Proposal for Python Final Project Xinyi Wang

The goal of this project is to create an analytical tool that can crawl, categorize, store, and visually present data on past U.S. stock market and funds (selecting years roughly from 2008 to 2022, depending on data accessibility), as well as sentiment data collected from social media and news channels. The core functions of the project are as follows:

Data Crawling: Automatically collect stocks, funds, and public sentiment data from financial websites and social platforms.

Data Processing: Cleanse, categorize the crawled data, and organize it into a tree structure by industry and year.

Local Caching: Since the project deals with static historical data, I will mainly rely on cached data after the first load, which will significantly improve data retrieval speed and reduce the burden on the database.

Data Presentation: Through a web interface, users can select specific times and industries to view stock price trends and sentiment analysis word clouds.

Here are the detailed steps:

Step 1: Data Crawling

Obtain data from financial websites and social platforms.

Libraries: Use requests for initiating HTTP requests, BeautifulSoup or lxml for parsing web pages, and pandas for processing and storing data.

Data Sources:

Financial Data: Crawl stock and fund data from websites like finance.yahoo.com.

Social Media Data: Plan to use APIs from social media platforms, such as api.twitter.com and reddit.com, to obtain sentiment data.

Step 2: Data Categorization, Tree Structure, and Local Caching

After data crawling, categorize the data and store it in a local cache using a tree structure.

Libraries: Use pandas for data categorization, and cachetools for data caching.

Data Categorization: Categorize stock market and sentiment data by industry and year.

Tree Structure: Construct a tree structure with "Data" as the root node, and "Financial Data" and "Sentiment Data" as sub-nodes. Further refine into "Industry Stock Market Data" and "Industry Sentiment Data", divided by year.

Local Caching: On the first run of the program, extract data from the database and store it in the local cache.

Step 3: Data Presentation and User Interaction

Finally, build a web application that allows users to query data based on time and industry and display sentiment word clouds and stock price trend charts.

Libraries: Use Flask to create a web server, and matplotlib and wordcloud to generate charts and word clouds.

Web Application: Use Flask and the HTML framework to facilitate interaction.

User Interaction: Add features to allow users to select specific times and industries.

Data Presentation: Combine user input and cached data to display stock price trend charts and generate sentiment word clouds.