Code Explanation Guide — Customer Feedback Sentiment Analyzer

1) High-level flow:

- User enters text in browser and clicks Analyze.
- Flask route '/predict' receives the text and calls sentiment_core.analyze(text).
- sentiment_core returns a label and score; Flask renders the result on the same page.
- API '/api/predict' allows programmatic access with JSON input.

2) Key files to show:

- sentiment_core.py : the entire logic is in a small function 'analyze' — easy to open and explain.
- flask_app/app.py : shows the Flask app and two routes ('/' and '/predict'), and an API '/api/predict'.
- templates/index.html : simple HTML where the user inputs text and sees results.

3) How to explain the core (2 minutes):

- Show sentiment_core.analyze: explain keywords lists, counting, and score formula (pos-neg)/(pos+neg).
- Mention: if no keywords found, label is Neutral and score 0.
- Explain modularity: the same core can be swapped with an ML model later, without changing the UI.

4) Interview talking points:

- "This is a demo tool to triage customer messages quickly for support teams."
- "Rule-based approach keeps outputs explainable and predictable during demos."
- "I can upgrade the core to ML later; UI and API will remain the same."