INTERNATIONAL ISLAMIC UNIVERSITY CHITTAGONG

Course Title : Artificial Intelligence (LAB)

Course Code: CSE-3636

Artificial Intelligence Lab Project Report OCD Support Chatbot

Submission Date: 09/07/2025

Submitted By:	**Submitted To:**
Shumana Barua Nitu (C223304)	Sara Karim
Nusaiba Ibnat (C223282)	Adjunct Faculty
Jumana Muhammad Harun (C223271)	Computer Science and Engineering (IIUC)
Semester - 6th (6CF)	

Index

Artificial Intelligence Lab Project Report
OCD Support Chatbot

1. Introduction

Mental health disorders, particularly Obsessive Compulsive Disorder (OCD), often go undiagnosed due to social stigma, lack of awareness, and limited access to mental health professionals. With the increasing integration of artificial intelligence (AI) in healthcare, this project aims to bridge that gap by developing a lightweight, accessible, and intelligent solution. This project introduces a Flask-based AI-powered web application that functions as a virtual assistant to detect signs of OCD through user-generated text inputs. By leveraging Natural Language Processing (NLP) and Machine Learning (ML) techniques, the system can analyze textual responses and classify them as either OCD-related or non-OCD. This classification is made possible through a trained machine learning model based on real, labeled datasets of statements reflecting both obsessive-compulsive behavior and normal thought patterns. The primary motivation behind this application is to provide a non-invasive, privacy-conscious, and empathetic platform where users can express their thoughts and feelings. The chatbot interacts with users in a conversational manner and, when OCD tendencies are detected, gently offers evidence-based coping strategies such as grounding techniques. This ensures the application not only identifies potential symptoms but also contributes toward managing them in real time. The system architecture is composed of two main components: a backend powered by Flask that processes user messages and interacts with the machine learning model, and a frontend interface built with HTML, CSS, and JavaScript that ensures smooth, responsive user interaction. The model itself is trained using TF-IDF vectorization and a Logistic Regression algorithm, and the entire workflow from user input to therapeutic response is designed to be both effective and empathetic.

2. Objective

The primary objective of this project is to design and develop an AI-powered web-based application that can intelligently detect Obsessive-Compulsive Disorder (OCD) tendencies through the analysis of user-written text. In an age where mental health awareness is growing yet access to personalized care remains limited for many, this project seeks to offer a lightweight, accessible, and non-intrusive digital tool that empowers individuals to better understand their own mental health. By combining the power of Natural Language Processing (NLP) with Machine Learning (ML), the system aims to classify user messages as OCD-related or non-OCD-related in real time. The tool is designed to simulate empathetic conversations, detect linguistic patterns commonly associated with obsessive or compulsive thoughts, and when such patterns are identified offer supportive suggestions and evidence-based coping strategies. Another key objective is to maintain user privacy and simplicity. The platform requires no login or personal data, allowing users to interact freely without fear of judgment or exposure. It is built with minimal system requirements to ensure accessibility across a wide range of devices, especially in resource-limited settings.

Furthermore, the project seeks to:

- Demonstrate the practical application of ML models (Logistic Regression) for binary text classification.
- Build an intelligent chatbot capable of engaging in supportive dialogue.
- Incorporate psychological grounding techniques to support users showing signs of anxiety or compulsive behaviors.
- Serve as a proof of concept for future expansion into more complex mental health detection systems.

Ultimately, this project is not intended to diagnose or replace professional care, but rather to act as a first layer of support an approachable and intelligent system that listens, understands, and gently guides users toward healthier thought patterns

3. Tools & Technologies Used

To develop the OCD Support Chatbot, a variety of modern programming languages, libraries, and frameworks were employed to ensure a robust, scalable, and responsive system. Each tool was selected for its relevance and contribution to the core functionality of the application, particularly in machine learning, natural language processing, data management, and web deployment.

Python: Python is the primary programming language used in this project. Its simplicity, readability, and vast collection of libraries make it ideal for machine learning and web development tasks. Python's extensive ecosystem supports rapid prototyping and seamless integration of AI models with web applications.

Flask: Flask is a lightweight and flexible web application framework used to build the backend server. It allows for the creation of RESTful routes and handles HTTP requests efficiently. Flask serves the frontend HTML interface, connects to the machine learning model, and returns chatbot responses in real time.

Scikit-learn: scikit-learn is a powerful and widely used machine learning library in Python. It was used for:

- Text classification using the Logistic Regression algorithm.

- Vectorization of text using TF-IDF.
- Splitting datasets and evaluating model performance (e.g., confusion matrix, accuracy score).
  Its simplicity and performance make it ideal for deploying models in production-level systems.

Pandas: Pandas is a Python library designed for data manipulation and analysis. It was used to:

- Load and explore the custom OCD dataset (.csv file).
- Preprocess and clean textual data.
- Organize training data into structured formats required by scikit-learn.

nltk: nltk is a key library used for Natural Language Processing (NLP). It helps preprocess user-generated text through:

- Tokenization (splitting text into words).
- Stop-word removal (removing irrelevant words like "and", "is", "the").

joblib: joblib is used to serialize and save Python objects, particularly the trained machine learning model (ocd_model.pkl) and the TF-IDF vectorizer (vectorizer.pkl). These files are then loaded during runtime to make real-time predictions without retraining the model.

HTML: HTML is the frontend structure language used to build the chatbot interface. It defines the layout, input fields, and message display areas in the browser. Combined with CSS and JavaScript (Fetch API), it allows users to interact with the bot smoothly and intuitively.

Table 1: Summary of Tools

Tool/Technology	**Purpose**
Python	Core programming for ML and backend logic
Flask	Web server and HTTP request handling
scikit-learn	Machine learning model training and evaluation
pandas	Dataset handling and preprocessing
nltk	Natural language text cleaning and tokenization
joblib	Saving/loading trained ML models
HTML	Designing the chatbot interface on the frontend

4. Datasets

The core functionality of the OCD Support Chatbot depends on the quality and structure of the dataset used to train the machine learning model. For this project, a custom dataset titled ocd_dataset_with_nonocd.csv was created and utilized. This dataset contains labeled text entries that reflect various psychological expressions, classified as either OCD-related (label: 0) or non-OCD (label: 1).

Structure of the Dataset:

Each row in the dataset contains:

- A text statement written in natural language (e.g., "I need to wash my hands again; they don't feel clean enough.")
- A corresponding label:
  - 0 = OCD-related expression
  - 1 = Non-OCD expression

Preprocessing Steps:

- Before using the dataset to train the model, several preprocessing tasks were performed:
- Lowercasing: All text was converted to lowercase to ensure uniformity.
- Punctuation removal: Unnecessary characters were removed to reduce noise.
- Tokenization: Each sentence was split into words.
- Stop-word removal: Common but non-informative words (like "is", "the", "and") were removed using nltk.
- TF-IDF Vectorization: Text was transformed into numerical feature vectors using Term Frequency–Inverse Document Frequency (TF-IDF). This helped capture the importance of words in each sentence relative to the entire dataset.

Sample Entries from Dataset:

Table 2: Sample Entries from Dataset

Text	Label
I need to check the door lock again.	0
I feel good after morning walks.	1
I keep counting things in my head.	0
I enjoy spending time with friends.	1

5. Step-by-Step Workflow

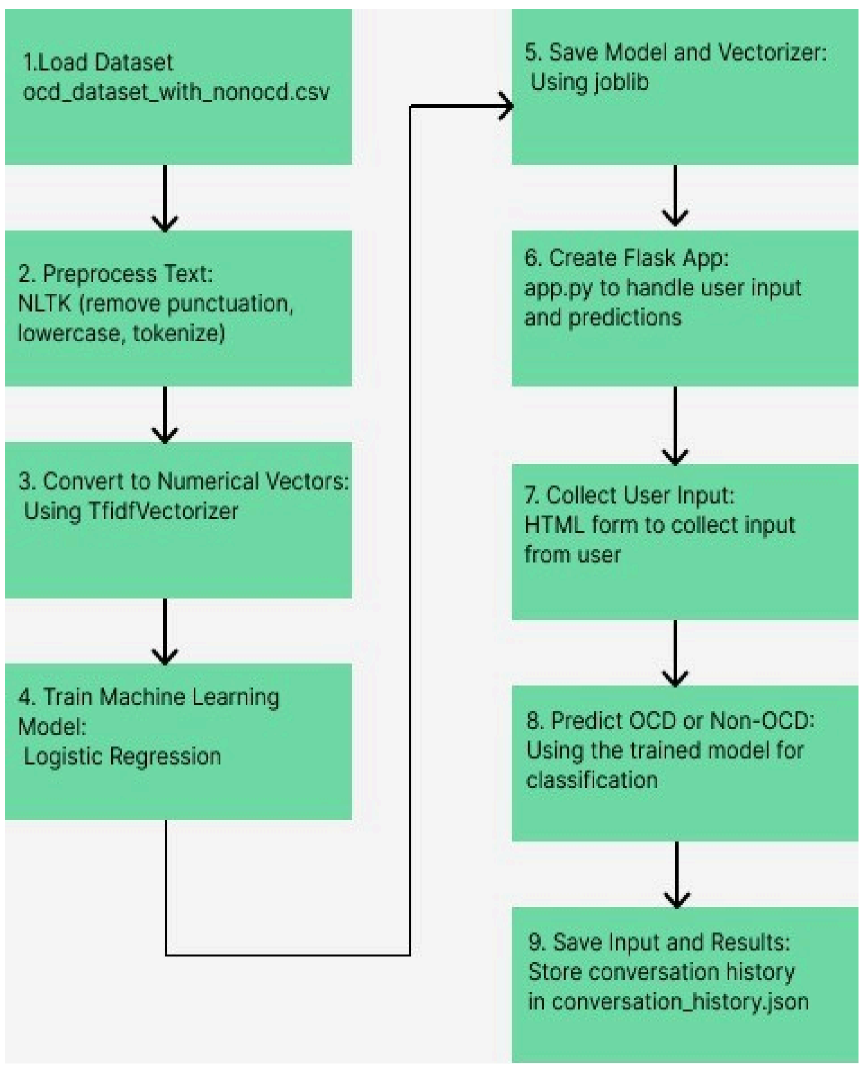


Figure 1: Flowchart of this project

Step 1: Load the dataset (`ocd_dataset_with_nonocd.csv`) using pandas.

Step 2: Preprocess text (remove punctuation, lowercase, tokenize) using NLTK.

Step 3: Convert text into numerical vectors using TfidfVectorizer.

Step 4: Train a machine learning model (e.g., Naive Bayes or SVM).

Step 5: Save the trained model and vectorizer as `ocd_model.pkl` and `vectorizer.pkl` using joblib.

Step 6: Create a Flask app (`app.py`) to load the model and vectorizer.

Step 7: Use a simple HTML form (`index.html`) to collect user input.

Step 8: Predict the result and display whether the input indicates OCD or not.

Step 9: Save input and results in `conversation_history.json` for record keeping.

6. Main Files Explained

Table 3: Summary Table of Key Files

File	Purpose
app.py	Main Flask backend, routing user input and displaying results.
train_model.py	Script to preprocess data, train, and save the model.
ocd_model.pkl	Saved machine learning model for predicting OCD vs non-OCD.
vectorizer.pkl	Saved TF-IDF vectorizer used for transforming user input into vectors.
ocd_dataset_with_nonocd.csv	Dataset containing OCD and non-OCD text samples used for model training.
index.html	Frontend HTML form for collecting user input and displaying predictions.
conversation_history.json	JSON file that stores user interactions and chatbot responses for analysis.

7. Backend Logic

The backend is responsible for processing the user's input, predicting if it’s related to OCD, and providing helpful responses. Here’s how it works:

- **Conversation Management:** The bot has different "states" it follows during the conversation, like greeting the user, listening to their issues, offering help, and suggesting coping activities when OCD is detected.
- **OCD Detection:** When the user types a message, the backend checks it using a **trained machine learning model** to see if it shows signs of OCD.
- **Storing Data:** Every conversation is saved in a JSON file so the bot can remember past interactions and improve future responses.

8. Frontend Features

The **frontend** is the part the user interacts with directly. Here’s how it’s built:

- **Chat Interface:** The bot has a **simple, easy-to-use chat window** where the user types their message and the bot responds.
- **Responsive Design:** The chat works well on both **phones and computers**—it adapts to different screen sizes.
- **Empathetic Responses:** The bot talks kindly, listens, and offers support when needed. It doesn’t diagnose but helps the user by suggesting coping techniques.

9. Example Conversation Flow

This section outlines an example conversation flow between the user and the chatbot. It demonstrates how the bot interacts with users and offers support in response to OCD related behavior.

User: Hello

Bot: Hello! How are you feeling today?

User: Not that good

Bot: Can you tell me more about what's bothering you?

User: Washing my hands again and again

Bot: That might be related to OCD. Would you like help with it?

User: Yes

Bot: Try the 5-4-3-2-1 Grounding Technique: Name 5 things you can see, 4 things you can touch, etc.

User: I feel a bit better

Bot: Great! How are you feeling now? Would you like more help?

User: No

Bot: Take care! I'm here if you need to talk more.

10. Evaluation Results

Confusion Matrix:

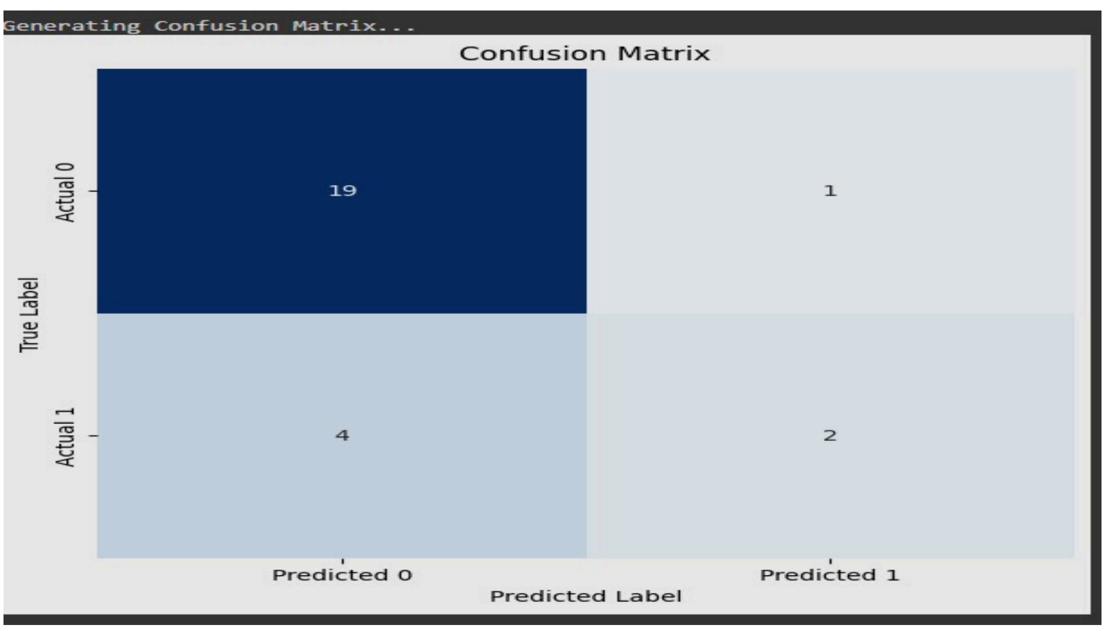


Figure 02: Confusion Matrix

The confusion matrix is an essential tool for evaluating the performance of a machine learning model, specifically for classification tasks. In this project, the confusion matrix shows how well the model is predicting whether a user's message indicates OCD-related behavior or non-OCD-related behavior.

- **Explanation:**
  - For Actual 0 (Non-OCD) messages, the model correctly predicted 19 instances, but 1 was incorrectly labeled as OCD (false positive).
  - For Actual 1 (OCD) messages, 2 were correctly identified as OCD, but 4 were misclassified as Non-OCD (false negatives).

- **Insight:**
  - The model performs well when detecting Non-OCD messages, as indicated by the high true negative rate.
  - However, it struggles with OCD detection, particularly with false negatives—OCD messages that are not correctly identified. This indicates that the model needs further training or improved data to better catch OCD-related symptoms.

Chatbot Interface and User Interaction Flow

Chatbot Interface – Initial View:

The chatbot has a **simple, dark-themed interface** designed to make users feel comfortable and relaxed while talking about sensitive topics like OCD.

- **Greeting Message**: The chatbot starts with a friendly message: "**Hello! I'm here to help with OCD-related concerns.**"
- **Conversation Layout**:
  - **Left Panel**: Shows past conversations, so users can keep track of what they've discussed.
  - **Right Panel**: Displays the current conversation, allowing users to focus on the latest messages.

Chatbot User Interaction – Step 1

In this step, the user begins the conversation, and the bot responds in a friendly and supportive way.

- **User:** "Hi"
- **Bot:** "What’s been on your mind?" (The bot encourages the user to share more.)
- **User:** "Facing some problems"
- **Bot:** "I’m here to listen. Can you tell me more about what’s bothering you?" (The bot listens empathetically and invites the user to elaborate.)

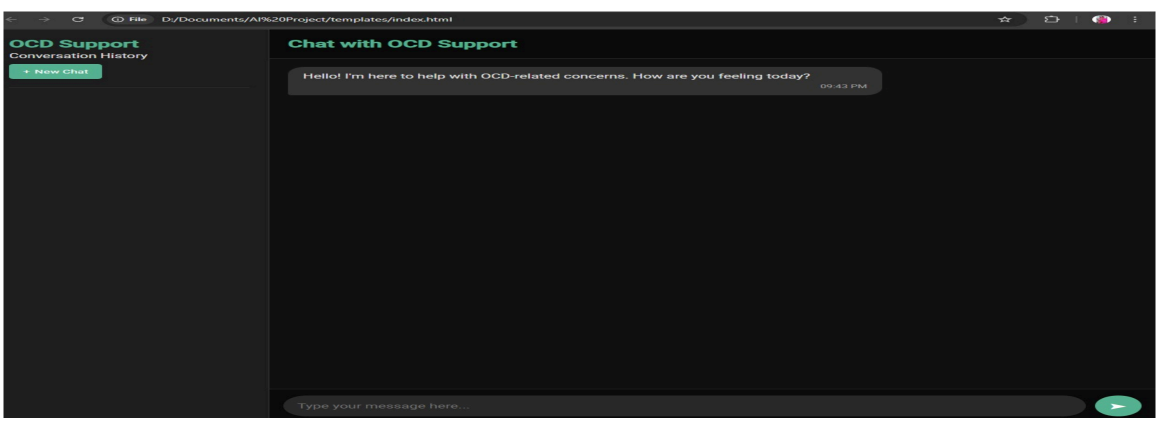
This flow shows how the chatbot **understands** the user’s input and responds accordingly, creating a **personal connection**.

Chatbot User Interaction – Step 2 (OCD Trigger Detected)

In this step, the chatbot detects a potential **OCD-related behavior** and offers support.

- **User:** "Washing my hands again and again"
- **Bot:** "From what you're describing, I notice some patterns that might relate to OCD. Would you like me to suggest a helpful activity?" (The bot identifies OCD-related behavior and offers help.)

The bot detects OCD-like behavior using its **machine learning model** and offers **coping activities**, such as grounding techniques, to help the user.



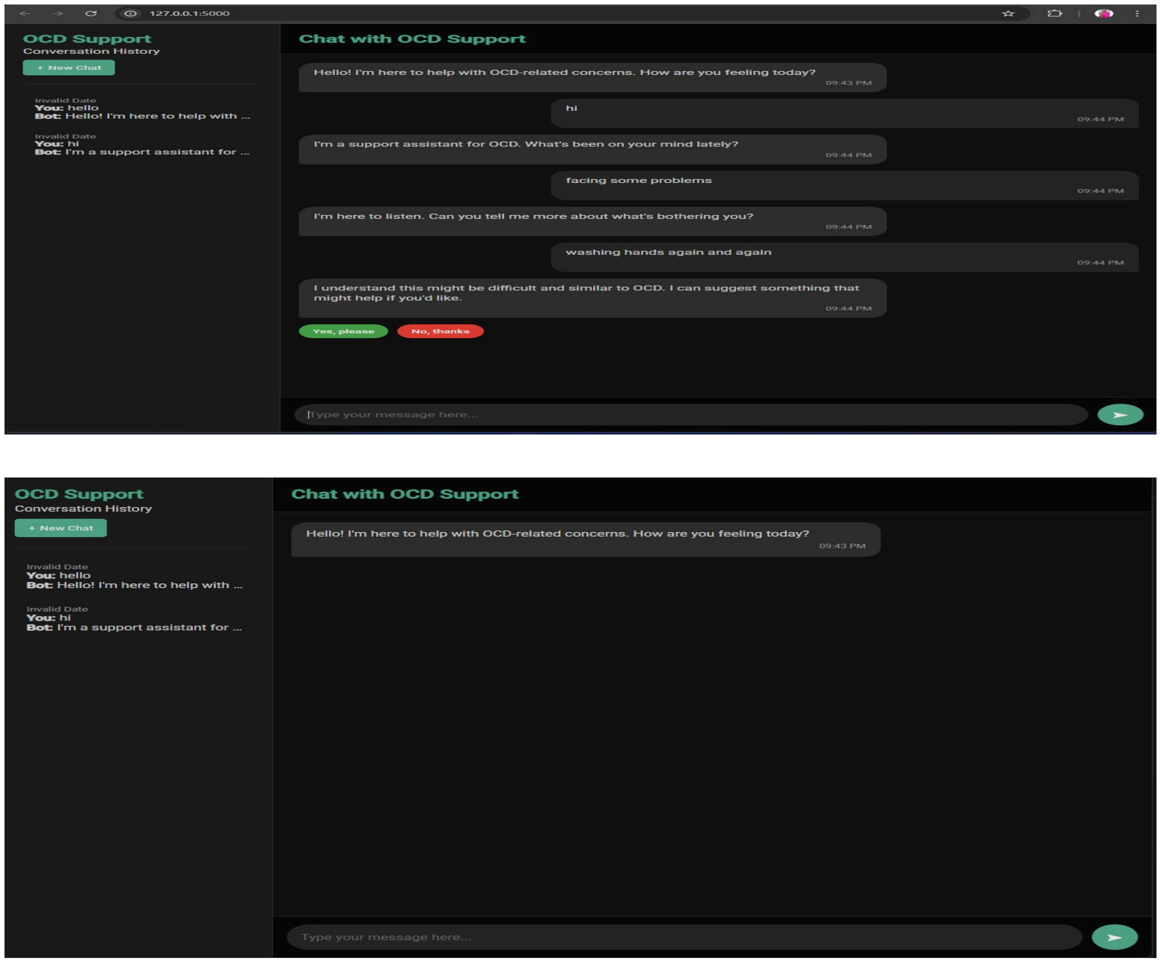


Figure 03: Chatbot Interface and User Interaction Flow

Classification Report:

The **classification report** provides a performance summary of the machine learning model based on key metrics such as **accuracy**, **precision**, **recall**, and **F1-score**.

- **Accuracy:**
    - The model has an **overall accuracy of approximately 81%**, meaning it correctly classifies the input most of the time.
- **Class 0 (Non-OCD):**
    - The model performs well with **high precision and recall**, indicating it is effective at identifying Non-OCD cases.

- **Class 1 (OCD):**
    - For OCD-related messages, the model shows **lower recall** (0.33), meaning it misses many actual OCD cases. This reflects that the model needs improvements in detecting OCD symptoms.
- **Macro Average/Weighted Average:**
    - These averages show the model's overall balance in performance between both classes, highlighting that improvements are needed for detecting OCD cases more accurately.

```
Test accuracy: 0.8076923076923077
              precision    recall  f1-score   support

           0       0.83      0.95      0.88        20
           1       0.67      0.33      0.44         6

    accuracy                           0.81        26
   macro avg       0.75      0.64      0.66        26
weighted avg       0.79      0.81      0.78        26

['vectorizer.pkl']
```

Figure 04: Classification result

Chatbot Suggests Grounding Activity:

Once the bot detects potential OCD-related behavior, it suggests a therapeutic activity to help manage the symptoms.

- **Activity Suggested:**
    - The chatbot offers the 5-4-3-2-1 Grounding Technique, a cognitive-behavioral therapy (CBT) method to help users manage anxiety by focusing on their surroundings:
        - 5 things you can see

- 4 things you can touch
- 3 things you can hear
- 2 things you can smell
- 1 thing you can taste

- **Therapeutic Value:**
    - This suggestion demonstrates the real-world mental health value of the chatbot, offering users practical, evidence-based tools to manage their symptoms.

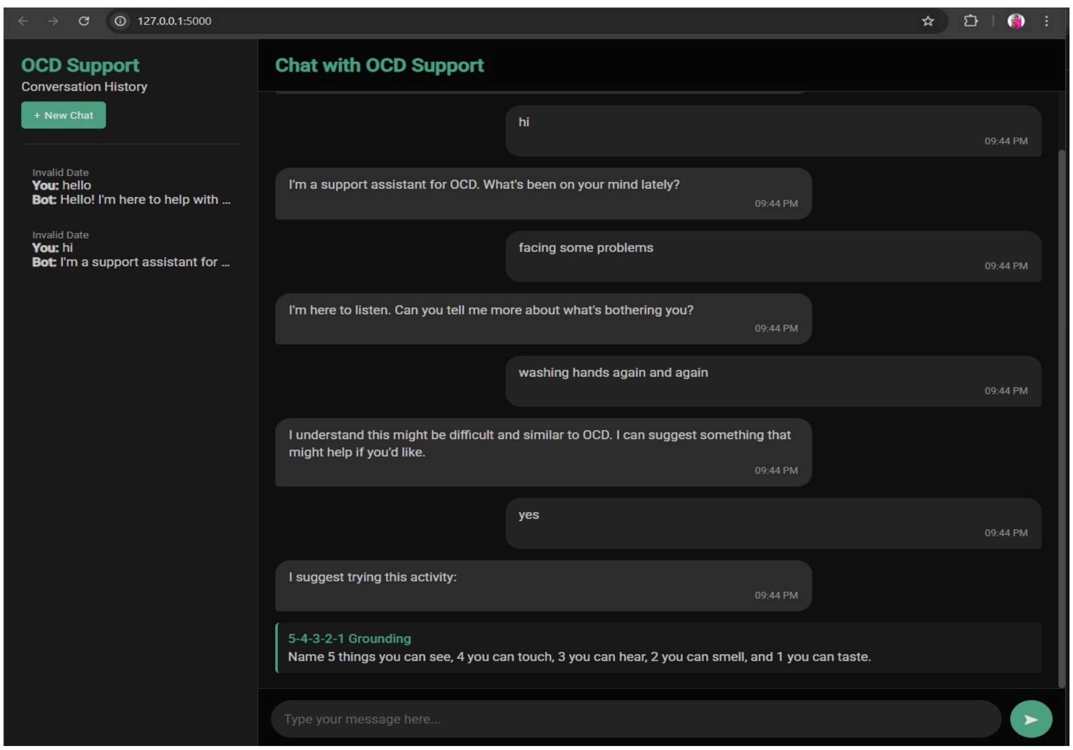


Figure 05: Task according to detection

ROC Curve:

The ROC Curve (Receiver Operating Characteristic Curve) is a graphical representation that shows the trade-off between the true positive rate and the false positive rate for various classification thresholds.

- **AUC Score:**
    - The Area Under Curve (AUC) score is 0.75, which indicates that the model has a good ability to distinguish between OCD and Non-OCD cases.
- **Meaning:**
    - An AUC score of 0.5 represents random guessing, while a score of 1 means perfect classification. A score of 0.75 is decent, but there's still room for improvement.

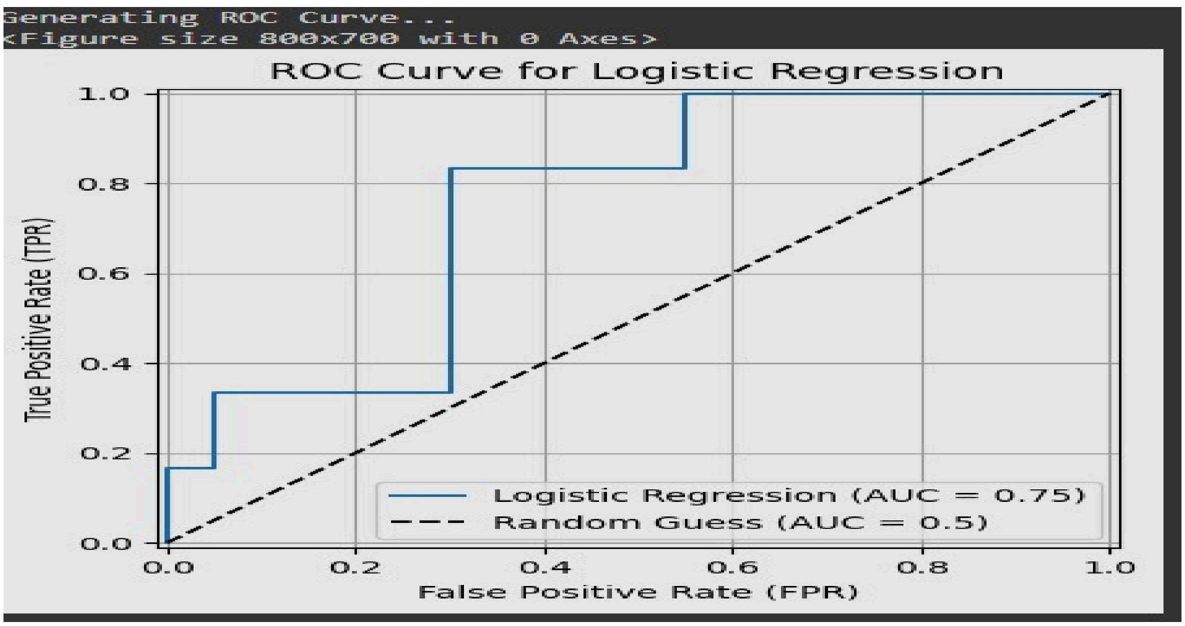


Figure 06: ROC curve

Predicted Probability Distribution:

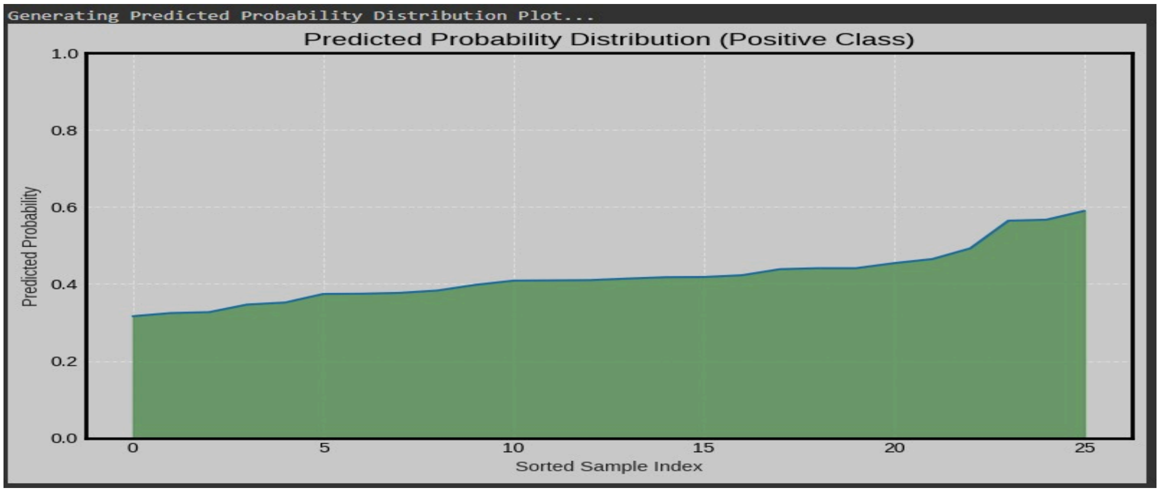


Figure 07: Predicted Probability Distribution curve

This image visualizes the model's confidence in predicting whether a given message is OCD-related or non-OCD-related.

- **Green Area:**
  - The green area in the graph indicates messages that the model is highly confident are OCD-related. As the confidence increases, the predictions become more reliable.
- **Insight:**
  - Most predictions fall in the mid-confidence zone (around 0.4 to 0.6), meaning the model is uncertain about many messages. This shows the model needs further tuning to achieve more decisive predictions.

11. Conclusion

This AI project successfully combines machine learning and Natural Language Processing (NLP) to create a chatbot that detects OCD-related behavior in user messages. The chatbot offers helpful responses and coping techniques for users, showing how AI can be used to support mental health awareness.

Through this project, the developer gained practical experience in full-stack AI development, from preparing data and training models to building a web application that allows users to interact with the chatbot. The project helped improve skills in both technical areas like data processing, machine learning, and backend development, and user-focused areas like designing a simple, responsive chat interface.

This project demonstrates the potential of AI to make a positive impact on mental health by offering accessible support for individuals with OCD. It also opens the door for future improvements, such as expanding the dataset, enhancing the chatbot's conversational abilities, and improving the detection model's accuracy.

12. Team Contribution

Table 4: Team Members Contributions

Team Member	Role/Responsibility
Member-1 Shumana Barua Nitu	Project ideation, dataset curation, machine learning model training, and initial backend setup
Member 2 Jumana Harun	Frontend integration (UI/UX), assisted in backend development, connected frontend to backend
Member 3 Nusaiba Ibnath	Code review, bug fixing, documentation, and testing

13. References

scikit-learn Documentation – https://scikit-learn.org/

Flask Documentation – https://flask.palletsprojects.com/

NLTK Documentation – https://www.nltk.org