Multimodal Product Review Analysis Service

1. **Project Title:** API Service for Multimodal Product Review Analysis
2. **Project Goal/Objective:** To build and deploy an API service that analyzes product reviews containing both text and images (provided via URL or identifier) to extract combined insights (e.g., sentiment, topics, image classifications) potentially richer than text-only analysis.
3. **Problem Statement/Background:** Online reviews often contain images providing context missed by NLP-only analysis (e.g., showing defects, usage scenarios). A multimodal approach offers a more comprehensive understanding of customer feedback for better business decisions.
4. **Scope:**
    - **In Scope:** Sourcing multimodal review data; developing NLP models (sentiment, topics); developing CV models (image classification/tagging); building a REST API endpoint that accepts review text and an image reference (e.g., URL) and returns combined analysis results (e.g., JSON); containerizing the application; deploying the API to a cloud platform.
    - **Out of Scope:** Building a UI; analysis of video reviews; real-time scraping/monitoring; handling direct image uploads (unless simplified).
5. **Key Deliverables:**
    - A dataset of multimodal reviews (or documentation of sourced data).
    - Trained NLP and CV models (code and saved model files).
    - Documented API specification (e.g., OpenAPI/Swagger).
    - A containerized (Docker) REST API application.
    - Deployed API endpoint accessible via URL, capable of processing sample reviews.
    - Final report detailing methodology, model performance, API design, deployment, combined insights, and limitations.
    - Final presentation demonstrating the API service.
6. **Potential Datasets to Get Started:**
    - **Amazon Berkeley Objects (ABO):** Product listings/images; link ASINs to review datasets.
        - *Link:* https://amazon-berkeley-objects.s3.amazonaws.com/index.html
    - **Large-Scale Amazon Product Data (Stanford SNAP):** Large text review dataset with ASINs/metadata. Combine with ABO or scrape images.
        - *Link:* http://jmcauley.ucsd.edu/data/amazon/
    - **Multimodal Sentiment Analysis Datasets (Research Focused):** e.g., MVSA, check Papers With Code.
        - *Example Search:* https://paperswithcode.com/task/multimodal-sentiment-analysis
    - **Web Scraping:** Carefully scrape text/image URLs from e-commerce sites for a specific product category.
7. **Potential Technical Approach:**

- **NLP:** Pre-trained models (e.g., BERT variants via Hugging Face) fine-tuned for sentiment/topic modeling.
- **CV:** Pre-trained CNNs (e.g., ResNet, EfficientNet) fine-tuned for image classification/tagging related to reviews (defect, usage, etc.). Frameworks like TensorFlow/PyTorch.
- **API Framework:** Flask or FastAPI.
- **Containerization:** Docker. Pay attention to large model sizes and dependencies.
- **Cloud Platform:** AWS/GCP/Azure using container services (Cloud Run, Fargate, Azure Container Apps) or ML platforms (SageMaker, Vertex AI, Azure ML) which might better handle large models. Serverless functions could work if models are small enough.

8. **Deployment Requirements & Tips:**
   - **Requirement:** The final system must be deployed as a containerized REST API on AWS, GCP, or Azure. The API should accept text and an image reference (URL preferred for simplicity) and return a JSON response with structured analysis results (e.g., text sentiment, topics, image category/tags). The endpoint URL must be functional.
   - **Tips for Students:**
     - **API Design:** Design the JSON request (input) and response (output) structure clearly first. How will the image be referenced (URL is often easiest)?
     - **Local Testing:** Build and test the API locally using tools like `curl` or Postman, providing sample text and image URLs.
     - **Containerize Carefully:** CV and NLP models can be large. Ensure they are included in the Docker image correctly. Consider:
       - Downloading models on container start-up (adds delay but keeps image smaller).
       - Using cloud provider model registries if using their ML platforms.
       - Optimizing/pruning models if possible.
     - **Handling External URLs:** The API needs to fetch the image from the provided URL. Use libraries like `requests` in Python. Handle potential errors (URL invalid, image not found).
     - **Cloud Service Choice:** Container platforms (Cloud Run, Fargate, Azure Container Apps) are generally robust for potentially large model containers. ML Platforms offer more ML-specific features but might be more complex to set up initially. Serverless functions may hit size/memory limits.
     - **Dependencies:** Ensure all libraries (ML frameworks, CV libs, NLP libs, API framework, image fetching libs) are in `requirements.txt` and installed in the Docker container.
     - **Resource:** Use documentation for chosen libraries, frameworks, Docker, and cloud deployment services. Leverage `Cloud Computing for ML`, `NLP`, `CV` course content.

9. **Potential Challenges:** Acquiring/linking multimodal data; combining insights meaningfully; large model sizes impacting deployment; handling image fetching errors; API performance.
10. **Success Metrics:** Deployed API functionality and reliability; performance of NLP/CV models; quality/depth of combined analysis; clarity of API documentation; successful demonstration.
11. **Team Size / Duration:** 3-4 Students / 14 Weeks