```python
import yfinance as yf
import nltk
from nltk.chat.util import Chat, reflections
import streamlit as st
import re  # For better handling of user input

# Download required NLTK data
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

# Define patterns and responses for the chatbot
# Define patterns and responses for the chatbot
patterns = [
    (r'hi|hello|hey', ['Hello! □', 'Hi there! □', 'Hey! □']),
    (r'how are you', ['I am doing well, thank you! □', 'I\'m great, how are you? □']),
    (r'what is your name', ['I am StockBot, your friendly stock market assistant! □']),
    (r'bye|goodbye', ['Goodbye! □', 'See you later! □', 'Have a great day! □']),
    (r'stock price of (.*)', ['Let me check the stock price for %1... □']),
    (r'help', ['I can help you with:\n- Stock prices □\n- Market information □\n- Basic conversation □□\nJust ask!']),
    (r'(.*)', ['I\'m not sure how to respond to that. □', 'Could you please rephrase that? □□□'])
]


# Create chatbot instance
chatbot = Chat(patterns, reflections)


def get_bot_response(user_input):
    try:
        return chatbot.respond(user_input)
    except Exception as e:
        return "I encountered an error. Could you please try again?"
```

```
33
34
35 # Fetch real-time stock price using yfinance
36 def fetch_stock_data(ticker):
37     try:
38         # Fetch data using yfinance
39         stock = yf.Ticker(ticker)
40
41         # Get the latest stock price
42         stock_info = stock.history(period="1d")
43
44         # If the stock data is available, return the
   last price
45         if not stock_info.empty:
46             last_price = stock_info['Close'].iloc[-1]
47             return last_price
48         else:
49             return None
50     except Exception as e:
51         print(f"Error fetching stock data: {e}")
52         return None
53
54
55 # Create Streamlit chat interface
56 st.subheader("StockBot 🤖")
57 user_input = st.text_input("You:", "")
58
59 if user_input:
60     response = get_bot_response(user_input)
61     st.text_area("StockBot:", value=response, height=
   100, disabled=True)
62
63     # If user asks about stock price, show the
   relevant stock information
64     if "stock price" in user_input.lower():
65         try:
66             # Extract ticker symbol using regex (
   handling more variations in input)
67             match = re.search(r"stock price of (\w+)"
   , user_input.lower())
68             if match:
```

```
69                  ticker = match.group(1).upper()  # Extract the ticker symbol from the input
70
71                  # Fetch stock price
72                  stock_price = fetch_stock_data(ticker)
73
74                  if stock_price is not None:
75                      st.metric(f"{ticker} Current Price □ ", f"{stock_price:.2f} USD □")
76                  else:
77                      st.warning(f"Could not fetch stock price for {ticker}. Please check the ticker symbol ⚠□.")
78              else:
79                  st.warning("Please provide a valid stock ticker, like 'AAPL' or 'GOOGL'. □")
80
81          except Exception as e:
82              st.warning("Could not fetch stock price. Please check the ticker symbol or try again later. □")
83
```