📚 HƯỚNG DẪN FLOW PROJECT - PHÂN TÍCH SENTIMENT FEEDBACK

🎯 Tổng quan kiến trúc

Project này là một ứng dụng web phân tích sentiment của feedback sinh viên sử dụng:

- **Frontend**: HTML + Bootstrap + JavaScript (gửi request, hiển thị kết quả)
- **Backend**: Flask (xử lý request, authentication, database)
- **Model**: PhoBERT Pair-ABSA (phân tích sentiment theo từng topic)
- **Database**: SQLite (lưu users, feedbacks) với backup tự động lên Hugging Face Hub

---

🚀 FLOW 1: KHỞI ĐỘNG ỨNG DỤNG

Khi chạy `python app.py`, ứng dụng khởi động theo các bước sau:

Bước 1: Import và khởi tạo cơ bản

File: `app.py` (dòng 1-24)

- Import các thư viện: Flask, PyTorch, transformers, SQLAlchemy, Flask-Login, các model và form
- Import config từ `model_config.py`: prompts, keywords, thresholds, helper functions
- Tạo Flask app instance và set secret key

Bước 2: Thiết lập Database và Authentication

File: `app.py` (dòng 64-82)

- Tạo đường dẫn database: `instance/feedback_analysis.db`
- Cấu hình SQLAlchemy để kết nối SQLite
- Khởi tạo Flask-Login để quản lý session user
- Đăng ký function `load_user()` để Flask-Login biết cách load user từ database

Bước 3: Load Model và Tokenizer

File: `app.py` (dòng 197-217)

- Xác định device (CPU hoặc GPU nếu có CUDA)
- Load tokenizer từ Hugging Face Hub: `Ptul2x5/Student_Feedback_Sentiment`
- Download file `model.bin` từ Hugging Face Hub (model đã được train và upload từ notebook)
- Khởi tạo model `PhoBERTPairABSA` với config từ `model_config.py`
- Load weights vào model, chuyển model sang device, set mode eval
- Nếu load thất bại → set `model = None` (app vẫn chạy được nhưng không thể phân tích)

Bước 4: Khởi tạo Scheduler Backup

File: `app.py` (dòng 43-52)

- Tạo scheduled task: backup database mỗi giờ
- Chạy scheduler trong background thread (daemon thread)
- Đăng ký backup khi app shutdown (dùng `atexit`)

Bước 5: Khởi tạo Database và Admin User

File: `app.py` (dòng 471-497)

- Tạo tables nếu chưa có (users, feedbacks)

- Kiểm tra và thêm cột `is_admin` nếu thiếu
- Nếu database trống → tạo admin user mặc định (username: `admin`, password: `123456`)
- Gọi `db_manager.initialize_database_if_needed()` để restore từ backup nếu cần

Bước 6: Start Server

File: `app.py` (dòng 636-638)

- Chạy Flask development server trên port 7860
- Server sẵn sàng nhận request

🔐 FLOW 2: ĐĂNG KÝ VÀ ĐĂNG NHẬP

Flow đăng ký (Register)

User hành động:

1. Truy cập `/register`
2. Nhập username, password, confirm password
3. Click nút "Đăng ký"

Frontend làm gì:

- File `templates/register.html`: Hiển thị form
- Validate cơ bản (required fields)

Backend xử lý:

- **File: `app.py` (dòng 224-239) - Route `/register`**
- Route `/register` nhận POST request (dòng 224)
- Nếu user đã đăng nhập → redirect về home (dòng 226-227)
- Validate form bằng `RegistrationForm` (dòng 229-230)

    - Kiểm tra username đã tồn tại chưa
    - Kiểm tra password >= 6 ký tự
    - Kiểm tra confirm password khớp
- Nếu hợp lệ (dòng 230-237):
    - Tạo User object mới (dòng 231)
    - Hash password bằng bcrypt (function `set_password()` trong `models.py`) (dòng 232)
    - Add vào database session (dòng 233)
    - Commit transaction (dòng 234)
    - Gọi `backup_database()` để backup (dòng 235)
    - Flash message thành công và redirect về `/login` (dòng 236-237)
- Nếu không hợp lệ → render template với form (dòng 239)

Flow đăng nhập (Login)

User hành động:

1. Truy cập `/login`
2. Nhập username và password
3. Click nút "Đăng nhập"

Backend xử lý:

- **File: `app.py` (dòng 241-257) - Route `/login`**
- Route `/login` nhận POST request (dòng 241)
- Nếu user đã đăng nhập → redirect về home (dòng 243-244)
- Validate form bằng `LoginForm` (dòng 246-247)
- Tìm user trong database theo username (dòng 248)
- Verify password bằng `check_password()` (so sánh với bcrypt hash) (dòng 249)
- Nếu đúng (dòng 249-253):

    - Gọi `login_user(user, remember=True)` để tạo session (dòng 250)
    - Flask-Login tự động set cookie session
    - Flash message thành công (dòng 251)
    - Redirect về `/` (home) hoặc `next` URL nếu có (dòng 252-253)
- Nếu sai → flash message lỗi (dòng 255)
- Render template với form (dòng 257)

Sau khi đăng nhập:

- User có session cookie
- Các route có `@login_required` sẽ cho phép truy cập
- `current_user` object có sẵn trong mọi route

📝 FLOW 3: PHÂN TÍCH FEEDBACK ĐƠN (SINGLE)

User hành động:

1. Đăng nhập thành công
2. Vào trang chủ `/`
3. Nhập feedback vào textarea
4. Click nút "Phân tích"

Frontend xử lý:

File: `static/js/app.js` (function `handleFormSubmit`)

1. Lấy text từ textarea
2. Validate: không được rỗng, không quá dài
3. Hiển thị loading spinner
4. Gửi POST request đến `/predict` với body: `{ "text": "feedback text" }`

Backend nhận request:

File: `app.py` (dòng 417-447) - Route `/predict`

- Route `/predict` với decorator `@login_required` (dòng 418)
- Parse JSON từ request body (dòng 421-422)
- Validate input (dòng 424-431):
  - Kiểm tra có field `text` không
  - Kiểm tra text không rỗng
  - Kiểm tra text không quá 1000 ký tự
  - Kiểm tra model và tokenizer đã load chưa
- Nếu validation fail → trả về error 400/500

Backend phân tích feedback:

File: `app.py` (dòng 84-168) - Function `analyze_feedback(text)`

Function này được gọi từ route `/predict` (dòng 433)

Bước 1: Preprocessing (dòng 86-93)

- Kiểm tra model và tokenizer có sẵn không → return `[]` nếu thiếu (dòng 86-87)
- Chuẩn hóa text: strip whitespace (dòng 89)
- Kiểm tra text có phải garbage không bằng `_is_garbage()` từ `model_config.py` → return `[]` nếu garbage (dòng 90-91)
- Normalize text để match keywords bằng `_norm_match()` từ `model_config.py` (dòng 93)

Bước 2: Inference cho từng aspect (dòng 100-115)

- Lặp qua 4 aspects: `lecturer`, `training_program`, `facility`, `others` (dòng 101)
- Với mỗi aspect:

- Map aspect EN → aspect VI bằng `ASPECT_REVERSE_MAPPING` (dòng 102)
- Lấy prompt: gọi `get_prompt(aspect_en, sentence=text, use_subprompt=True)` (dòng 103)
  - Function `get_prompt()` trong `model_config.py` (dòng 330-335) sẽ gọi `_pick_subprompt()` (dòng 322-328)
  - Check keywords trong text và match với subtopic keywords trong `SUBTOPIC_KW`
  - Trả về prompt cụ thể hoặc default prompt
- Tokenize: `tokenizer(prompt, text, ...)` (dòng 105-111) → input_ids và attention_mask
- Model inference: `model(input_ids, attention_mask)` (dòng 113) → logits [4]
- Lưu logits vào list (dòng 114)
- Check keywords: `_aspect_has_kw()` (dòng 115) → lưu vào `has_keywords`

Bước 3: Tính confidence và filter (dòng 117-136)

- Stack tất cả logits thành tensor [4, 4] (dòng 117)
- Softmax để có probabilities (dòng 119)
- Tính `conf_not_none = 1.0 - p_none` (dòng 120-121)
- **Keyword boost**: Nếu aspect có keywords → boost confidence +8% (dòng 123-127)
- **Adaptive thresholding** (dòng 129-136):
  - Lần 1: Giữ các aspects có confidence >= threshold (dòng 129)
  - Nếu đã detect được >= 1 aspect → giảm threshold 12% cho các aspects còn lại (dòng 131-136)
  - Chỉ giữ aspects mới nếu có keywords HOẶC confidence đủ cao
- Nếu không có aspect nào → return `[]` (dòng 138-139)

Bước 4: Tính sentiment cho mỗi aspect (dòng 141-165)

- Với mỗi aspect đã được giữ lại (dòng 142):
  - Lấy probabilities của sentiment, bỏ lớp none (dòng 143)
  - Tìm sentiment có probability cao nhất (dòng 144-145)
  - Tính margin = top_p - second_p (dòng 147-149)
  - **Adaptive margin filtering**: Nếu aspect có keywords → giảm margin threshold 1% (dòng 151-153)
  - Filter: chỉ giữ nếu probability >= `MIN_SENT_PROB` và margin >= `min_margin_adj` (dòng 155-156)
  - Map sentiment ID → sentiment string bằng `LABEL_MAP` (dòng 158)
  - Tạo result object với: topic, sentiment, confidence, sentiment_confidence, margin (dòng 159-165)

Bước 5: Sort và return (dòng 167-168)

- Sort results theo confidence giảm dần (dòng 167)
- Return list of results (dòng 168)

Backend lưu vào database:

File: `app.py` (dòng 170-183) - Function `save_feedback_to_db()`

- Function này được gọi từ route `/predict` (dòng 436)
- Với mỗi result (dòng 172):
  - Tạo Feedback object với: text, sentiment, topic, confidence, user_id (dòng 175-182)
  - Add vào database session (dòng 183)
- Commit transaction trong route `/predict` (dòng 437)
- Gọi `backup_database()` để backup (dòng 438)

Backend trả response:

File: `app.py` (dòng 442-445) - Route `/predict`

- Trả về JSON:

```
{
  "results": [
    {
      "topic": "lecturer",
      "sentiment": "negative",
      "confidence": 0.85,
      "sentiment_confidence": 0.82,
      "margin": 0.15
    }
  ],
  "has_multiple_topics": true
}
```

Frontend hiển thị kết quả:

File: `static/js/app.js` (function `displayMultipleResults`)

1. Parse JSON response
2. Với mỗi result → tạo card hiển thị:
    - Topic name (tiếng Việt)
    - Sentiment badge với màu (negative=đỏ, neutral=vàng, positive=xanh)
    - Confidence percentage
    - Margin
3. Hiển thị original text
4. Clear textarea
5. Reload feedback history sau 500ms

📊 FLOW 4: PHÂN TÍCH FILE CSV (BATCH)

User hành động:

1. Đăng nhập
2. Chọn file CSV chứa feedback
3. Click nút "Phân tích File CSV"

Frontend xử lý:

File: `static/js/app.js` (function `handleCsvUpload`)

1. Lấy file từ input
2. Validate: có file, extension là .csv
3. Tạo FormData và append file
4. Gửi POST request đến `/analyze-csv` với FormData
5. Hiển thị loading progress

Backend nhận file:

File: `app.py` (dòng 499-544) - Route `/analyze-csv`

- Route `/analyze-csv` với `@login_required` (dòng 500)
- Lấy file từ `request.files['csvFile']` (dòng 506)
- Validate (dòng 507-516):
  - File có tồn tại không
  - Filename không rỗng
  - Extension là .csv
  - Decode được UTF-8
- Parse CSV (dòng 518-544):
  - Đọc file content
  - Tạo `StringIO` stream
  - Parse bằng `csv.DictReader`
  - Tìm cột chứa feedback (tên: feedback, text, content, comment - không phân biệt hoa thường)

    - Convert rows thành list
- Nếu validation fail → trả về error 400

Backend xử lý từng dòng:

File: `app.py` (dòng 546-612) - Route `/analyze-csv`

- Khởi tạo counters: `processed_count`, `error_count` (dòng 547-548)
- Lặp qua từng row (dòng 550):
    1. Lấy feedback text từ cột đã tìm được (dòng 551)
    2. Nếu text rỗng → đếm error, thêm vào results với error message (dòng 553-560)
    3. Nếu có text (dòng 562-612):
        - Gọi `analyze_feedback(feedback_text)` (dòng 571)
        - Lưu kết quả vào database (dùng `save_feedback_to_db()`) (dòng 574)
        - Nếu có results → lấy result đầu tiên (dòng 576-581)
        - Nếu không có results → set default (dòng 582-586)
        - Thêm vào results list (dòng 588-596)
        - Đếm processed_count (dòng 597)
    4. Nếu có lỗi → đếm error_count, thêm vào results với error message (dòng 598-612)

Backend commit và trả response:

File: `app.py` (dòng 614-628) - Route `/analyze-csv`

- Commit tất cả feedbacks vào database (dòng 615)
- Gọi `backup_database()` để backup (dòng 616)
- Nếu commit lỗi → rollback và trả error 500 (dòng 617-619)
- Trả về JSON (dòng 621-628):

```
{
  "success": true,
  "total_rows": 100,
  "processed_count": 95,
  "error_count": 5,
  "results": [...],  // Tối đa 50 rows đầu
  "message": "Đã xử lý 95/100 feedback thành công"
}
```

Frontend hiển thị kết quả:

File: `static/js/app.js` (function `showCsvResults`)

1. Parse JSON response
2. Hiển thị bảng kết quả với các cột: Row, Text, Topic, Sentiment, Confidence
3. Hiển thị statistics: tổng số, đã xử lý, lỗi
4. Reload feedback history sau 500ms

📜 FLOW 5: XEM LỊCH SỬ FEEDBACK

User hành động:

1. Đăng nhập
2. Scroll xuống phần "Lịch Sử Feedback"
3. Chọn filter (today, week, month, all, custom)
4. Click pagination để xem trang khác

Frontend gửi request:

File: `static/js/app.js` (function `loadFeedbackHistory`)

1. Lấy filter parameters từ UI
2. Build query string: `page`, `per_page`, `time_filter`, `start_date`, `end_date`

3. Gửi GET request đến `/api/feedback-history?page=1&per_page=10&time_filter=all`

Backend xử lý:

File: `app.py` (dòng 354-415) - Route `/api/feedback-history`

- Route `/api/feedback-history` với `@login_required` (dòng 354-355)
- Parse query parameters (dòng 357-362): page, per_page, time_filter, start_date, end_date
- Filter theo user_id (chỉ lấy feedback của user hiện tại) (dòng 364)
- Áp dụng time filter (dòng 366-389):
  - `today`: Lấy feedback từ 00:00:00 hôm nay (dòng 369-372)
  - `week`: Lấy feedback từ 7 ngày trước (dòng 373-376)
  - `month`: Lấy feedback từ 30 ngày trước (dòng 377-380)
  - `custom`: Lấy feedback trong khoảng start_date → end_date (dòng 381-389)
  - Chuyển đổi timezone từ UTC sang Vietnam time để filter chính xác
- Paginate results (mặc định 10 items per page) (dòng 391-392)
- Convert feedback objects thành JSON format (dòng 394-404)
- Chuyển đổi created_at từ UTC sang Vietnam time để hiển thị (dòng 403)
- Trả về JSON (dòng 406-413):


```
{
  "feedbacks": [
    {
      "id": 1,
      "text": "Giảng viên rất nhiệt tình",
      "sentiment": "positive",
      "topic": "lecturer",
      "sentiment_confidence": 0.85,
      "topic_confidence": 0.9,
      "created_at": "14:30:00 25/12/2024"
    }
```

```
  ],
  "total": 100,
  "pages": 10,
  "current_page": 1,
  "has_next": true,
  "has_prev": false
}
```

Frontend hiển thị:

File: `static/js/app.js` (function `displayFeedbackHistory`)

1. Parse JSON response
2. Render danh sách feedbacks vào table
3. Hiển thị pagination buttons
4. Update total count
5. Scroll đến phần history nếu cần

Flow xem thống kê:

File: `app.py` (dòng 270-312) - Route `/my-statistics`

- Route `/my-statistics` với `@login_required` (dòng 270-271)
- Query database để lấy (dòng 273-302):
  - Tổng số feedbacks của user (dòng 274-275)
  - Phân bố sentiment (dòng 277-281)
  - Phân bố topic (dòng 283-287)
  - Số lượng feedback theo ngày - 30 ngày gần nhất (dòng 289-299)
  - 10 feedbacks gần nhất (dòng 301-302)
- Render template `my_statistics.html` với data (dòng 304-309)
- Hiển thị charts và statistics

Flow xem database (Admin):

File: `app.py` (dòng 314-352) - Route `/admin/database`

- Route `/admin/database` với `@admin_required` (dòng 314-315)
- Query database để lấy (dòng 317-341):
  - Tổng số users và feedbacks (dòng 318-319)
  - Phân bố sentiment và topic (toàn bộ hệ thống) (dòng 322-332)
  - Số lượng feedback theo ngày - 7 ngày gần nhất (dòng 334-341)
  - 10 feedbacks gần nhất (dòng 320)
- Render template `database_view.html` với data (dòng 343-349)

💾 FLOW 6: BACKUP VÀ RESTORE DATABASE

Backup tự động (Scheduled)

File: `app.py` (dòng 43-52)

- Scheduler chạy mỗi giờ → gọi `backup_database()`
- Chạy trong background thread (không block main thread)
- Khi app shutdown → tự động backup (dùng `atexit`)

Quá trình backup:

File: `database_manager.py` (dòng 155-192) - Method `backup_database()`

1. Kiểm tra có HF_TOKEN không → nếu không có thì return True (local mode)
2. Convert SQLite → JSON:
   - Duyệt tất cả tables trong database
   - Convert mỗi table thành JSON format (columns + data)
   - Lưu cả sqlite_sequence nếu có
3. Tạo temporary JSON file:
   `backups/feedback_backup_<timestamp>.json`

4. Upload lên Hugging Face Hub:
    - Repo: dataset repo (khác với model repo)
    - File name: `feedback_backup.json`
    - Nếu file không thay đổi → Hugging Face Hub báo "No files have been modified" → vẫn return True
5. Xóa temporary file
6. Return True/False

Restore database:

File: `database_manager.py` (dòng 194-217) - Method `restore_database()`

1. Kiểm tra có HF_TOKEN và API không
2. Download `feedback_backup.json` từ Hugging Face Hub
3. Parse JSON file
4. Convert JSON → SQLite:
    - Tạo database mới (xóa database cũ nếu có)
    - Tạo tables theo schema từ JSON
    - Insert data vào tables
    - Restore sqlite_sequence nếu có
5. Xóa temporary directory
6. Return True/False

Manual backup/restore (Admin):

File: `app.py` (dòng 449-469) - Routes `/admin/backup` và `/admin/restore`

- Route `/admin/backup` (POST, admin only): Gọi `backup_database()` và trả JSON

- Route `/admin/restore` (POST, admin only): Gọi `restore_database()` và trả JSON
- Frontend gọi API này khi admin click button backup/restore

🎓 FLOW 7: TRAIN MODEL VÀ EXPORT (NOTEBOOK)

Quá trình training:

File: `Main_FeedBack_Analysis.ipynb`

Bước 1: Load và chuẩn hóa data

- Load CSV files: `train_data.csv`, `train.csv`
- Chuẩn hóa text (Unicode normalization)
- Merge train và val thành một dataset
- Drop duplicates
- Tạo mapping: topic → topic_id, sentiment → sentiment_id

Bước 2: Build training pairs

- Function `build_pairs_df()` (Cell 11):
    - Với mỗi sentence trong dataset:
        - Tạo positive pair: (sentence, true_aspect, true_sentiment)
        - Tạo negative pairs: (sentence, other_aspects, label=0)
        - **Quan trọng**: Với `skip_hard_negatives=True`:
            - Không tạo negative pairs cho aspects có keywords trong sentence
            - Chỉ tạo negative pairs cho aspects không có keywords
            - Điều này giúp model học rằng nhiều aspects có thể cùng tồn tại
    - Return DataFrame với columns: text, aspect, label

Bước 3: Training

- Khởi tạo model: `PhoBERTPairABSA(base_model="vinai/phobert-base", num_cls=4, dropout=0.3)`
- Optimizer: AdamW với learning rate khác nhau cho backbone và classifier
- Scheduler: CosineAnnealingLR
- GradScaler: Mixed precision training (FP16)
- Class weights: Tính balanced weights, giảm weight cho class 0 (none) để giảm penalty
- Training loop:
  - Mỗi epoch: rebuild pairs (với random seed khác nhau)
  - Forward pass → tính loss (CrossEntropyLoss với label smoothing)
  - Backward pass → update weights
  - Track metrics: accuracy, F1 score
- Chỉ lưu final model (không lưu best model)

Bước 4: Export model

- Save model weights: `torch.save(model.state_dict(), "model.bin")`
- Save tokenizer: `tokenizer.save_pretrained("tokenizer")`

Bước 5: Upload lên Hugging Face Hub

- Login vào Hugging Face Hub với token
- Upload tokenizer directory
- Upload `model.bin` file
- Commit message: "Upload model weights (final model from training)"

Sau khi upload:

- Backend (`app.py`) sẽ tự động load model từ Hugging Face Hub khi khởi động
- Model sẵn sàng để inference

🔄 FLOW 8: MODEL INFERENCE CHI TIẾT

Input → Output flow:

```
User Input: "Giảng viên đi trễ, wifi kém"
    ↓
Preprocessing: Normalize text, check garbage
    ↓
For each aspect (4 aspects):
    ↓
    Get Prompt:
    - Check keywords: "đi trễ" → match "dung_gio" subtopic
    - Return prompt: "ĐÁNH GIÁ TÍNH ĐÚNG GIỜ/LÊN LỚP của GIẢN
    ↓
    Tokenize: (prompt, text) → input_ids, attention_mask
    ↓
    Model Forward:
    - PhoBERT encode → hidden states
    - Extract [CLS] token
    - Classifier → logits [4] (none, neg, neu, pos)
    ↓
    Softmax → probabilities [4]
    ↓
    Check keywords: aspect có keywords không?
    ↓
Stack all logits → tensor [4, 4]
    ↓
Calculate confidence: conf_not_none = 1.0 - p_none
    ↓
Keyword Boost: Nếu có keywords → +8% confidence
    ↓
Adaptive Thresholding:
- First pass: Keep aspects với confidence >= 0.50
- Nếu có >= 1 aspect → giảm threshold 12% cho aspects còn lại
- Chỉ giữ nếu có keywords HOẶC confidence đủ cao
    ↓
For each kept aspect:
    ↓
    Calculate sentiment:
    - Get sentiment probabilities (bỏ none)
    - Find top sentiment và margin
```

```
    - Adaptive margin: Nếu có keywords → giảm margin threshol
    - Filter: probability >= 0.40 và margin >= adjusted margi
    ↓
Sort by confidence (descending)
    ↓
Output: [
    {topic: "lecturer", sentiment: "negative", confidence: 0.
    {topic: "facility", sentiment: "negative", confidence: 0.
]
```

Các điểm quan trọng:

1. **Subprompt selection**: Model sử dụng prompt cụ thể dựa trên keywords trong text → tăng độ chính xác
2. **Keyword boost**: Aspects có keywords được boost confidence → giảm false negative
3. **Adaptive thresholding**: Nếu đã detect được aspect → giảm threshold cho aspects khác → hỗ trợ multi-topic detection
4. **Adaptive margin**: Aspects có keywords có thể có sentiments gần nhau → giảm margin threshold
5. **Multi-topic support**: Model có thể detect nhiều topics trong một câu

---

📁 MAPPING FILE VÀ CHỨC NĂNG

File	Chức năng chính	Vị trí trong flow
`app.py`	Main Flask application	Tất cả routes, authentication, inference logic
`model_config.py`	Model configuration	Prompts, keywords, thresholds, helper functions

File	Chức năng chính	Vị trí trong flow
`PhoBERTPairABSA.py`	Model architecture	Model definition, forward pass
`models.py`	Database models	User, Feedback models, password hashing
`forms.py`	Form validation	RegistrationForm, LoginForm
`database_manager.py`	Backup/restore	SQLite ↔ JSON conversion, Hugging Face Hub upload/download
`templates/*.html`	Frontend templates	HTML pages (login, register, index, statistics)
`static/js/app.js`	Frontend logic	AJAX requests, DOM manipulation, display results
`static/css/*.css`	Frontend styles	CSS styling
`Main_FeedBack_Analysis.ipynb`	Training notebook	Train model, export, upload to Hugging Face Hub

---

🎯 TÓM TẮT FLOW CHÍNH

1. User phân tích 1 feedback:

```
User nhập text → Frontend gửi POST /predict → Backend validat
analyze_feedback() → Model inference (4 aspects) → Filter & s
Save DB → Return JSON → Frontend display results
```

2. User phân tích CSV:

```
User upload CSV → Frontend gửi POST /analyze-csv → Backend pa
Loop: analyze_feedback() cho mỗi row → Save DB → Commit → Bac
Return JSON → Frontend display table
```

3. User xem lịch sử:

```
User chọn filter → Frontend gửi GET /api/feedback-history →
Backend query DB (filter + paginate) → Convert timezone →
Return JSON → Frontend display list + pagination
```

4. Backup database:

```
Scheduler (mỗi giờ) → backup_database() → SQLite → JSON →
Upload Hugging Face Hub → Cleanup
```

5. Train model:

```
Notebook load data → Build pairs → Train model → Export model
Upload Hugging Face Hub → Backend tự động load khi khởi động
```
