**FinGPT**

[Beginner’s Guide to FinGPT: Training with LoRA and ChatGLM2–6B | by ByFintech @ AI4Finance Foundation | Medium](https://ai4finance-foundation.medium.com/beginners-guide-to-fingpt-training-with-lora-chatglm2-6b-9eb5ace7fe99)

### Feasible Features to Implement

1. **Financial Sentiment Analysis**
   * **Description**: Analyze financial news and social media to gauge market sentiment.
   * [**Resources**: Use pre-trained models like FinBERT for sentiment analysis1](https://github.com/AI4Finance-Foundation/FinGPT).
   * **Documentation**: FinBERT on Hugging Face
2. **Stock Price Prediction**
   * **Description**: Predict future stock prices using historical data.
   * [**Resources**: Implement LSTM models for time series forecasting2](https://huggingface.co/FinGPT).
   * **Documentation**: LSTM Stock Price Prediction GitHub
3. **Financial Data Retrieval**
   * **Description**: Fetch real-time financial data from APIs like Alpha Vantage or Yahoo Finance.
   * [**Resources**: Use Python libraries to interact with these APIs3](https://konfuzio.com/en/fingpt/).
   * **Documentation**: Alpha Vantage API Documentation
4. **Portfolio Optimization**
   * **Description**: Suggest optimal asset allocation based on user preferences.
   * [**Resources**: Use libraries like PyPortfolioOpt for portfolio optimization](https://github.com/AI4Finance-Foundation/FinGPT)[4](https://github.com/samkamau81/FinGPT_).
   * **Documentation**: PyPortfolioOpt GitHub
5. **Interactive Chat Interface**
   * **Description**: Develop a user-friendly chat interface to interact with the bot.
   * [**Resources**: Use frameworks like Rasa or Dialogflow for building chatbots](https://github.com/AI4Finance-Foundation/FinGPT)[5](https://www.marktechpost.com/2023/06/16/meet-fingpt-an-open-source-financial-large-language-model-llms/).
   * **Documentation**: Rasa Documentation

### Implementation Plan

1. **Month 1-2: Research and Planning**
   * Define project scope and features.
   * Gather necessary datasets and resources.
   * Set up development environment.
2. **Month 3-4: Financial Sentiment Analysis**
   * Implement sentiment analysis using FinBERT.
   * Test and validate the model with financial news data.
3. **Month 5-6: Stock Price Prediction**
   * Develop and train LSTM models for stock price prediction.
   * Integrate the model with real-time data sources.
4. **Month 7: Financial Data Retrieval**
   * Implement API calls to fetch real-time financial data.
   * Ensure data is correctly formatted and stored.
5. **Month 8: Portfolio Optimization**
   * Develop portfolio optimization algorithms.
   * Create a user interface for inputting preferences and displaying results.
6. **Month 9: Integration and Testing**
   * Integrate all components into a cohesive system.
   * Conduct thorough testing and debugging.
   * Prepare comprehensive documentation and user guides.

### Additional Resources

* **FinGPT GitHub Repository**: [FinGPT on GitHub](https://github.com/AI4Finance-Foundation/FinGPT" \t "_blank)
* **FinGPT Documentation**: [FinGPT Documentation](https://github.com/AI4Finance-Foundation/FinGPT/blob/master/README.md" \t "_blank)

By focusing on these key features and following a structured implementation plan, you should be able to create a robust finance chatbot within your timeframe. Good luck with your project! If you have any more questions or need further assistance, feel free to ask. 😊

[1](https://github.com/AI4Finance-Foundation/FinGPT)[: FinBERT on Hugging Face](https://github.com/AI4Finance-Foundation/FinGPT)[2](https://huggingface.co/FinGPT)[: LSTM Stock Price Prediction GitHub 3](https://konfuzio.com/en/fingpt/)[: Alpha Vantage API Documentation 4](https://github.com/samkamau81/FinGPT_)[: PyPortfolioOpt GitHub](https://github.com/AI4Finance-Foundation/FinGPT)[5](https://www.marktechpost.com/2023/06/16/meet-fingpt-an-open-source-financial-large-language-model-llms/): Rasa Documentation

### User Journey

1. **Sign-Up/Login**
   * **Description**: Users create an account or log in to access the chatbot.
   * **Interface**: Simple sign-up form with email and password or social media login options.
2. **Dashboard**
   * **Description**: After logging in, users are greeted with a dashboard displaying key financial metrics and recent activity.
   * **Interface**: Clean layout with widgets showing portfolio performance, latest news, and market trends.
3. **Chat Interface**
   * **Description**: Users can interact with the chatbot through a conversational interface.
   * **Interface**: A chat window where users can type or use voice commands to ask questions.
4. **Financial Sentiment Analysis**
   * **User Action**: Users ask the chatbot for the latest sentiment on a particular stock or market.
   * **Response**: The chatbot analyzes recent news and social media posts to provide a sentiment score and summary.
   * **Example**: “What’s the sentiment on Tesla stock today?”
5. **Stock Price Prediction**
   * **User Action**: Users request predictions for specific stocks.
   * **Response**: The chatbot provides a forecast based on historical data and LSTM models.
   * **Example**: “Predict the price of Apple stock for the next week.”
6. **Real-Time Financial Data**
   * **User Action**: Users ask for real-time data on stocks, indices, or other financial instruments.
   * **Response**: The chatbot fetches and displays the latest data from financial APIs.
   * **Example**: “Show me the current price of Bitcoin.”
7. **Portfolio Optimization**
   * **User Action**: Users input their investment preferences and risk tolerance.
   * **Response**: The chatbot suggests an optimal portfolio allocation.
   * **Example**: “Optimize my portfolio for moderate risk.”
8. **Personal Finance Management**
   * **User Action**: Users can track their expenses, set budgets, and get financial advice.
   * **Response**: The chatbot provides insights and recommendations based on user data.
   * **Example**: “Help me create a monthly budget.”
9. **Notifications and Alerts**
   * **Description**: Users receive notifications for important events, such as significant market movements or portfolio updates.
   * **Interface**: Push notifications or in-app alerts.
10. **Help and Support**
    * **Description**: Users can access help and support for any issues or questions.
    * **Interface**: FAQ section, live chat with support, or email support options.

### Example Interaction

**User**: “What’s the latest news on Amazon stock?”

**Chatbot**: “The latest sentiment on Amazon stock is positive. Recent news highlights include strong quarterly earnings and a new product launch. Here’s a summary of the top articles: [links to articles].”

**User**: “Can you predict the price of Amazon stock for the next month?”

**Chatbot**: “Based on historical data and our predictive model, the forecasted price of Amazon stock for the next month is expected to be around $3,500. Please note that predictions are not guaranteed.”

**User**: “Optimize my portfolio for low risk.”

**Chatbot**: “Based on your preference for low risk, I recommend the following asset allocation: 60% bonds, 30% large-cap stocks, and 10% cash. Would you like to see more details?”

### Financial Sentiment Analysis

* **Feasibility**: High
* **Difficulty**: Moderate
* **Details**: Using pre-trained models like FinBERT simplifies the process. You will need to fine-tune the model on financial data, which requires some understanding of NLP and model training.
* **Challenges**: Acquiring a large, labeled dataset for fine-tuning can be time-consuming. Ensuring the model’s accuracy and relevance to current market conditions is crucial.

### 2. Stock Price Prediction

* **Feasibility**: Moderate
* **Difficulty**: High
* **Details**: Implementing LSTM models for time series forecasting is feasible but requires significant computational resources and expertise in deep learning.
* **Challenges**: Training LSTM models can be computationally intensive. You will need a high-performance computer or access to cloud-based GPU resources. Collecting and preprocessing historical stock data is also necessary.

### 3. Real-Time Financial Data Retrieval

* **Feasibility**: High
* **Difficulty**: Low
* **Details**: Fetching data from APIs like Alpha Vantage or Yahoo Finance is straightforward using Python libraries.
* **Challenges**: Ensuring the reliability and accuracy of the data. Handling API rate limits and potential downtime.

### 4. Portfolio Optimization Tool

* **Feasibility**: Moderate
* **Difficulty**: Moderate
* **Details**: Using libraries like PyPortfolioOpt simplifies the implementation. You will need to understand optimization algorithms and financial metrics.
* **Challenges**: Ensuring the tool provides meaningful and actionable recommendations. Handling user-specific constraints and preferences.

### 5. Interactive Chat Interface

* **Feasibility**: High
* **Difficulty**: Moderate
* **Details**: Building a chat interface using frameworks like Rasa or Dialogflow is feasible. You will need to design conversational flows and integrate various components.
* **Challenges**: Ensuring a smooth and intuitive user experience. Handling natural language understanding and generating accurate responses.

### 6. Comprehensive Documentation

* **Feasibility**: High
* **Difficulty**: Low
* **Details**: Documenting the project is straightforward but requires attention to detail.
* **Challenges**: Ensuring the documentation is clear, comprehensive, and useful for future users or developers.

### Hardware Constraints

* **Feasibility**: Moderate
* **Difficulty**: High
* **Details**: Training models, especially deep learning models, requires significant computational power. A high-performance computer with a powerful GPU is recommended.
* **Challenges**: If you don’t have access to such hardware, consider using cloud-based services like AWS, Google Cloud, or Azure, which can be costly.

### Getting Training Data

* **Feasibility**: Moderate
* **Difficulty**: High
* **Details**: Acquiring and preprocessing large datasets for training models is crucial.
* **Challenges**: Finding high-quality, labeled datasets can be challenging. You may need to combine multiple data sources and clean the data extensively.

### Overall Feasibility and Difficulty

* **Feasibility**: Moderate to High
* **Difficulty**: Moderate to High
* **Details**: The project is ambitious but achievable within 9 months if you plan carefully and focus on the most critical features.
* **Challenges**: Balancing the scope of the project with the available time and resources. Prioritizing features that provide the most value and are feasible to implement.

### Recommendations

1. **Start Small**: Begin with the core features like real-time data retrieval and sentiment analysis. These are relatively easier to implement and provide immediate value.
2. **Iterative Development**: Use an iterative approach to gradually add more complex features like stock price prediction and portfolio optimization.
3. **Leverage Existing Tools**: Utilize pre-trained models and existing libraries to save time and effort.
4. **Focus on Documentation**: Ensure thorough documentation from the start to make the project easier to manage and understand.

**Chatbot Sentiment Analysis from Coders perspective**

**User Input**

* **User Action**: The user types the query “What’s the latest news on Amazon stock?” into the chat interface.
* **Chat Interface**: The chatbot interface captures the user’s input and sends it to the backend for processing.

**2. Natural Language Understanding (NLU)**

* **NLU Module**: The chatbot uses an NLU module to understand the user’s intent and extract relevant entities.
  + **Intent Recognition**: The NLU module identifies that the user’s intent is to get the latest news about a specific stock.
  + **Entity Extraction**: The NLU module extracts the entity “Amazon” as the stock of interest.

**3. Query Formulation**

* **Backend Processing**: The backend formulates a query to fetch the latest news about Amazon stock.
  + **API Request**: The backend prepares an API request to a financial news API (e.g., Alpha Vantage, Yahoo Finance, or a news aggregator like NewsAPI).

**4. API Call**

* **API Integration**: The backend sends the API request to fetch the latest news articles related to Amazon stock.
  + **Example API Request**:

**Python**

import requests

api\_key = 'YOUR\_API\_KEY'

url = f'https://newsapi.org/v2/everything?q=Amazon&apiKey={api\_key}'

response = requests.get(url)

news\_data = response.json()

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

**5. Data Processing**

* **Response Handling**: The backend receives the response from the API, which contains news articles related to Amazon stock.
  + **Data Extraction**: Extract relevant information such as article titles, summaries, publication dates, and URLs from the API response.
  + **Example Data Extraction**:

**Python**

articles = news\_data['articles']

latest\_news = []

for article in articles:

news\_item = {

'title': article['title'],

'summary': article['description'],

'url': article['url'],

'published\_at': article['publishedAt']

}

latest\_news.append(news\_item)

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

**6. Sentiment Analysis with FinBERT**

* **Sentiment Analysis**: Use FinBERT to analyze the sentiment of each news article.
  + **Example Sentiment Analysis**:

**Python**

from transformers import BertTokenizer, BertForSequenceClassification

import torch

model\_name = "ProsusAI/finbert"

tokenizer = BertTokenizer.from\_pretrained(model\_name)

model = BertForSequenceClassification.from\_pretrained(model\_name)

def analyze\_sentiment(text):

inputs = tokenizer(text, return\_tensors="pt", truncation=True, padding=True)

outputs = model(\*\*inputs)

logits = outputs.logits

sentiment = torch.argmax(logits, dim=1).item()

return sentiment # 0: negative, 1: neutral, 2: positive

for news in latest\_news:

sentiment = analyze\_sentiment(news['summary'])

news['sentiment'] = sentiment

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

**7. Response Generation with LLM**

* **LLM Integration**: Use a large language model like FinGPT to generate a dynamic, context-aware response based on the processed data.
  + **Example Response Generation**:

**Python**

from transformers import GPT2LMHeadModel, GPT2Tokenizer

gpt\_model\_name = "gpt2"

gpt\_tokenizer = GPT2Tokenizer.from\_pretrained(gpt\_model\_name)

gpt\_model = GPT2LMHeadModel.from\_pretrained(gpt\_model\_name)

def generate\_response(news\_items):

prompt = "Here are the latest news articles on Amazon stock:\n"

for news in news\_items[:5]: # Limit to top 5 articles

sentiment\_map = {0: "negative", 1: "neutral", 2: "positive"}

prompt += f"- {news['title']} (Published on {news['published\_at']})\n"

prompt += f" Sentiment: {sentiment\_map[news['sentiment']]}\n"

prompt += f" Summary: {news['summary']}\n"

prompt += f" Read more: {news['url']}\n\n"

inputs = gpt\_tokenizer.encode(prompt, return\_tensors="pt")

outputs = gpt\_model.generate(inputs, max\_length=500, num\_return\_sequences=1)

response = gpt\_tokenizer.decode(outputs[0], skip\_special\_tokens=True)

return response

response\_text = generate\_response(latest\_news)

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

**8. User Response**

* **Chat Interface**: The generated response is sent back to the chat interface, where it is displayed to the user.
  + **Example User Response**:
  + Here are the latest news articles on Amazon stock:
  + - Amazon's Q2 Earnings Beat Expectations (Published on 2024-08-16)
  + Sentiment: positive
  + Summary: Amazon reported better-than-expected earnings for Q2, driven by strong e-commerce sales and AWS growth.
  + Read more: https://example.com/article1
  + - Amazon Launches New AI-Powered Shopping Features (Published on 2024-08-15)
  + Sentiment: positive
  + Summary: Amazon has introduced new AI-powered features to enhance the shopping experience for its customers.
  + Read more: https://example.com/article2

**Summary of the Process with FinBERT and LLM**

1. **User Input**: User submits a query.
2. **NLU**: Understand the intent and extract entities.
3. **Query Formulation**: Formulate an API request.
4. **API Call**: Fetch data from a financial news API.
5. **Data Processing**: Extract and process relevant information.
6. **Sentiment Analysis**: Use FinBERT to analyze the sentiment of news articles.
7. **LLM Integration**: Use a large language model to generate a dynamic, context-aware response.
8. **User Response**: Display the response to the user.

**Chatbot Stock Prediction from Coders perspective**

### . User Input

* **User Action**: The user types the query “Predict the price of Amazon stock for the next week” into the chat interface.
* **Chat Interface**: The chatbot interface captures the user’s input and sends it to the backend for processing.

### 2. Natural Language Understanding (NLU)

* **NLU Module**: The chatbot uses an NLU module to understand the user’s intent and extract relevant entities.
  + **Intent Recognition**: The NLU module identifies that the user’s intent is to get a stock price prediction.
  + **Entity Extraction**: The NLU module extracts the entity “Amazon” as the stock of interest and the time frame “next week”.

### 3. Data Collection

* **Historical Data Retrieval**: The backend fetches historical stock price data for Amazon from a financial data API (e.g., Alpha Vantage, Yahoo Finance).
  + **Example API Request**:

**Python**

import requests

api\_key = 'YOUR\_API\_KEY'

symbol = 'AMZN'

url = f'https://www.alphavantage.co/query?function=TIME\_SERIES\_DAILY&symbol={symbol}&apikey={api\_key}'

response = requests.get(url)

data = response.json()

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

### 4. Data Preprocessing

* **Data Cleaning and Normalization**: Clean and normalize the historical data to prepare it for model input.
  + **Example Data Preprocessing**:

**Python**

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

# Convert the API response to a DataFrame

df = pd.DataFrame(data['Time Series (Daily)']).T

df = df.astype(float)

df = df[['4. close']] # Use closing prices

# Normalize the data

scaler = MinMaxScaler(feature\_range=(0, 1))

scaled\_data = scaler.fit\_transform(df)

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

### 5. Model Input Preparation

* **Create Input Sequences**: Prepare the data in sequences suitable for LSTM input.
  + **Example Sequence Preparation**:

**Python**

import numpy as np

def create\_sequences(data, time\_step=60):

X, Y = [], []

for i in range(len(data) - time\_step - 1):

X.append(data[i:(i + time\_step), 0])

Y.append(data[i + time\_step, 0])

return np.array(X), np.array(Y)

time\_step = 60

X, Y = create\_sequences(scaled\_data, time\_step)

X = X.reshape(X.shape[0], X.shape[1], 1)

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

### 6. Model Training (One-Time Setup)

* **Train the LSTM Model**: Train an LSTM model using historical data (this step is typically done once and the model is saved for future predictions).
  + **Example Model Training**:

**Python**

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

model = Sequential()

model.add(LSTM(50, return\_sequences=True, input\_shape=(time\_step, 1)))

model.add(LSTM(50, return\_sequences=False))

model.add(Dense(25))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mean\_squared\_error')

model.fit(X, Y, batch\_size=1, epochs=1)

# Save the model

model.save('stock\_prediction\_model.h5')

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

### 7. Prediction

* **Load the Trained Model**: Load the pre-trained LSTM model and use it to make predictions.
  + **Example Prediction**:

**Python**

from tensorflow.keras.models import load\_model

# Load the model

model = load\_model('stock\_prediction\_model.h5')

# Prepare the input for prediction

last\_60\_days = scaled\_data[-60:]

X\_input = last\_60\_days.reshape(1, -1)

X\_input = X\_input.reshape((1, time\_step, 1))

# Make the prediction

predicted\_price = model.predict(X\_input)

predicted\_price = scaler.inverse\_transform(predicted\_price)

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

### 8. Response Generation with LLM

* **LLM Integration**: Use a large language model to generate a dynamic, context-aware response based on the predicted stock price.
  + **Example Response Generation**:

**Python**

from transformers import GPT2LMHeadModel, GPT2Tokenizer

gpt\_model\_name = "gpt2"

gpt\_tokenizer = GPT2Tokenizer.from\_pretrained(gpt\_model\_name)

gpt\_model = GPT2LMHeadModel.from\_pretrained(gpt\_model\_name)

def generate\_response(predicted\_price):

prompt = f"The predicted price of Amazon stock for the next week is ${predicted\_price[0][0]:.2f}."

inputs = gpt\_tokenizer.encode(prompt, return\_tensors="pt")

outputs = gpt\_model.generate(inputs, max\_length=100, num\_return\_sequences=1)

response = gpt\_tokenizer.decode(outputs[0], skip\_special\_tokens=True)

return response

response\_text = generate\_response(predicted\_price)

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

### 9. User Response

* **Chat Interface**: The generated response is sent back to the chat interface, where it is displayed to the user.
  + **Example User Response**:
  + The predicted price of Amazon stock for the next week is $3450.75.

### Summary of the Process with Stock Prediction

1. **User Input**: User submits a query for stock price prediction.
2. **NLU**: Understand the intent and extract entities.
3. **Data Collection**: Fetch historical stock price data.
4. **Data Preprocessing**: Clean and normalize the data.
5. **Model Input Preparation**: Create input sequences for the LSTM model.
6. **Model Training**: Train the LSTM model (one-time setup).
7. **Prediction**: Use the trained model to predict future stock prices.
8. **LLM Integration**: Generate a dynamic, context-aware response.
9. **User Response**: Display the response to the user.

**NLU**

Natural Language Understanding (NLU) is a subfield of artificial intelligence (AI) and natural language processing (NLP) that focuses on enabling machines to understand and interpret human language in a meaningful way. NLU involves several key tasks that help machines comprehend the context, intent, and entities within a given text or speech input. Here are the main components and tasks involved in NLU:

### Key Components of NLU

1. **Intent Recognition**:
   * **Purpose**: Identify the user’s intention behind a given input.
   * **Example**: In the query “Predict the price of Amazon stock for the next week,” the intent is to get a stock price prediction.
2. **Entity Extraction**:
   * **Purpose**: Identify and extract specific pieces of information (entities) from the input.
   * **Example**: In the same query, “Amazon” is the entity representing the stock, and “next week” is the time frame.
3. **Context Understanding**:
   * **Purpose**: Understand the context in which the input is given to provide relevant responses.
   * **Example**: Recognizing that the user is asking about financial data and predictions.
4. **Sentiment Analysis**:
   * **Purpose**: Determine the sentiment or emotional tone of the input.
   * **Example**: Analyzing whether a user’s comment about a stock is positive, negative, or neutral.
5. **Semantic Parsing**:
   * **Purpose**: Convert natural language input into a structured format that machines can process.
   * **Example**: Parsing a sentence to identify its grammatical structure and meaning.

### How NLU Works in a Financial Chatbot

1. **User Input**: The user types a query into the chatbot interface.
2. **Text Preprocessing**: The input text is cleaned and normalized (e.g., removing punctuation, converting to lowercase).
3. **Intent Recognition**: The NLU module identifies the user’s intent (e.g., asking for a stock price prediction).
4. **Entity Extraction**: The NLU module extracts relevant entities from the input (e.g., “Amazon” and “next week”).
5. **Context Understanding**: The chatbot uses the recognized intent and extracted entities to understand the context of the query.
6. **Response Generation**: Based on the understood context, the chatbot generates an appropriate response, which may involve fetching data, performing calculations, or using a language model to generate a natural language reply.

### Example of NLU in Action

For the query “What’s the latest news on Amazon stock?”:

1. **User Input**: “What’s the latest news on Amazon stock?”
2. **Text Preprocessing**: “what’s the latest news on amazon stock”
3. **Intent Recognition**: Identify the intent as “get\_latest\_news”.
4. **Entity Extraction**: Extract the entity “Amazon” as the stock of interest.
5. **Context Understanding**: Recognize that the user wants recent news articles about Amazon stock.
6. **Response Generation**: Fetch the latest news articles about Amazon stock and generate a response.

### Tools and Libraries for NLU

* **Rasa**: An open-source framework for building conversational AI and NLU models.
* **spaCy**: A popular NLP library with built-in NLU capabilities.
* **Hugging Face Transformers**: Provides pre-trained models for various NLU tasks.
* **Dialogflow**: A Google service for building conversational interfaces with NLU.

NLU is a critical component of any intelligent chatbot, enabling it to understand and respond to user queries accurately and contextually.