13. Source Code

```
# Install necessary libraries
!pip install -q nltk scikit-learn

# Import necessary libraries
import nltk
import random
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

# Download necessary NLTK data
nltk.download('punkt')
nltk.download('wordnet')

# Define the chatbot's knowledge base
knowledge_base = {
   "intents": [
      {
         "tag": "greeting",
```

```
            "patterns": ["hello", "hi", "hey", "good morning", "good evening"],
            "responses": ["Hi, how can I assist you?", "Hello! What's up?", "Hey,
how's it going?"]
        },
        {
            "tag": "goodbye",
            "patterns": ["bye", "see you later", "goodbye"],
            "responses": ["See you later!", "Bye! Have a great day.", "Goodbye!"]
        },
        {
            "tag": "thanks",
            "patterns": ["thanks", "thank you", "appreciate it"],
            "responses": ["You're welcome!", "No problem!", "Anytime!"]
        },
        # Add more intents as needed
    ]
}

# Prepare the training data
patterns = []
tags = []

for intent in knowledge_base["intents"]:
    for pattern in intent["patterns"]:
        patterns.append(pattern.lower())
        tags.append(intent["tag"])

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(patterns, tags, test_size=0.2,
random_state=42)

# Convert text to numerical features using TF-IDF
vectorizer = TfidfVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

```
# Train a Naive Bayes classifier
clf = MultinomialNB()
clf.fit(X_train_vec, y_train)

# Evaluate the model
y_pred = clf.predict(X_test_vec)
print("Model Accuracy:", accuracy_score(y_test, y_pred))

# Function to generate a chatbot response
def generate_response(user_input):
    user_input = user_input.lower()
    input_vec = vectorizer.transform([user_input])
    predicted_tag = clf.predict(input_vec)[0]

    for intent in knowledge_base["intents"]:
        if intent["tag"] == predicted_tag:
            return random.choice(intent["responses"])
    return "I'm not sure how to respond to that."

# Start the chatbot interface
print("\nStart chatting with the bot! Type 'quit' to stop.")
while True:
    user_input = input("You: ")
    if user_input.lower() == "quit":
        print("Chatbot: Goodbye!")
        break
    response = generate_response(user_input)
    print("Chatbot:", response)
```