Problem Statement :

To build a TikTok scraper that can retrieve fashion posts

Code:

https://github.com/harshv72/finesse-tiktok-challenge

Approach :

There are several methods to retrieve fashion posts as mentioned below:

1) Scrap accounts of trending fashion influencers on TikTok.
2) Scrap posts linked with specific hashtags related to fashion.

The issue with method-1 is that a trending influencer might be promoting a certain style of outfit. For ex. traditional. In that case, heavily relying upon such accounts can lead our program to generate a very biased dataset as output.

For such reasons, I chose to go with method-2. In this method, I chose 4 hashtags and tried to scrap 25 top posts from each hashtag. The final list of hashtags included: #fashion, #femaleoutfit, #fashionweek, #femalestreetwear.

Implementation :

Phase – 1: Extracting data using BeautifulSoup

For this challenge, I needed to create a TikTok scraper to collect TikTok posts related to fashion and containerize the system using Docker. I started by observing and analyzing the TikTok site, delving into its HTML structure, the type of framework used, and the API calls made to retrieve as much information as possible.

I discovered that the site uses a React-based JavaScript framework with hydration support (rendering some parts of the site on the server to reduce overall load time on the client). Due to this hydration step, the site provides pre-rendered data like user information, comments, and initial posts, which I later extracted using BeautifulSoup and utilized in some parts of the scraping process.

Phase – 2: Using TikTok Public APIs

Recognizing that fetching and reading from HTML is expensive, I opted for a better approach—using APIs. I found many useful APIs for different purposes, such as recommendation, search, user posts, and post comments.

However, TikTok has implemented security measures to safeguard against typical data scraping. APIs don't use user IDs in string format but rather use a secUID to identify users. While the required data for APIs may change depending on usage, some common eye-catching parameters include "ms_Token," "X-Bogus," and "_signature." "ms_Token" is a session token given by the server and is easy to obtain using cookies. It keeps updating for every API call. "X-Bogus" seems like a nonce, and "_signature" is the crypto signature of the whole request appended to the request URL after calculation.

Further investigation into the origin of API calls revealed that both "X-Bogus" and "_signature" are generated using a JS file called "webmssdk.js," which is heavily obfuscated. Depending on the state of the browser, it generates different values for each request call. To successfully request the APIs, I needed to obtain these values, with only "ms_Token" being feasible to get. Although I found a GitHub repo that could have helped generate these values, it mostly failed for me.

Phase – 3: Browser Automation using Playwright

To complete the work within a tight timeframe, I explored alternative options like browser automation. The reason for choosing this approach was that during API inspection, I discovered that the JS fetch function was hooked by another JS file, automatically adding "ms_Token," "X-Bogus," and "_signature" to the API request and calling it. However, browser automation has a few disadvantages, such as the size of the browser runtime, high memory usage, and slow startup time, even in headless mode.

Having experience with Selenium but aware of its drawbacks, I chose another library called Playwright. Playwright ships browser runtime and its dependencies for multiple OSes. The installation was straightforward, with no need for web drivers. It is also faster compared to Selenium in headless mode. I set up Playwright on a headless Chromium browser, and device emulation is set on a mobile device to avoid some captchas utilized in desktop mode.

During development, I noticed that search and comments APIs were not working even in browser automation, so I opted for an alternative method. For posts on topics related to fashion, I used hashtags to get the post, and unfortunately, I parsed the comments from the post page HTML, which only gave me the top 2 comments.

Phase – 4: Output

To convert my program into a web service, I utilized Flask, and then Docker was used to containerize the system. Docker will help with the scalability of the system. To save scraped data, I used the Pandas library to convert it into a CSV file. For the sample, I scraped the top 25 posts from four different tags: "#fashion," "#femaleoutfit," "#fashionweek," and "#femalestreetwear," which was one of the goals of this challenge. The overall scraping time for 100 posts on my MacBook on Docker was around 3-4 minutes.

Scalability and Future Improvements :

Currently, scraping is happening on 1 task - 1 thread. I would add the ability to scrape with multiple processes and multiple task threads to do more work in parallel. For comments, I'm using HTML to scrape data, which is slow, so I would fix the comment API to fetch data faster by avoiding fetching and parsing a big HTML file. This will significantly improve the scraping time and provide more comment data simultaneously. To improve the data quality of the scraping, I also need to fix the Search API.

Other possible improvement suggestions would be to utilize ML algorithms for sentiment analysis on comments and explore image recognition to enhance the identification of fashion-related content from multiple different accounts and recommendation posts. The browser instance is slow to start, which is not easy to solve, so in the future, I can move away from browser automation and go with a purely API route by figuring out a way to call APIs without any browser instance. For example, proxy servers.

Additional written questions :

1) **If you could improve one thing about FINESSE right now - as you experience it – what would it be and why?**

   From my user experience so far, I would like to suggest a few changes to the UI of the website as follows:
   - Contrast background color for outfit images
   - Font size and font weight can be improved to increase readability.
   - Dark theme enabled on the site
   - We have a best seller section in UI in which all tiles are of the same size for ranking 1-10. Maybe we can emphasize the no.1 seller Dress by showing its image as larger along with smaller images for ranking 2-10 with a scrollable list.

2) **Your plane crashes and you land on an island. What do you do? How do you plan your survival?**

Surviving on an island following a plane accident necessitates a combination of resourcefulness, adaptation, and fundamental survival abilities. Here's a broad overview of some actions to consider:

A) Examine the circumstances:
   - Inspect others and myself for wounds. As needed, provide first aid.
   - Look over the region for resources and any threats.
B) Try to generate a help signal:
   - Make a visible rescue signal using any materials I have on hand. For example, make big SOS signs for the beach.
   - Use a mirror or other reflective item to indicate ships or airplanes that are going by.
C) Build a shelter for stay:
   - Building a shelter should be my top priority if I want to keep myself safe from the weather, especially if I am somewhere with harsh weather.
   - Utilize any materials I have on hand, such as branches, leaves, or any plane debris.
D) Look for Water, Fire, and Food:
   - Locate a supply of freshwater. Look for indications of plant or animal activity if there's no evident source; these could point to the presence of water nearby.
   - Gather rainfall and, if I can, store it in containers.
   - For cooking, heating, and signaling, build a fire. Use any combustible stuff I can find, like branches or dried leaves.
   - Always have a fire going, especially at night to keep the animals away and stay warm.
   - Discover the edible plants and animals in my area by studying the flora and fauna.
   - Set traps for small animals, fish, and forage for fruits and vegetables.
E) Navigate frequently and search for human existence on the island:
   - Make reference points so I can find my way around the island. Exploration and improving my prospects of rescue depend on this.
   - Be alert to any signs of human existence on the island. If there is any human already present on that island, things could be much easier for me to survive.
F) Remain Upbeat and Intense:
   - To survive, one must have an optimistic outlook. Keeping occupied with work can assist in shielding against depressing thoughts.
   - To keep my attention on surviving, set priorities and establish a daily schedule.
G) Conserve Resources and Energy:
   - Save energy by taking breaks when necessary.
   - Try not to overwork and use my resources carefully.
H) Boost and Preserve Morale:

- If I am accompanied by others, then I will build a community. Take part in games, singing, or storytelling as ways to improve everyone's mood. Keep in mind that a variety of circumstances, such as geography, climate, and accessible resources, might affect survival scenarios.
- Being resourceful and adaptable is essential, and it's critical to regularly review and modify my survival tactics in light of my current situation.

3) What matters to you most and why?

From a career viewpoint, the most important thing to me as a recent graduate software engineer is the chance for constant learning and growth. I appreciate working in an innovative place, that promotes teamwork among experienced coworkers and offers difficult assignments that let me develop new skills. Furthermore, it is critical to have a company culture that embraces developing technologies, supports employee development, and appreciates mentorship. In my opinion, a dedication to lifelong learning guarantees a rewarding and significant career in the fast-paced profession of software engineering. It also helps me grow personally and makes the team and organization more successful and competitive.