Effective Implementation of Natural Language Processing (NLP) via a program/inference engine | 25 |
|  | Use of persistent storage (e.g. relational database, text files) to store facts (knowledge base) | 5 |
|  | An Indication of machine learning (i.e. chat bot updating its own knowledge base) | 8 |
|  | Text based natural language interface | 4 |
|  | Overall functionality, quality and appearance | 6 |
| **Technical Documentation (20%)** | Clear description of the product | 2 |
|  | Research | 4 |
|  | Design architecture (i.e. main components of the Bot) | 2 |
|  | Description of **P.E.A.S (Performance, Environment, Actuators, Sensors)** of the Bot | 2 |
|  | **Key design ideas** supported by **code snippets** ( **lemmatizing, small talk, random answers, getting database answers, training the Bot** **etc.**) | 4 |
|  | Algorithms (e.g. flow charts, pseudo codes) | 2 |
|  | Fully annotated source code listing | 2 |
|  | Test data and results | 1 |
|  | Conclusion + references | 1 |
| **Communication skills (20%)** | Facts presented clearly and coherently | 8 |
|  | Effective demonstration of the product | 8 |
|  | Overall quality of the Viva | 4 |
| **Creativity (10%)** | Originality of ideas/concepts | 5 |
|  | Artifact is attractive and is good enough to showcase | 5 |