**I. Tổng Quan Vai Trò và Mục Tiêu Cụ Thể**

**1. Vai Trò Chính**

* **Tích hợp các thành phần của dự án:**
  + **Dữ liệu:** Nhận dữ liệu sau tiền xử lý (file CSV/JSON), đảm bảo dữ liệu đã được làm sạch, tokenized và có nhãn đầy đủ.
  + **Model phân loại:** Nhận các model artifacts, code, và tham số đánh giá.
  + **Kết quả EDA:** Nhận các báo cáo, biểu đồ, wordcloud, thống kê từ quá trình EDA.
* **Xây dựng API và giao diện người dùng (UI):**
  + **API:** Tạo ra các endpoint để nhận dữ liệu từ phía người dùng, gọi mô hình dự đoán và trả về kết quả dưới dạng JSON.
  + **UI:** Xây dựng giao diện người dùng cho phép nhập review, hiển thị kết quả dự đoán (nhãn, xác suất) và các biểu đồ, dashboard trực quan.
* **Triển khai ứng dụng:**
  + Kiểm tra tính ổn định, bảo mật, hiệu suất của hệ thống tích hợp.
  + Deploy ứng dụng lên nền tảng đám mây (Heroku, Streamlit Sharing, VPS, …).
* **Tài liệu hóa & báo cáo tích hợp:**
  + Soạn thảo tài liệu hướng dẫn cài đặt, cấu hình, chạy ứng dụng và bảo trì.
  + Chuẩn bị báo cáo kỹ thuật, slide trình bày tổng kết dự án.

**2. Mục Tiêu Cụ Thể**

* **Tích hợp liền mạch:** Đảm bảo rằng các module (dữ liệu, model, EDA) giao tiếp hiệu quả thông qua API và giao diện người dùng.
* **Trải nghiệm người dùng mượt mà:** Giao diện trực quan, dễ sử dụng, hiển thị kết quả dự đoán và dashboard hấp dẫn.
* **Hiệu suất và bảo mật:** Hệ thống phản hồi nhanh, bảo mật tốt, có khả năng mở rộng khi triển khai ở quy mô lớn.
* **Tài liệu hóa chi tiết:** Mọi quy trình, đoạn code và cấu hình đều được ghi chép tỉ mỉ, dễ hiểu cho việc bảo trì và chuyển giao.

**II. Lộ Trình Chi Tiết 1 Tháng**

**Tuần 1: Nghiên Cứu, Lập Kế Hoạch & Thiết Lập Môi Trường Triển Khai**

**1.1. Hiểu Rõ Các Thành Phần Dự Án và Vai Trò**

* **Tiếp nhận thông tin từ các thành viên khác:**
  + **Dữ liệu:** Nhận file dữ liệu sau tiền xử lý (CSV/JSON). Kiểm tra các trường cần thiết: review\_text, tokens, label, và các metadata bổ sung (tiêu đề, ngày đăng, …).
  + **Model phân loại:** Nhận model artifacts và code; lưu ý định dạng file (ví dụ: .pkl, .joblib) và cách load model.
  + **Kết quả EDA:** Nhận báo cáo, biểu đồ (histogram, boxplot, wordcloud) và các thống kê trực quan từ quá trình EDA để hiển thị trên dashboard.
* **Xác định mục tiêu tích hợp:**
  + **Mục tiêu chính:** Xây dựng hệ thống demo cho phép nhập review, trả về dự đoán từ mô hình và hiển thị các biểu đồ EDA.
  + **Luồng dữ liệu:** Dữ liệu đi từ giao diện người dùng → API backend (xử lý, gọi model) → trả về kết quả → hiển thị trực quan.
  + **Tiêu chí thành công:** API phản hồi nhanh (dưới 1-2 giây), giao diện thân thiện, dashboard trực quan và dễ hiểu.
* **Những kiến thức cần nghiên cứu:**
  + Cách định dạng dữ liệu JSON, cách truyền và nhận dữ liệu giữa API và UI.
  + Quy trình tích hợp các module độc lập thành hệ thống thống nhất.
  + Nguyên tắc thiết kế giao diện người dùng tối giản nhưng hiệu quả.

**1.2. Thiết Lập Môi Trường Phát Triển & Cấu Trúc Dự Án**

* **Tạo môi trường phát triển:**
  + **Virtual Environment:** Sử dụng virtualenv hoặc conda để tạo môi trường riêng, đảm bảo không bị xung đột thư viện.
  + **Cài đặt các thư viện:**
    - **Backend:** Flask hoặc FastAPI, joblib/pickle để load model.
    - **Frontend:** Nếu dùng Streamlit, cài đặt streamlit; nếu dùng Flask thì cần cài đặt các thư viện hỗ trợ template (Jinja2, Flask-CORS,...).
    - **Visualization:** Plotly, matplotlib, seaborn.
  + **Cập nhật requirements.txt:** Ghi rõ phiên bản của từng thư viện để đảm bảo tính tái tạo.

**Cấu trúc thư mục dự án (ví dụ):**  
/project\_root

  ├── src/

  │    ├── api/           # Code API (ví dụ: api.py, config.py)

  │    ├── ui/            # Code giao diện người dùng (ví dụ: app.py, templates/, static/)

  │    ├── deployment/    # Mã nguồn tích hợp, file chạy tổng hợp

  ├── data/

  │    ├── processed/     # Dữ liệu đã qua tiền xử lý

  ├── docs/              # Tài liệu hướng dẫn, báo cáo, slide

  ├── requirements.txt   # Danh sách thư viện cần thiết

  └── Procfile           # Nếu deploy trên Heroku

* **Lập kế hoạch tính năng cần triển khai:**
  + **Các tính năng chính:**
    - Form nhập review và nút “Submit”.
    - Khu vực hiển thị kết quả dự đoán từ API.
    - Dashboard hiển thị các biểu đồ trực quan từ kết quả EDA.
  + **Xác định các API endpoint:**
    - Endpoint /predict nhận dữ liệu review và trả về dự đoán.
    - (Tùy chọn) Endpoint lấy thông tin thống kê hoặc cập nhật dashboard.
* **Kiến thức cần học:**
  + Xây dựng RESTful API với Flask/FastAPI.
  + Các tiêu chuẩn định dạng JSON và cách xử lý dữ liệu trong Python.
  + Các framework UI đơn giản như Streamlit hoặc xây dựng giao diện web với HTML/CSS/JavaScript.

**1.3. Giao Tiếp và Phân Chia Công Việc Nội Bộ**

* **Họp nhóm:**
  + Tổ chức buổi họp để trao đổi về định dạng dữ liệu giữa các module.
  + Thống nhất chuẩn giao tiếp giữa API và UI (ví dụ: cấu trúc JSON trả về của endpoint /predict).
* **Lập kế hoạch timeline:**
  + **Danh sách công việc chi tiết:** Ghi rõ từng bước cần hoàn thành và ước tính thời gian (giờ/ngày).
  + **Milestone chính:**
    - Hoàn thiện API và kiểm thử cục bộ.
    - Xây dựng giao diện UI và kết nối với API.
    - Tích hợp end-to-end và kiểm thử toàn bộ hệ thống.
  + **Công cụ quản lý:** Sử dụng Trello, Asana hoặc Google Sheets để theo dõi tiến độ và giao tiếp nội bộ.

**Tuần 2: Phát Triển API & Giao Diện Người Dùng**

**2.1. Xây Dựng API Cho Dự Đoán (Backend)**

* **Thiết kế kiến trúc API:**
  + **Endpoint chính /predict:**
    - **Đầu vào:** JSON với trường "review", ví dụ: { "review": "Đây là review của sản phẩm..." }.
    - **Đầu ra:** JSON chứa kết quả dự đoán, ví dụ: { "label": "tốt", "confidence": 0.93 }.
    - **Quy trình xử lý:** Nhận dữ liệu → (tiền xử lý nếu cần) → gọi model dự đoán → trả kết quả.
* **Viết mã API:**
  + **Cấu trúc file:** Tạo file api.py trong thư mục src/api/.

**Ví dụ mã mẫu (sử dụng Flask):**  
from flask import Flask, request, jsonify

import joblib

import traceback

app = Flask(\_\_name\_\_)

# Load model khi khởi động để tránh load lại mỗi request

try:

    model = joblib.load("path/to/model.pkl")

except Exception as e:

    app.logger.error("Error loading model: {}".format(e))

    model = None

@app.route('/predict', methods=['POST'])

def predict():

    data = request.get\_json(force=True)

    review\_text = data.get("review", "")

    try:

        # Nếu cần, thực hiện bước tiền xử lý tương tự như đã dùng cho model

        # result = model.predict(preprocess(review\_text))

        result = model.predict([review\_text])

        confidence = max(model.predict\_proba([review\_text])[0])

        return jsonify({"label": result[0], "confidence": confidence})

    except Exception as e:

        app.logger.error(traceback.format\_exc())

        return jsonify({"error": str(e)}), 500

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True, port=5000)

* **Cấu hình bổ sung:** Tạo file config.py nếu cần chứa các thông số như PORT, DEBUG flag,...
* **Kiểm thử API cục bộ:**
  + Dùng Postman hoặc curl để gửi request mẫu và xác minh kết quả.
  + Kiểm tra các trường hợp ngoại lệ: review trống, lỗi tiền xử lý, model không load được,...

**2.2. Xây Dựng Giao Diện Người Dùng (Frontend)**

* **Thiết kế giao diện:**
  + **Phác thảo wireframe:** Vẽ sơ đồ giao diện gồm:  
      
    - **Phần nhập liệu:** Text area cho review.
    - **Nút “Submit”:** Khi nhấn gửi, dữ liệu sẽ được chuyển tới API.
    - **Khu vực hiển thị dự đoán:** Hiển thị label dự đoán và xác suất.
    - **Dashboard trực quan:** Hiển thị các biểu đồ từ EDA (ví dụ: histogram độ dài review, wordcloud từ review “tốt” và “xấu”).
* **Lập trình giao diện:**
  + **Nếu sử dụng Streamlit:**

Tạo file app.py trong thư mục src/ui/ với nội dung mẫu:  
  
import streamlit as st

import requests

import json

import plotly.express as px

import pandas as pd

st.title("Demo Phân Loại Review Sản Phẩm")

review\_input = st.text\_area("Nhập nội dung review của bạn:")

if st.button("Submit"):

    if review\_input:

        # Gọi API từ Streamlit

        try:

            response = requests.post("http://localhost:5000/predict", json={"review": review\_input})

            result = response.json()

            if "error" in result:

                st.error("Lỗi từ API: " + result["error"])

            else:

                st.success(f"Dự đoán: {result['label']} (Xác suất: {result['confidence']:.2f})")

        except Exception as e:

            st.error("Lỗi kết nối API: " + str(e))

    else:

        st.warning("Vui lòng nhập review trước khi gửi.")

# Phần dashboard: Ví dụ, hiển thị biểu đồ từ dữ liệu EDA đã được cung cấp

st.subheader("Dashboard Trực Quan")

# Giả sử có file CSV chứa dữ liệu EDA

try:

    df = pd.read\_csv("data/eda\_statistics.csv")

    fig = px.histogram(df, x="length", title="Phân bố độ dài review")

    st.plotly\_chart(fig)

except Exception as e:

    st.error("Không thể tải dashboard: " + str(e))

* **Nếu sử dụng Flask:**
  + Xây dựng giao diện bằng HTML/CSS trong thư mục templates/ và tích hợp JavaScript (AJAX) để gửi request đến API.
  + Ví dụ: Tạo file index.html và sử dụng jQuery để gọi API khi nút submit được nhấn.
* **Kết nối UI với API:**
  + Viết hàm trong UI để gửi dữ liệu nhập vào API /predict và nhận kết quả.
  + Xử lý dữ liệu trả về và cập nhật giao diện (ví dụ: hiển thị thông báo, cập nhật biểu đồ).D
* **Kiểm thử giao diện cục bộ:**
  + Chạy ứng dụng, nhập dữ liệu mẫu, kiểm tra tính đúng đắn của kết quả dự đoán và dashboard.
  + Debug từng thành phần: form nhập liệu, nút submit, hiển thị kết quả, cập nhật dashboard.

**Tuần 3: Tích Hợp, Kiểm Thử & Tối Ưu Hệ Thống**

**3.1. Tích Hợp Các Module**

* **Kết nối các thành phần:**
  + Đảm bảo rằng khi người dùng nhập review từ UI, dữ liệu được gửi qua API, model dự đoán xử lý và kết quả trả về được hiển thị đúng trên giao diện.
  + Kiểm tra luồng dữ liệu từ UI → API → Model → API → UI. Vẽ sơ đồ luồng dữ liệu để tham khảo.
* **Viết mã tích hợp bổ sung (nếu cần):**
  + Tạo các hàm chuyển đổi dữ liệu giữa các định dạng nếu API trả về dữ liệu chưa phù hợp với yêu cầu hiển thị của dashboard.
  + Tích hợp thêm các thành phần trực quan như biểu đồ tương tác, wordcloud từ kết quả EDA vào giao diện.
* **Kiểm thử tích hợp end-to-end:**
  + Thực hiện kiểm thử toàn bộ hệ thống: nhập review → xử lý dự đoán → hiển thị kết quả và dashboard.
  + Ghi nhận các lỗi, sử dụng log để theo dõi các request lỗi và thời gian phản hồi.

**3.2. Tối Ưu Hệ Thống và Giao Diện**

* **Tối ưu hiệu năng:**
  + Sử dụng caching để load model chỉ một lần khi ứng dụng khởi động thay vì mỗi request.
  + Tối ưu code backend bằng cách giảm thiểu các xử lý không cần thiết và dùng các thư viện tối ưu.
  + Tối ưu code giao diện: giảm thời gian tải biểu đồ, sử dụng lazy-loading nếu cần.
* **Cải thiện trải nghiệm người dùng:**
  + Thêm loading indicator trong UI khi đang chờ phản hồi từ API.
  + Hiển thị thông báo lỗi hoặc cảnh báo nếu review nhập vào không hợp lệ.
  + Tích hợp các hiệu ứng chuyển động mượt mà khi cập nhật kết quả trên giao diện.
* **Kiểm thử tính ổn định:**
  + Test với nhiều loại review khác nhau (ngắn, dài, chứa ký tự đặc biệt, tiếng Việt không dấu,…).
  + Kiểm tra tương thích trên nhiều trình duyệt (Chrome, Firefox, Edge) và trên các thiết bị (desktop, mobile).

**3.3. Chuẩn Bị Môi Trường Triển Khai**

* **Lựa chọn nền tảng triển khai:**
  + Đánh giá các lựa chọn: Heroku, Streamlit Sharing, hoặc VPS riêng.
  + Tạo file cấu hình cần thiết:  
      
    - **Procfile:** Nếu deploy trên Heroku (ví dụ: web: gunicorn api:app).
    - **runtime.txt:** Xác định phiên bản Python nếu cần.
    - **Environment variables:** Cấu hình các biến môi trường cần thiết (ví dụ: PORT, SECRET\_KEY, …).
* **Viết hướng dẫn triển khai:**
  + Soạn tài liệu chi tiết hướng dẫn deploy ứng dụng lên nền tảng đã chọn, bao gồm các bước:  
      
    - Cài đặt các gói phụ thuộc (pip install -r requirements.txt).
    - Cấu hình file Procfile, runtime.txt.
    - Đẩy mã nguồn lên GitHub và kết nối với nền tảng deploy.
  + Thực hiện triển khai trên môi trường staging để kiểm tra bảo mật, tốc độ và khả năng mở rộng.

**Tuần 4: Demo, Tài Liệu Hóa & Báo Cáo Tổng Kết**

**4.1. Hoàn Thiện Demo và Kiểm Thử Toàn Bộ Ứng Dụng**

* **Kiểm tra toàn bộ hệ thống:**
  + Chạy lại toàn bộ pipeline tích hợp từ lúc người dùng nhập review cho đến khi hiển thị dashboard.
  + Mời các thành viên khác trong nhóm kiểm tra và thu thập phản hồi chi tiết.
  + Ghi nhận các lỗi, thời gian phản hồi và các vấn đề phát sinh để tối ưu lại.
* **Sửa lỗi và tối ưu:**
  + Fix các lỗi về giao diện, lỗi backend hoặc lỗi kết nối.
  + Tối ưu lại hiệu năng ứng dụng để đảm bảo thời gian phản hồi nhanh nhất có thể.
  + Kiểm tra bảo mật ứng dụng (ví dụ: hạn chế truy cập trái phép, mã hóa dữ liệu nhạy cảm).

**4.2. Tài Liệu Hóa và Hướng Dẫn Sử Dụng**

* **Soạn thảo tài liệu hướng dẫn:**
  + Viết README chi tiết cho phần triển khai, bao gồm:  
      
    - Hướng dẫn cài đặt môi trường (virtualenv, cài đặt các thư viện).
    - Cách chạy ứng dụng (API và UI) ở môi trường cục bộ và trên nền tảng deploy.
    - Cấu trúc dự án, giải thích các thư mục và chức năng của từng module.
    - Hướng dẫn sử dụng giao diện cho người dùng cuối: cách nhập review, cách xem kết quả dự đoán và dashboard.
* **Tài liệu kỹ thuật:**
  + Soạn báo cáo kỹ thuật mô tả quy trình tích hợp chi tiết:  
      
    - Sơ đồ luồng dữ liệu giữa các module.
    - Cách xử lý ngoại lệ và tối ưu hiệu năng.
    - Các điểm cần lưu ý khi triển khai và bảo trì hệ thống.
  + Lưu tài liệu trong thư mục docs/ trên repository và cập nhật định kỳ.

**4.3. Chuẩn Bị Báo Cáo Tổng Kết & Slide Trình Bày**

* **Tổng hợp báo cáo tổng kết:**
  + Kết hợp các kết quả từ dữ liệu, model và trực quan hóa thành báo cáo chung của dự án.
  + Đưa ra các chỉ số hiệu năng: thời gian phản hồi trung bình, độ chính xác của dự đoán, mức độ hài lòng của người dùng, ….
  + Đánh giá toàn bộ quy trình tích hợp, từ việc phát triển API, xây dựng UI đến triển khai và bảo trì.
* **Chuẩn bị slide trình bày:**
  + Tạo slide tóm tắt quá trình triển khai:  
      
    - Các bước chính trong quy trình tích hợp.
    - Các tính năng nổi bật của hệ thống (form nhập review, dashboard, API dự đoán).
    - Kết quả đạt được và bài học kinh nghiệm.
  + Slide nên có hình ảnh minh họa, sơ đồ luồng dữ liệu và biểu đồ trực quan.

**4.4. Họp Nhóm Cuối Cùng và Chuyển Giao**

* **Trình bày demo cho nhóm:**
  + Trình bày toàn bộ ứng dụng, mô tả chi tiết từng module: API, giao diện người dùng, dashboard trực quan.
  + Thu thập phản hồi cuối cùng từ các thành viên khác và ghi nhận các đề xuất cải tiến.
* **Chuyển giao tài liệu & mã nguồn:**
  + Đẩy toàn bộ mã nguồn, tài liệu hướng dẫn, báo cáo và file cấu hình lên repository chung.
  + Tạo bản backup cuối cùng cho dự án (sử dụng Git, Google Drive, Dropbox,…).
  + Thông báo với các thành viên về trạng thái hoàn thiện của hệ thống và hướng dẫn bảo trì, phát triển sau này.