BAROZ FINANCIAL CHATBOT

I started everything from scratch by opening a new environment and installing all the dependencies listed in requirements.txt. I also designed the logo for the chatbot application.

After creating a new Conda environment, I activated it and installed everything from requirements.txt.

Phase 1: Understanding Financial Chatbots

A financial chatbot primarily involves financial statistics, queries related to finance (most banking companies use them), and of course, graphs, which are included in each phase. While I was able to execute some tasks, others are still a challenge. However, I really enjoyed making this chatbot!

Step-by-Step Process

First, I wanted to get familiar with the entire environment, so I initially focused on data retrieval from a CSV dataset via a chatbot. My goal was to build a robust backend, and in multiple implementations, I used **GPT-3 and GPT-J**, successfully retrieving answers.

The frontend structure remains consistent with index.html.

User Interaction Flow

I planned out conversation flows for common queries, such as:

- "What is the stock price of XYZ?"
- "Show me recent financial news."

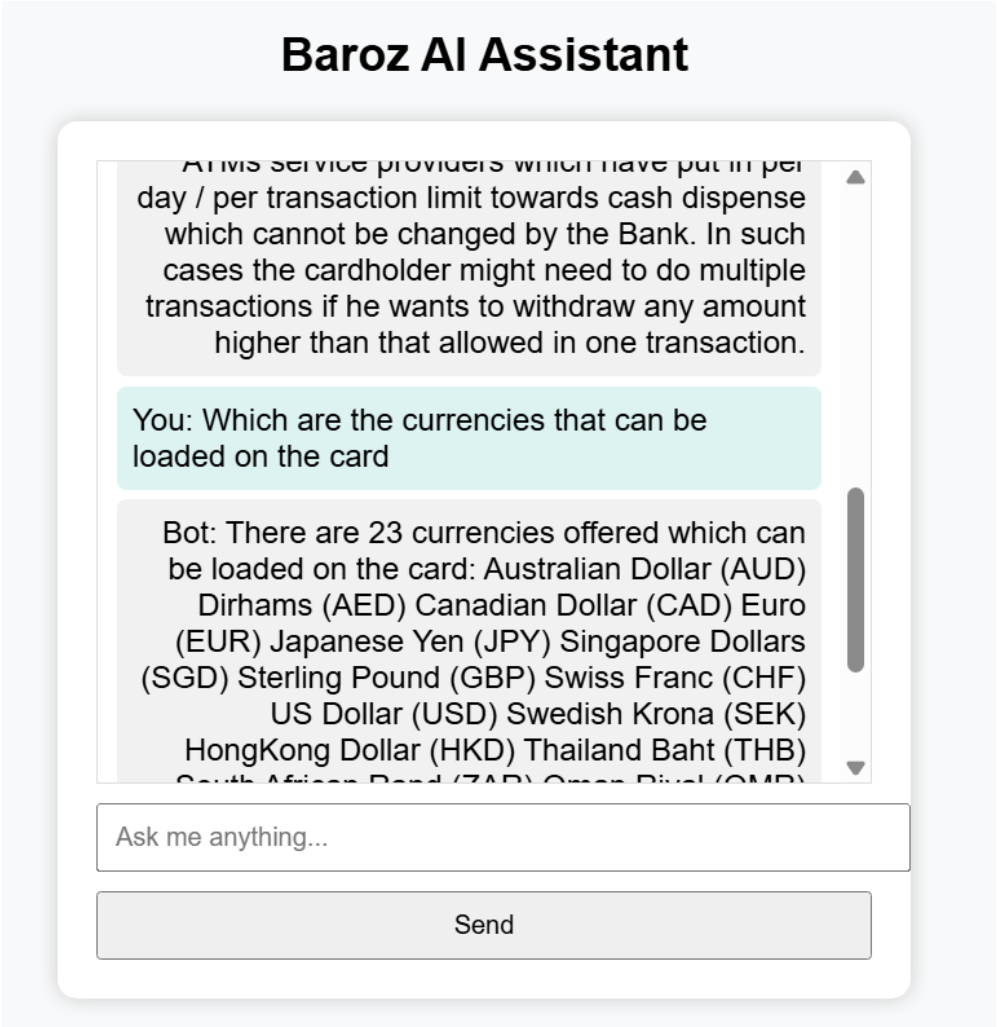
- "What is my investment portfolio performance?"

This functionality was implemented in app.py.

Vector Embeddings & Architecture Design

- **Architecture:** Designed a cloud-native architecture using microservices, serverless functions, and message queues for high performance and scalability.
- **Cloud Provider:** Selected free-tier or open-source services for multi-cloud deployment, including AWS, Google Cloud, and Azure, for real-time data handling.

Since I didn't have much time to deploy on Azure and Google Cloud, I was only able to deploy on **AWS** and successfully implemented real-time data handling.

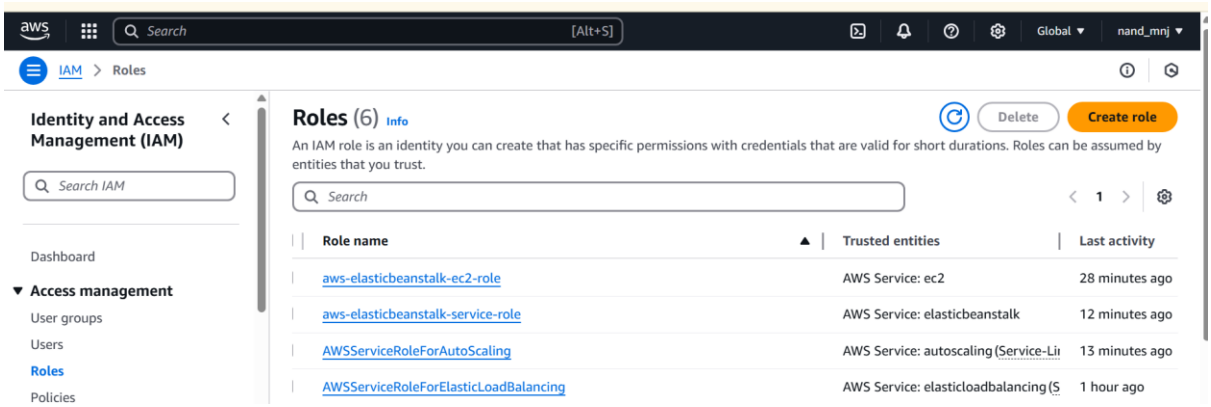


```
Is may still change according to user feedback. Please submit any feedback you may have in this issue: https://github.co
m/pytorch/vision/issues/6753, and you can also check out https://github.com/pytorch/vision/issues/7319 to learn more abo
ut the APIs that we suspect might involve future changes. You can silence this warning by calling torchvision.disable_be
ta_transforms_warning().
  warnings.warn(_BETA_TRANSFORMS_WARNING)
PermissionError: [WinError 32] The process cannot access the file because it is being used by another process: 'faiss_do
cument_store.db'. Retrying in 1 seconds...
PermissionError: [WinError 32] The process cannot access the file because it is being used by another process: 'faiss_do
cument_store.db'. Retrying in 1 seconds...
PermissionError: [WinError 32] The process cannot access the file because it is being used by another process: 'faiss_do
cument_store.db'. Retrying in 1 seconds...
Failed to remove faiss_document_store.db after 3 retries.
Failed to remove faiss_index after 3 retries.
FAISS Document Store initialized successfully.
Device set to use cpu
 * Debugger is active!
 * Debugger PIN: 103-829-097
127.0.0.1 - - [07/Jan/2025 11:24:10] "GET / HTTP/1.1" 200 -
Writing Documents: 10000it [00:01, 7958.46it/s]
Batches: 100%|                                                                                | 1/1 [00:00<00:00,  1.68it/s]
Batches: 100%|                                                                                | 13/13 [00:55<00:00,  4.25s/it]
Documents Processed: 10000 docs [00:55, 180.37 docs/s]
127.0.0.1 - - [07/Jan/2025 11:25:17] "POST /initialize HTTP/1.1" 200 -
Batches: 100%|                                                                                | 1/1 [00:00<00:00, 72.47it/s]
127.0.0.1 - - [07/Jan/2025 11:25:43] "GET /ask?query=how%20are%20you HTTP/1.1" 200 -
```

VECTOR EMBEDDINGS – FAISS

```
-7.99922273e-02  9.51894745e-02  2.87396330e-02 -9.31561962e-02
 7.81395286e-02  1.51837226e-02 -5.32930307e-02  3.73006128e-02
-9.28860717e-03  2.55008438e-03  1.39783412e-01 -2.36369576e-02
 3.16954330e-02  6.42700046e-02 -3.68429795e-02  5.94265349e-02
-1.22152433e-01  8.72031972e-02  4.21575978e-02  8.06708410e-02
-7.81026259e-02 -3.75187546e-02 -2.86708195e-02 -6.59003258e-02
 2.09816098e-02 -3.38083766e-02 -3.14570926e-02 -8.87605101e-02
-2.52922028e-02 -9.50792059e-03 -6.75161257e-02  8.69221911e-02
 1.07280679e-01 -1.81774683e-02 -4.47932035e-02 -2.53155660e-02
 2.06599338e-03 -9.35200676e-02  4.61754709e-04  2.03861427e-02
 4.66542244e-02  6.24116583e-05 -1.74690410e-02  4.70426567e-02
 2.41117738e-02 -4.25333995e-03  8.14483911e-02 -1.30149465e-08
-1.28905512e-02 -4.07029390e-02  1.94521081e-02 -6.13717735e-02
-3.06136701e-02 -7.61105074e-03 -4.14322428e-02  3.81088024e-03
-5.18862866e-02  3.96385863e-02  3.73026133e-02 -1.03896789e-01
-1.31597277e-02 -2.03386229e-02 -3.71955819e-02 -5.35645038e-02
-3.26641053e-02 -3.65897454e-02  7.67884254e-02  1.84700545e-02
 8.56179371e-03  3.15458863e-03  9.59504172e-02  1.67213026e-02
-1.94673128e-02  8.66204947e-02  1.59361977e-02  6.12183399e-02
 7.59519869e-03  8.39055981e-03 -9.90616065e-03  3.95239145e-02
 3.10579943e-03 -6.12430014e-02 -1.11253403e-01 -2.44702604e-02
 3.27657759e-02  2.34473720e-02 -9.74710379e-03 -7.57475523e-03
-3.65024991e-02  7.69933546e-03  5.16887009e-02 -7.08034844e-04
-3.59880738e-02  2.77542230e-02 -4.11142409e-02  5.01930937e-02
 2.67442353e-02 -4.74321730e-02 -2.80732829e-02 -1.95737667e-02
 1.52277332e-02  7.85472840e-02 -4.42028530e-02 -2.42939107e-02
-9.44590475e-03 -1.85781289e-02 -6.22452945e-02  6.01181462e-02
-3.30157718e-03  8.09450969e-02 -2.94133257e-02 -7.80349225e-02]
127.0.0.1 - - [07/Jan/2025 14:54:21] "POST /initialize HTTP/1.1" 200 -
```

AWS



Phase 2: Data Collection, Preprocessing & Integration

2.1 Data Collection & API Integration

I integrated multiple financial data sources, including:

- Stock Market Data: Alpha Vantage, Yahoo Finance (for stock prices and market trends).
- Cryptocurrency Data: CoinGecko, CryptoCompare (for real-time crypto prices).
- News APIs: NewsAPI, Newspaper3k (for financial news retrieval).
- Banking APIs: Open banking APIs such as Plaid and Yodlee (for real-time account data).
- Historical Financial Data: Collected and used for model training on trends and financial event predictions.

All of these were implemented in app3.py. Since Yahoo Finance is open-source, I used it along with Newspaper3k for news retrieval. Unfortunately, I faced some errors with Newspaper3k.

2.2 Data Preprocessing & Multi-Modal Input Handling

- Textual Data Preprocessing: Used spaCy/NLTK for Named Entity Recognition (NER), keyword extraction, and sentiment analysis.
- Graph & Image Data Processing: Integrated OCR (Optical Character Recognition) to extract text from financial reports and graphs.
- Tabular Data Processing: Used Pandas/PySpark to parse stock performance and financial statements.

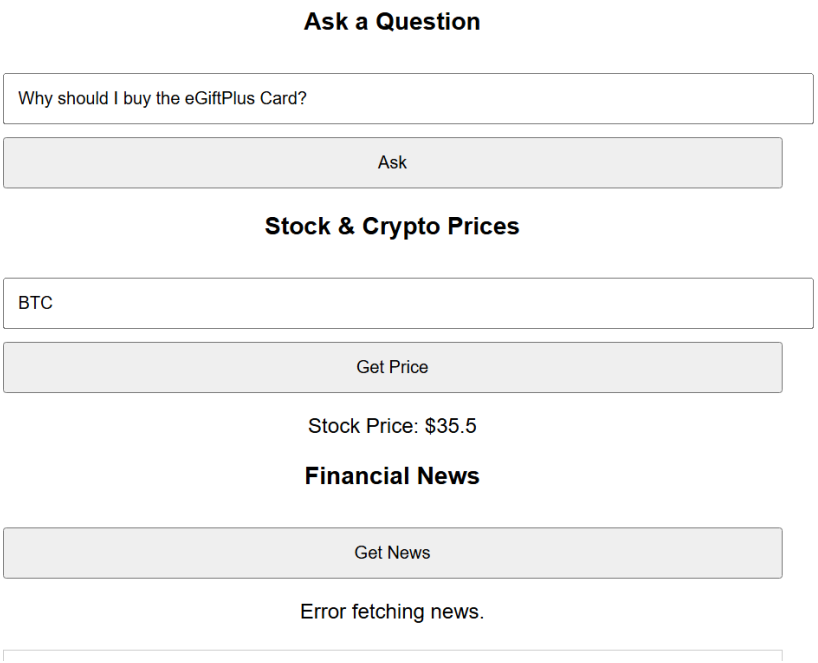
Sub-Steps:

- Data Augmentation: Used GPT-3 for synthetic financial data generation to increase dataset diversity.
- Contextual Memory: Implemented a multi-modal memory that integrates text, images, and tables into a single query context.

These were implemented in yahoo.py and app3.py, although some graph-related queries encountered errors.

The entire process involved breaking down each part, running them individually, and then integrating all elements together.

DEMO



Phase 3: Model Selection, Fine-Tuning & AI Agent Integration

3.1 Choosing the LLM for the Chatbot

- **Open-Source LLMs Used:**
  - **GPT-Neo / GPT-J** (for natural language understanding and generation).
  - **T5 / BERT** (for financial question answering).
- **Fine-Tuning:** Trained the model on financial datasets like **FinBERT** to specialize in financial terminology.

Sub-Steps:

- **Transfer Learning:** Adapted pre-trained language models for financial data.
- **Hyperparameter Tuning:** Used **Optuna** to optimize learning rate, batch size, etc.

3.2 Multi-Modal Context Retrieval with AI Agents

- **AI Agent for Data Retrieval:** Integrated AI agents to fetch real-time data from APIs, news sources, and user accounts.
- **Multi-Modal Retrieval Pipelines:**
    - **CLIP** for image-text matching.
    - **BERT** for text retrieval.
    - **Graph Neural Networks (GNNs)** for processing complex financial graphs.

I used **GPT-J and GPT-3** for the necessary phases, but some parts encountered errors.

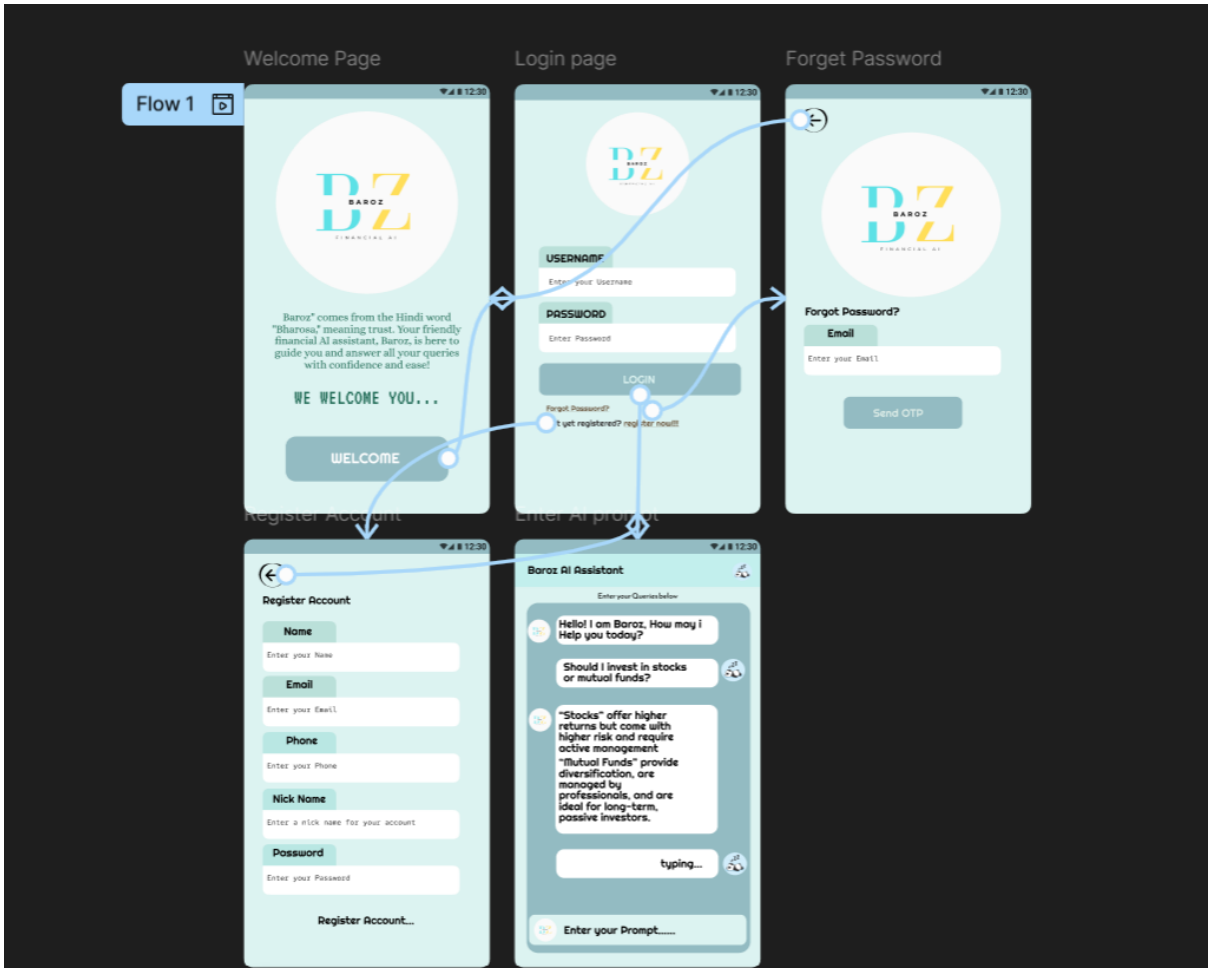
First Prototype & Vision

My first prototype provided an initial working version. I identified errors, improved functionality, and continued refining the chatbot. I had a clear vision for the next phase.

Phase 4: Frontend Integration & UI Design

For the frontend, I implemented **Vue.js**, and the compilation was successful. However, I initially faced backend connection issues with Flask, which I later resolved.

I created a **Figma layout** for the application's UI design.



VUE IMPLEMENTATION

```
 App running at:
 - Local:   http://localhost:8081/
 - Network: http://172.20.131.97:8081/

 Note that the development build is not optimized.
 To create a production build, run npm run build.

erminate batch job (Y/N)? y

haystack_env) C:\Users\lenovo\anaconda3\haystack_env\Scripts\frontend\src>npm run serve

 frontend@0.1.0 serve
 vue-cli-service serve

INFO  Starting development server...


DONE  Compiled successfully in 7093ms


 App running at:
 - Local:   http://localhost:8081/
 - Network: http://172.20.131.97:8081/

 Note that the development build is not optimized.
 To create a production build, run npm run build.
```

****error backend connection with frontend**

Phase 5: Disaster Recovery & AWS Deployment

- Containerized Deployment: Used Docker to containerize the chatbot and deploy it to AWS.
- Kubernetes for Scaling: Implemented Kubernetes for orchestrating and auto-scaling based on user demand.

Sub-Steps:

- Cloud Storage: Used AWS S3 to store historical data, model checkpoints, and logs.
- Disaster Recovery Plan: Ensured high availability with automatic failover strategies.

I successfully deployed the chatbot on AWS and maintained a disaster recovery plan by retrieving all necessary elements from my AWS account.

```
Enter Application Name
(default is "Scripts"): Scripts
Application Scripts has been created.

It appears you are using Docker. Is this correct?
(Y/n): n
Select a platform.
1) .NET Core on Linux
2) .NET on Windows Server
3) Docker
4) Go
5) Java
6) Node.js
7) PHP
8) Packer
9) Python
10) Ruby
11) Tomcat
(make a selection): 9

Select a platform branch.
1) Python 3.13 running on 64bit Amazon Linux 2023
2) Python 3.12 running on 64bit Amazon Linux 2023
3) Python 3.11 running on 64bit Amazon Linux 2023
4) Python 3.9 running on 64bit Amazon Linux 2023
5) Python 3.8 running on 64bit Amazon Linux 2 (Deprecated)
(default is 1): 2

Cannot setup CodeCommit because there is no Source Control setup, continuing with initialization
```

```
Cannot setup CodeCommit because there is no Source Control setup, continuing with initialization
Do you want to set up SSH for your instances?
(Y/n): y

Type a keypair name.
(Default is aws-eb): aws-eb
Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\lenovo\.ssh\aws-eb
Your public key has been saved in C:\Users\lenovo\.ssh\aws-eb.pub
The key fingerprint is:
SHA256:Cersb4S8aWizP2T7gi9Bcsnjkk+YC9thz1th0LVDQ0k aws-eb
The key's randomart image is:
+--[ED25519 256]--+
|       oE.       |
|      . o.o      |
| . .. o o        |
|. *  o . o       |
| O o..o S        |
|* *o=...         |
|.O Xo=.          |
|o O.X..          |
| ..B=Bo          |
+----[SHA256]-----+
```

```
2025-03-14 19:57:33,492 (ERROR) ebcli.lib.aws : Botocore Error
Uploading: [#########################################----------] 77% 2025-03-14 19:58:12,829 (ERROR) ebcli.lib.aws : Boto
core Error
Uploading: [##############################################---------] 82% 2025-03-14 20:00:39,986 (ERROR) ebcli.lib.aws : Boto
core Error
Uploading: [###############################################-------] 86% 2025-03-14 20:01:23,940 (ERROR) ebcli.lib.aws : Boto
core Error
Uploading: [################################################-----] 91% 2025-03-14 20:02:17,496 (ERROR) ebcli.lib.aws : Boto
core Error
Uploading: [##################################################--] 95% 2025-03-14 20:10:30,662 (ERROR) ebcli.lib.aws : Boto
core Error
Uploading: [##################################################] 100% Done...
Environment details for: flask-app
  Application name: Scripts
  Region: us-east-1
  Deployed Version: app-250314_190636071399
  Environment ID: e-5p6t5suj2a
  Platform: arn:aws:elasticbeanstalk:us-east-1::platform/Python 3.12 running on 64bit Amazon Linux 2023/4.4.1
  Tier: WebServer-Standard-1.0
  CNAME: UNKNOWN
  Updated: 2025-03-14 14:40:40.948000+00:00
Printing Status:
2025-03-14 14:40:38    INFO    createEnvironment is starting.
2025-03-14 14:40:41    INFO    Using elasticbeanstalk-us-east-1-767397879015 as Amazon S3 storage bucket for environment
 data.
2025-03-14 14:41:14    INFO    Created security group named: sg-0149f8b9bc3c54c4e
2025-03-14 14:41:30    INFO    Created target group named: arn:aws:elasticloadbalancing:us-east-1:767397879015:targetgro
up/awseb-AWSEB-ETLWVZY7NLKL/d51bedb1d0332a38
2025-03-14 14:41:30    INFO    Created security group named: awseb-e-5p6t5suj2a-stack-AWSEBSecurityGroup-dvffh2lEQ6rr
```

Phase 6: Documentation & Testing

Unit Testing:

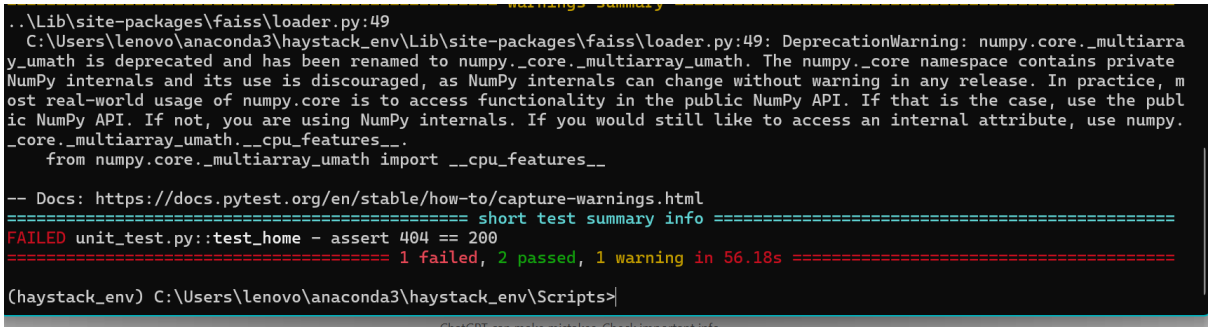
- Installed pytest.
- Created test_unit.py, which contained basic queries like “What is finance?” to validate chatbot responses.

Integration Testing:

- Used Postman API for testing API endpoints.

Model Drift Detection:

- Implemented techniques to track changes in model performance over time.



This Task was both challenging and exciting, and I look forward to making further improvements!