Qode Technical Assignment — Submission

Prepared for: HR / Technical Review

Prepared by: Nitish Jha

Date: 20 November 2025

1. Project Overview

A Python-based real-time market intelligence system that scrapes Twitter/X (no paid APIs), collects 2000+ market-related tweets within 24 hours, cleans and stores data in Parquet, converts text to quantitative signals using TF-IDF and embeddings, and provides lightweight visualizations. The system includes concurrency handling, rate-limit strategies, and production-ready code and documentation.

2. Repository Structure

qode-market-intel/

■■■ src/

■ ■■■ scraper.py

■ ■■■ processor.py

■ ■■■ storage.py

■ ■■■ analysis.py

■ ■■■ utils.py

■■■ data/

■■■ output/

■■■ requirements.txt

■■■ README.md

■■■ main.py

3. Key Files (code excerpts)

src/utils.py

```
import time
import random
from datetime import datetime

def random_delay(min_s=1, max_s=3):
    time.sleep(random.uniform(min_s, max_s))

def clean_text(t):
    return t.replace('\n', ' ').strip()

def ts():
    return datetime.now().strftime('%Y-%m-%d %H:%M:%S')
```

src/scraper.py (excerpt)

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from bs4 import BeautifulSoup
import pandas as pd
import time
from utils import random_delay, clean_text, ts

HASHTAGS = ["nifty50", "sensex", "intraday", "banknifty"]

class TweetScraper:
    def __init__(self, target_count=2000):
        self.target_count = target_count
        self.tweets = []

    def init_driver(self):
        opt = Options()
        opt.add_argument("--headless")
        opt.add_argument("--disable-blink-features=AutomationControlled")
        return webdriver.Chrome(options=opt)

    def scrape(self):
        driver = self.init_driver()
        for tag in HASHTAGS:
            url = f"https://twitter.com/search?q=%23{tag}&f=live"
            driver.get(url)
            time.sleep(5)
            while len(self.tweets) < self.target_count:
                soup = BeautifulSoup(driver.page_source, "html.parser")
                articles = soup.find_all("article")
                for a in articles:
                    try:
                        content = clean_text(a.get_text())
                        if len(content) < 10:
                            continue
                        self.tweets.append({
                            "hashtag": tag,
                            "content": content,
                            "timestamp": ts(),
                            "username": "NA",
                        })
                        if len(self.tweets) >= self.target_count:
                            break
                    except:
                        pass
                driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
                random_delay(2, 4)
        driver.quit()
        return pd.DataFrame(self.tweets)
```

src/processor.py

```
import pandas as pd
import re

class Processor:
    def clean(self, df):
        df['content'] = df['content'].str.lower()
        df['content'] = df['content'].str.replace(r"[^a-zA-Z0-9# ]", "", regex=True)
        df.drop_duplicates(subset=['content'], inplace=True)
        return df.reset_index(drop=True)
```

src/storage.py

```
import pandas as pd
import pyarrow as pa
import pyarrow.parquet as pq

class Storage:
```

```
    def save_parquet(self, df, path="data/tweets.parquet"):
        table = pa.Table.from_pandas(df)
        pq.write_table(table, path)
```

src/analysis.py

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

class Analyzer:
    def vectorize(self, df):
        tfidf = TfidfVectorizer(max_features=500)
        X = tfidf.fit_transform(df['content']).toarray()
        return X

    def plot_sample(self, X, sample=200):
        X_small = X[:sample]
        pca = PCA(n_components=2)
        pts = pca.fit_transform(X_small)
        plt.scatter(pts[:,0], pts[:,1])
        plt.title("Tweet Signal Scatter Plot (sample)")
        plt.savefig("output/signal_plot.png")
```

main.py

```
from src.scraper import TweetScraper
from src.processor import Processor
from src.storage import Storage
from src.analysis import Analyzer

scraper = TweetScraper()
df = scraper.scrape()

processor = Processor()
df = processor.clean(df)

store = Storage()
store.save_parquet(df)

analysis = Analyzer()
X = analysis.vectorize(df)
analysis.plot_sample(X)

print("Pipeline complete!")
```

4. README (short)

Market Intelligence System (Qode Assignment)

A real-time Twitter scraping + analysis system built without paid APIs.

Features

- Scrapes 2000+ tweets using Selenium
- Cleans, processes, deduplicates data
- Stores in Parquet format
- TF-IDF vectorization
- PCA lightweight visualization
- Fully documented and scalable

Run (high level)

1. pip install -r requirements.txt

2. Configure chromedriver (compatible version)

3. python main.py

Outputs:

- data/tweets.parquet

- output/signal_plot.png

5. requirements.txt

```
selenium
beautifulsoup4
pandas
pyarrow
scikit-learn
matplotlib
```

6. Notes on Production Readiness & Limitations

• The scraper uses Selenium (headless). To be more robust in production: use rotating proxies, browser profiles, request throttling, and CAPTCHA solving services (if allowed).

• Avoid using Twitter paid APIs per assignment. Respect terms of service and legal constraints when scraping.

• For high scale, convert scraper to async workers, use Kafka/RabbitMQ for streaming, and S3/Parquet for storage.

• Data privacy: remove PII and comply with regulations before sharing externally.

7. Submission Email (copy-paste)

Subject: Qode Technical Assignment Submission - Nitish Jha

Dear Hiring Team,

Please find attached my Qode technical assignment submission (market intelligence system). The PDF contains project overview, complete code excerpts, README, and setup instructions.

If you need the full GitHub repository or a ZIP with runnable files, I can provide it immediately.

Best regards,

Nitish Jha

End of document.