

Artificial Intelligence

**Project Idea For**

**NAAN MUDHALVAN**

**SABIKA THUSLEEMA A**

[**953621106085@ritrjpm.ac.in**](mailto:953621106085@ritrjpm.ac.in)

**ICEYA R**

[**953621106030@ritrjpm.ac.in**](mailto:953621106030@ritrjpm.ac.in)

**GAYATHRI M**

[**953621106021@ritrjpm.ac.in**](mailto:953621106021@ritrjpm.ac.in)

**KAVIYA V**

[**953621106044@ritrjpm.ac.in**](mailto:953621106044@ritrjpm.ac.in)

**DIABETES PREDICTION CHATBOT DESIGN**

**Machine learning category**

Python is a programming language. Some useful tools and libraries are Chatterbot, Scikit-Learn, Tensor-flow, Numpy, and Pandas.

Front end: Python Tkinter Data set, HTML, CSS, and Java script the following

**URL:** [**https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot**](https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot)

**Introduction**

Python-based chatbots are already widely used in a variety of businesses, redefining user interactions. These intelligent systems, which can mimic real human language, have a variety of uses, including in digital commerce and healthcare. This manual offers an in-depth tutorial on using the ChatterBot library to build a Python chatbot.

**Problem Definition**

The goal is to create an artificial intelligence (AI)-powered diabetes prediction system that uses machine learning algorithms to assess medical data and forecast a person's risk of getting diabetes. The system's goal is to offer early risk assessment and individualized preventive strategies so that people may take charge of their health.

**Design Thinking**

**Functionality**

The primary features of the chatbot are as follows:

**1.Risk assessment:**

Evaluate the likelihood of acquiring diabetes by analyzing medical data.

**2. Advice and Information:**

Offer advice on methods for preventing diabetes and sensible lifestyle decisions.

**3. User Interaction:**

Engage users in casual and educational discussions on queries pertaining to diabetes.

**4. Resource Direction**:

Point visitors to pertinent sites for more details or expert assistance.

**User Interface**

**1. Integration Platform**:

To ensure accessibility, the chatbot will be incorporated into a user-friendly website.

**2. User Input:**

Create a user-friendly interface that allows users to enter pertinent medical data.

**3. Output Display:**

Present risk assessment findings and tailored advice in an easily readable way.

**Natural Language Processing (NLP)**

**1. User Intent Recognition:**

Use NLP approaches to identify user intentions for health-related searches.

**2. Extraction of Entities:**

Take user input and extract pertinent entities like age, family medical history, and lifestyle choices

**3. Flow of the Conversation:**

Create a conversational flow that feels natural to engage people and collect vital data.

**Responses**

**1. Accurate Answers:**

Formulate information-rich responses based on the evaluated medical data.

**2. Recommendations**

Make individualized recommendations for dietary adjustments, exercise regimens, and lifestyle improvements.

**3. Support:**

Help people grasp medical terminology and evaluate risk assessment findings.

**Integration**

**1. Website Integration:**

Ensure a seamless and responsive user experience by integrating the chatbot into the website.

**2. Data security**

Put in place strong security measures to safeguard user information and uphold privacy.

**3. Optional API Integration**

Look into the possibility of connecting the chatbot to external health databases or APIs to gain further information.

**Testing and Improvement**

**1. User testing:**

To find potential problems and gain input, thoroughly test users.

**2. Performance Measurements**

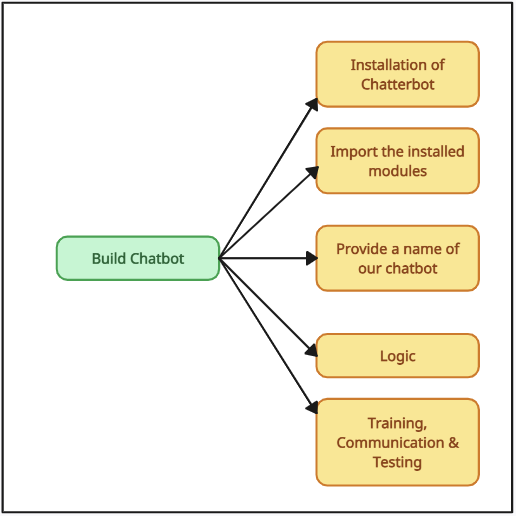
Establish performance measures for the chatbot, such as prediction accuracy and user happiness.

**3. Constant Development:**

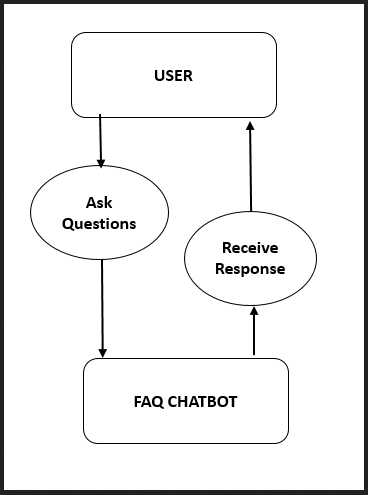
Update the chatbot frequently in response to user input, modifications in medical standards, and developments in machine learning techniques.

**Deployment**

Use a suitable platform to deploy the chatbot, like a website or messaging service, to enable user engagement.

**Building**

**Use-Case Diagram:**

****

**How Does the Chatbot Python Work?**

**Rule-Based Methodology**

Python chatbots employ pre-established rules to understand and reply to user requests. The rules are manually programmed by developers.

**Self-Learning Strategy**

Machine learning is used by chatbots to gradually improve their conversational abilities. Two groups comprise:

Retrieve predefined responses from a knowledge base using retrieval-based models.

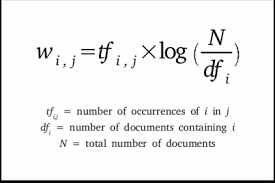
Generative Models: Create responses from scratch using advanced models.

**What is Chatter Bot Library?**

A Python library called Chatter Bot makes creating chatbots easier and takes care of problems with natural language processing. Language independence and TF-IDF-based answer creation are important aspects.

**How Does Chatter Bot Library Work?**

Using TF-IDF and cosine similarity, Chatter Bot combines a language database with an AI system to match user input to appropriate responses.



**How To Install Chatter Bot In Python**

1. Launch a terminal or command prompt and ensure Python is configured.
2. Use the command "pip install chatterbot" to install ChatterBot and its dependencies.
3. Import Chatter Bot in your Python script or interactive environment to start building your chatbot.

**Conclusion:**

The creation of a chatbot that can anticipate diabetes using AI needs a variety of cutting-edge tools. The chatbot's analytical capabilities are enhanced by leveraging Python for its robust ecosystem, machine learning libraries like Scikit-learn for predictive modeling, and NLP frameworks like NLTK. Chatter Bot makes conversational logic easier to develop, while Flask or Django makes a solid backend framework. HTML, CSS, JavaScript, or Python Tkinter can all be used for user interaction. These technology' flexibility enables the chatbot to efficiently offer individualized health information and advice.

**Program:**

import numpy as np

import pandas as pd

import math, time, datetime

import random as rd

import statsmodels.api as sm

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import scale, StandardScaler

from sklearn.model\_selection import train\_test\_split, GridSearchCV, cross\_val\_score

from sklearn.metrics import confusion\_matrix, accuracy\_score, mean\_squared\_error, r2\_score, roc\_auc\_score, roc\_curve, classification\_report

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.neural\_network import MLPClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import GradientBoostingClassifier

from lightgbm import LGBMClassifier

from sklearn.model\_selection import KFold

import warnings

warnings.simplefilter(action = "ignore")