

# **Rutgers, The State University Of New Jersey**

## **DEPARTMENT OF COMPUTER SCIENCE**



### **Project Report: FinGPT**

Under the Guidance of Professor: Xintong Wang

Topics in AI: Market Design (CS-598)

Project Team Members:

Dheekshith Dev Manohar Mekala(dm1653)

Mohanti Reddy Bommu(mb2210)

Nishka Reddy Gorrai(ng683)

**Abstract:**

This project intends to solve key investment difficulties, specifically investor anxiety and uncertainty. FinGPT uses cutting-edge AI technologies to provide individualized reassurance and investing recommendations based on individual risk tolerances and psychological profiles. This is accomplished using a sophisticated machine learning model that evaluates investor behavior and tailors communication effectively. Furthermore, the method promotes direct connection between investors and organizations by alerting companies when an investor expresses interest, so increasing the transparency and efficiency of investor-company interactions. The project's significance stems from its ability to improve investor confidence and decision-making quality in a turbulent market by harnessing insights from recent advances in AI and machine learning. This approach not only promises to improve investment outcomes but also aims to foster a more informed and less stressful investment environment.

**Introduction:**

When it comes to making investment decisions in today's volatile financial markets, people frequently find themselves paralyzed by fear. Traditional investment techniques sometimes fail to take into consideration individual investors' unique psychological profiles and risk tolerances. This oversight can result in inferior decision-making, significantly raising investor stress and anxiety. Furthermore, the widespread lack of personalization and ineffective communication between investors and investment firms pose significant impediments to effective investment management. These concerns appear in a variety of negative behaviors, including panic selling during market downturns and insufficient portfolio diversification, which can jeopardize financial outcomes and overall portfolio stability.

To address these difficulties, innovative technology must be integrated that can customize investment plans to individual needs and enable real-time, effective communication between investors and corporations. Such advances would not only improve decision-making processes, but would also serve to reduce the psychological stress associated with investing, resulting in a healthier and more productive investment environment.

**Significance:**

The FinGPT project is significant because of its ability to provide personalized investment recommendations based on individual investors' needs and risk tolerances. This tailored approach reduces uncertainty and increases investor confidence, particularly during volatile market conditions. FinGPT enables investors to make informed and confident financial decisions by giving personalized advice.

**Personalized Risk Tolerance Assessment:**

Risk tolerance is the level of risk, including financial loss, an individual is comfortable taking with their investment portfolio. It's influenced by a combination of factors including the investor's financial goals, time horizon, life stage, personality, and past experiences. Properly assessing risk tolerance helps in tailoring investment strategies that an individual is likely to stick with, even in market downturns.

To address investors' concerns and enhance their confidence, the project features personalized reassurance prompts. These prompts are tailored to individual risk tolerance levels, ensuring that communication is specifically aligned with each investor's comfort with risk.

**Data:****1. Psychological Evaluation Data:**

By employing scenario-specific questionnaires, we examine the behavioral traits and tolerance levels of participants. This data, encompassing demographics, financial goals, investment history, current status, and risk tolerance, is vital for customizing reassurance messages and investment advice.

**2. Investors Personal Data:**

This information encompasses investor's personal data such as names, email addresses, passwords and additional personal details.

**3. Market Data:**

Utilizing real-time and historical investment data on mutual funds aids in evaluating current market trends, assessing investment performances, and formulating strategic diversification recommendations.

**Method:***User data format: JSON**Companies format: JSON**External stocks retrieval api: Polygon.IO*

1. OpenAI Tools: We utilized generative pre-trained transformer (GPT) models for natural language processing tasks. These models are designed to understand and respond to investor inquiries effectively, simulating a conversational interface that can engage users in a meaningful and contextually appropriate manner.

2. FinGPT **AssistantManager** Class:

```

class AssistantManager:
    # Need to hardcode the ids after first run.
    assistant_id = "asst_ZpUua33VpkZL8xTKoBL94eTY"
    thread_id = "thread_UyL8mxPC2dBk7on4I07H8aYq"

    # Only setup everything but don't create assistant yet.
    def __init__(self, username, password, name, tolerance, model: str = model):
        self.client = openai.OpenAI(api_key=os.environ.get('OPENAI_API_KEY'))
        self.model = model
        self.assistant = None
        self.thread = None
        self.run = None
        self.vector_store = None
        self.file_embed = None
        self.file_companies = None
        # User Variables
        self.username = username
        self.password = password
        self.name = name
        self.tolerance = tolerance

        if AssistantManager.assistant_id:
            self.assistant = self.client.beta.assistants.retrieve(assistant_id=AssistantManager.assistant_id)
        if AssistantManager.thread_id:
            self.thread = self.client.beta.threads.retrieve(thread_id=AssistantManager.thread_id)

        if not (self.assistant and self.thread):
            # Create a vector store called "Financial Docs"
            vector_store = self.client.beta.vector_stores.create(name="Clients Financial Data")

```

- a. This is the **AssistantManager** class. It takes care of the assistant's creation as well as its abstracted functions that are required for model running and thread appending. It has the following functions -

**create\_assistant(), create\_thread(), initiate\_run(), add\_msg\_to\_thread(), process\_msgs(), call\_required\_functions(), wait\_for\_completed, run\_steps, get\_current\_stocks().**

- i. **Create\_assitant():** Creates the assistant api object.
- ii. **Create\_thread():** Creates the thread. Thread is an object where it holds the entire conversation. Each response from the user and the assistant is a thread.
- iii. **Initiate\_run():** Object to call (run) the model. Whenever the user types in his request and presses 'Return' key, this function is called.
- iv. **Add\_msg\_to\_thread():** This function adds the responses from both the user and the assistant to the thread.
- v. **Process\_msgs():** This function is used for extracting the text content (string type) from the message as all responses are returned in JSON format. It also discerns between text type (string) messages and image type (image\_file) type and invokes other methods that are required to process these types.
- vi. **call\_required\_functions():** This function is dynamically invoked by the assistant when it needs responses from external functions. The way OpenAI api works is, the model invokes external functions and pauses execution thread. After that external method is invoked, the method has to execute its contents and return some value (our case stock prices of other companies from Polygon.IO api), now this value has to be plugged back into the model run time which is currently on pause and after receiving this values the openai api continues model execution to the end.

```

def call_required_functions(self, required_actions):
    if self.run:
        tool_outputs = []
        for action in required_actions['tool_calls']:
            func_name = action['function']['name']
            arguments = json.loads(action['function']['arguments'])

            if func_name == 'get_current_stocks':
                output = self.get_current_stocks(company=arguments['company'])
                # print("OUTPUT::: ", output)
                tool_outputs.append({"tool_call_id": action['id'], "output": str(output)})
            else:
                # If func_name other than get_stocks()
                raise ValueError(f"Unknown Function {func_name}")

        print("Submitting required back to the Assistant...")
        if tool_outputs:
            try:
                self.client.beta.threads.runs.submit_tool_outputs_and_poll(thread_id=self.thread.id, run_id=self.run.id, tool_outputs=tool_outputs)
                # self.add_msg_to_thread(role="assistant", content="Please analyze the JSON data returned by the get_current_stocks() API.")
                # print("Tool outputs submitted successfully.")
                logging.info("Tool outputs submitted successfully.")
            except Exception as e:
                # print("Failed to submit tool outputs:", e)
                logging.warning(f"Failed to submit tool outputs: {e}")
        else:
            print("[WARN] No tool outputs to submit.")
            logging.warning("No tool outputs to submit.")

```

- vii. **get\_current\_stocks()**: This is the external stocks retrieval api function invoked dynamically by the assistant api. This function when invoked returns the actual stock values in JSON format. This format should be converted into plain text for the model to display it to the user on its thread.

```

def get_current_stocks(self, company):
    stocks_api_key = os.environ.get('STOCKS_API_KEY')
    stocksTicker = company
    multiplier = 1
    timespan = "month"
    from_ = "2024-04-20"
    to_ = "2024-05-25"
    limit = 200
    url = (
        f"https://api.polygon.io/v2/aggs/ticker/{stocksTicker}/range/{multiplier}/{timespan}/{from_}/{to_}?adjusted=true&sort=asc&limit={limit}"
    )
    url_2 = (
        f"https://api.polygon.io/v3/reference/tickers?ticker=AAPL&market=stocks&active=true&apiKey={stocks_api_key}"
    )

    try:
        response = requests.get(url)
        if response.status_code == 200:
            # print("RESPONSE::: ", response.json(), type(response.json()))
            # data = json.dumps(response.json(), indent=4)
            # print("DATA::: ", data, type(data))
            stocks = response.json()
            # ticker = stocks["ticker"]
            # results = stocks["results"] # List
            # status = stocks["status"]
            # request_id = stocks["request_id"]
            return stocks
        else:
            return []

    except requests.exceptions.RequestException as e:
        print("Get current stocks exception: ", e)

```

- viii. **run\_steps()**: This function is used for debugging. It displays all the list of threads up until its invocation.

3. The next code deals with converting user and companies .json files to vectors and store it in openai's vector store. For information retrieval from files, it has to be converted into word embeddings which the vector store does. An embedding is a vector (list) of floating point numbers. The distance between two vectors measures their relatedness. Small distances suggest high relatedness and large distances suggest low relatedness.

```

if not (self.assistant and self.thread):
    # Create a vector store called "Financial Docs"
    vector_store = self.client.beta.vector_stores.create(name="Clients Financial Data")
    # Ready the files for upload to OpenAI
    file_paths = [filepath_1, filepath_2]
    file_streams = [open(path, "rb") for path in file_paths]
    file_batch = self.client.beta.vector_stores.file_batches.upload_and_poll(vector_store_id=vector_store.id, files=file_streams)
    # You can print the status and the file counts of the batch to see the result of this operation.
    print("[INFO] File Batch Status:::", file_batch.status)

    self.vector_store = vector_store

# Attaching files to the assistant
with open(filepath_1, "rb") as file:
    response = self.client.files.create(file=file.read(), purpose="assistants")
self.file_embed = response
with open(filepath_2, "rb") as file:
    response = self.client.files.create(file=file.read(), purpose="assistants")
self.file_companies = response

```

4. Here is a snippet of the user data in JSON format:

```

{
  "users": [
    {
      "name": "Jessie",
      "username": "jessie98",
      "password": "Pass@123",
      "age": 26,
      "trait": "Active",
      "tolerance": 1.0,
      "investments": [
        {
          "company": "Tesla",
          "amount": 600
        },
        {
          "company": "Apple",
          "amount": 1100
        },
        {
          "company": "Meta",
          "amount": 900
        }
      ]
    }
  ],

```

5. Here is a snippet of the CSS code used for the login page.

```
css = """
<style>
  html, body, [class*="css"] {
    margin: 0;
    padding: 0;
    height: 100%;
    background-color: teal !important;
  }
  .container {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100%;
  }
  .login-box {
    background-color: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0,0,0,0.1);
    width: 300px;
  }
  .stTextInput>div>div>input {
    padding: 10px;
    border-radius: 5px;
    border: 1px solid #ccc;
  }
  .stButton>button {
    width: 100%;
    padding: 10px;
    border-radius: 5px;
    border: none;
  }

```

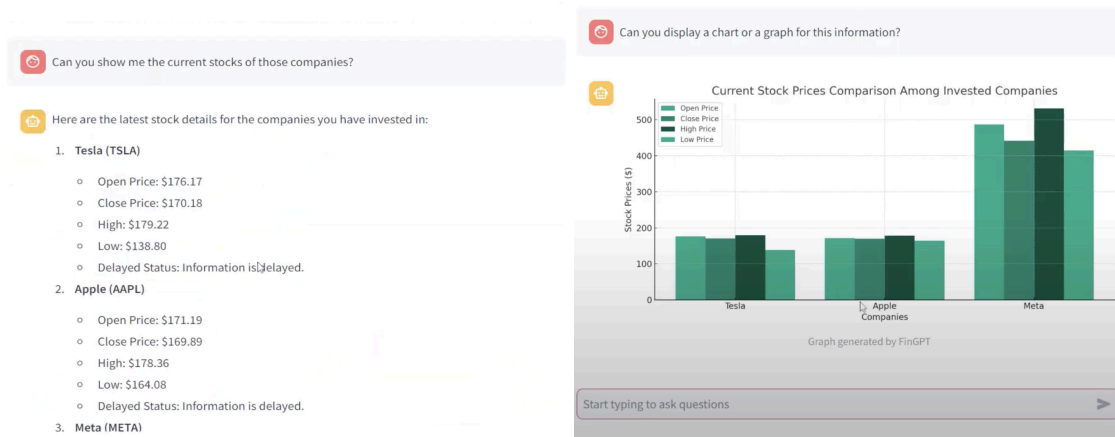


**Results:**

ChatGPT	FinGPT
<p><b>User:</b> Hi, what is my name?</p> <p><b>ChatGPT:</b> I can't tell you your name since I don't have access to personal information unless you've shared it with me in our conversation. If you've mentioned your name before, feel free to remind me! Otherwise, feel free to give me a name you'd like to go by.</p>	<p><b>User:</b> Hi, what is my name?</p> <p><b>FinGPT:</b> Your name is Jessie. If you have any more questions or need assistance with anything else, feel free to let me know!</p>
<p><b>User:</b> What companies did I invest in?</p> <p><b>ChatGPT:</b> I'm sorry for any confusion, but I don't have access to personal data about individuals unless it's shared with me in the course of our conversation. This includes information about investments or financial matters. However, I can provide general advice or information about investing if you're interested!</p>	<p><b>User:</b> What companies did I invest in?</p> <p><b>FinGPT:</b> You have invested in the following tech companies:</p> <ol style="list-style-type: none"> <li>1. Tesla (TSLA)</li> <li>2. Apple (AAPL)</li> <li>3. Meta (META)</li> </ol> <p>These companies are part of your portfolio managed by Arthur.Inc. If you need more detailed information about your investments or have any other questions, feel free to ask!</p>
<p><b>User:</b> Can you give me current stock price of Tesla?</p> <p><b>ChatGPT:</b> I can't provide real-time data or browse the internet, but as of my last update, Tesla's stock price has been subject to fluctuations based on various factors such as market conditions, company performance, and news events. For the most accurate and up-to-date stock price of Tesla, I recommend checking a reliable financial news website, using a stock market app, or consulting a financial advisor.</p>	<p><b>User:</b> Can you give me current stock price of Tesla?</p> <p><b>FinGPT:</b> The current stock price of Tesla (TSLA) is as follows (please note the data is delayed):</p> <p>Most Recent Closing Price: \$183.28  Open Price: \$176.17  High: \$198.87  Low: \$138.50</p> <p>Additionally, here's another data point:</p> <p>Latest Open Price: \$182.00  Latest Close Price: \$181.19  Latest High: \$185.86  Latest Low: \$176.02</p> <p>This information provides an indication of Tesla's stock price movements. If you need a</p>

<p><b>User:</b> Can you predict stock price of Tesla for the next month?</p> <p><b>ChatGPT:</b> Predicting specific stock prices, especially for a volatile stock like Tesla, is highly uncertain and speculative. Stock prices are influenced by a multitude of factors including market sentiment, economic indicators, company performance, and global events, making accurate predictions difficult. It's important to note that stock market investing carries inherent risks, and it's generally advisable to focus on long-term investment strategies rather than short-term price predictions. If you're considering investing in Tesla or any other stock, it's a good idea to conduct thorough research, consider your investment goals and risk tolerance, and potentially consult with a financial advisor.</p>	<p>more real-time price or specific analytics, you may want to consult a live finance tracking service. If there's anything else you'd like to know or if you need further assistance, please let me know!</p> <p><b>User:</b> Can you predict stock price of Tesla for the next month?</p> <p><b>FinGPT:</b> Predicting future stock prices, particularly for a volatile and high-profile company like Tesla, involves complex financial analysis and consideration of numerous factors, such as market trends, economic indicators, company news, and technological developments. While providing a precise forecast here is beyond my current capabilities, I can suggest a general trend based on the recent stock performance and overall market conditions. Considering Tesla's recent stock price fluctuations and assuming no significant negative market or company-specific events, Tesla's stock could potentially either maintain a stable trend or experience an increase, reflecting investor confidence in the tech and automotive sectors. However, Tesla's stock is known for its volatility, influenced by market trends, regulatory news, and Elon Musk's public statements. For a more structured prediction, consulting financial analysts or using sophisticated financial modeling software would be beneficial. These resources can provide a more comprehensive analysis based on detailed and current financial data. If you have access to investment advisory services or advanced financial analysis tools, you might get a more precise forecasting.</p>
---	--

## Sample Graph:



## Strategic Aspects:

**Investment Advice Personalization:** The project's strategic goal is to provide investors with investment advice that is specifically catered to their unique risk tolerance and financial objectives. By meeting their specific needs, this strategy aims to increase investor happiness and confidence.

**Improving direct contact between Investors and Organizations:** Encouraging direct contact between investors and the organizations they invest in is a crucial strategic component. This is accomplished through informing organizations when a person indicates interest in making additional investments, which cultivates an ecosystem of investing that is more responsive and involved.

**Application of Advanced AI and Machine Learning Technologies:** To assess investor profiles, forecast risk tolerances, and provide customized recommendations, the project cleverly makes use of state-of-the-art AI technologies, including as OpenAI's GPT models and machine learning algorithms. Because of its innovative use of cutting-edge technology, the initiative is at the forefront of financial services innovation.

**Enhancing Decision-Making During Market Volatility:** The project's strategic goal is to lessen panic selling and bad decisions made during market downturns by offering dependable guidance based on current market data and the unique characteristics of each investor. This keeps investor engagement and market stability even in erratic times.

### **Key Findings:**

Our project's primary findings show that customized reassurance reminders and tailored investment recommendations based on individual risk tolerance considerably increase investor happiness. These methods serve to reduce panic selling during market downturns and encourage informed investment decisions. Quantitative tests revealed that user engagement with the AI chatbot improved, and the recommended investment methods frequently met or surpassed benchmarks. Furthermore, customers' portfolio diversity improved significantly after following our guidance.

### **Hypothesis:**

The results from the project strongly support our original hypothesis that personalized reassurance and tailored investment advice would improve investor outcomes and satisfaction. The positive changes in investment behavior, such as reduced panic selling and more informed decision-making, further corroborate this hypothesis.

### **Merits:**

- **Personalization:** Our approach leverages advanced machine learning models to provide personalized reassurance prompts and investment suggestions tailored to each investor's risk tolerance. This personalization enhances user satisfaction by addressing individual needs and concerns, which is not commonly found in traditional investment advisories.
- **Reduced Panic Selling:** By providing targeted, informed advice during market volatility, our system helps investors make rational decisions, reducing the likelihood of panic selling. This improves overall investment outcomes and stabilizes market participation among our users.
- **Improved Investment Decisions:** With the aid of AI-driven insights and recommendations, investors are better equipped to diversify their portfolios and make decisions that align with their long-term financial goals. This is particularly beneficial for

new or anxious investors who may lack the confidence or knowledge to navigate complex market scenarios.

**Conclusion:**

In conclusion, FinGPT is designed to provide investors with customized advice and recommendations tailored to their specific risk tolerances. This approach not only reassures investors but also enhances their decision-making process and deepens their understanding of their investment choices.

**Future Scope:**

FinGPT can be integrated into banking apps, allowing the AI to provide refined, tailored advice that considers all the factors of an individual. By collecting more data over time and refining its algorithms, FinGPT could offer even more finely tuned advice that accounts for subtle nuances in investor behavior and preferences, further personalizing the user experience.

**References:**

- <https://www.nber.org/digest/202205/using-machine-learning-predict-mutual-fund-performance>
- <https://www.nber.org/papers/w29723>

**Implementation Videos:**

- <https://docs.google.com/file/d/1qtX0cwQhMknRYr1Veps04WoT4Z-SdpHB/preview>
- <https://drive.google.com/file/d/1Sjlt4eO3MJrjnW04chYhZN8J5IFK0F76/view?ts=6635b0c2>

Github link: <https://github.com/DheekshithDev>