FAQ Chatbot — Step-by-Step Implementation

1. Tech Stack

Python for backend logic

NLTK or spaCy for text preprocessing

scikit-learn for TF-IDF vectorization and cosine similarity

Optional UI:

Tkinter for a desktop chat interface

Streamlit for a web interface

---

2. Workflow

1. Collect FAQ data

Store in a JSON or CSV format: each entry has a "question" and "answer".

2. Preprocess text

Convert to lowercase

Tokenize text

Remove stopwords and punctuation

Lemmatize words

3. Vectorize

Use TF-IDF to represent each question numerically.

4. Match user queries

Convert user input to a TF-IDF vector.

Compare with all stored question vectors using cosine similarity.

Pick the highest-scoring match.

5. Return the answer

Display it in the console or a chat UI.

---

3. Example Code (Console Version)

```
import nltk
import string
```

```
from
sklearn.feature_extraction.text
import TfidfVectorizer

from sklearn.metrics.pairwise
import cosine_similarity


# Download NLTK data

nltk.download('punkt')

nltk.download('stopwords')

nltk.download('wordnet')


from nltk.corpus import
stopwords

from nltk.stem import
WordNetLemmatizer
```

```
# Sample FAQ dataset
faq_data = [
    {"question": "What is your return policy?", "answer": "You can return products within 30 days of purchase."},
    {"question": "Do you ship internationally?", "answer": "Yes, we ship worldwide with extra shipping charges."},
    {"question": "How do I track my order?", "answer": "Use the tracking link sent to your email after dispatch."}
```

```
]


# Preprocessing
lemmatizer =
WordNetLemmatizer()
stop_words =
set(stopwords.words('english'))


def preprocess(text):
    text = text.lower()
    tokens =
nltk.word_tokenize(text)
    tokens =
[lemmatizer.lemmatize(t) for t in
tokens if t not in
```

```
string.punctuation and t not in
stop_words]
    return ' '.join(tokens)


# Prepare FAQ questions
questions =
[preprocess(f['question']) for f in
faq_data]


# Vectorize questions
vectorizer = TfidfVectorizer()
tfidf_matrix =
vectorizer.fit_transform(questions
)
```

```
# Function to get best match
def get_answer(user_query):
    user_query_processed = preprocess(user_query)
    user_vec = vectorizer.transform([user_query_processed])
    similarity = cosine_similarity(user_vec, tfidf_matrix)
    best_match_index = similarity.argmax()
    return faq_data[best_match_index]['answer']
```

```
# Chat loop
print("FAQ Chatbot (type 'quit' to exit)")
while True:
    user_input = input("You: ")
    if user_input.lower() == 'quit':
        break
    print("Bot:", get_answer(user_input))
```

---

4. Optional — Add a Simple Chat UI (Tkinter)

If you want, I can make this same chatbot run inside a chat window with:

Scrollable chat history

Text input field

Send button