

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN CUỐI KỲ

Giáo viên hướng dẫn: TS. Bùi Tiến Lên

Môn học: Học máy

Hồ Chí Minh, tháng 07 năm 2025

MỤC LỤC

I. GIỚI THIỆU ĐỀ TÀI	3
1. Bối cảnh và động lực	3
2. Mục tiêu của đề tài.....	3
3. Phạm vi thực hiện	3
II. Giới thiệu mô hình và bộ dữ liệu	4
1. Kiến trúc mô hình được sử dụng	4
2. Lý do lựa chọn mô hình	4
3. Giới thiệu bộ dữ liệu.....	4
4. Đặc trưng và thống kê bộ dữ liệu.....	5
5. Tiền xử lí và chuẩn hóa dữ liệu.....	5
III. MÔ HÌNH GIẢI PHÁP ĐỀ XUẤT	5
1. Quy trình tổng quát	5
2. Thiết kế pipeline mô hình.....	6
3. Huấn luyện mô hình	6
IV. ĐÁNH GIÁ MÔ HÌNH.....	7
1. Kết quả trên tập huấn luyện & kiểm tra	7
2. Confusion Matrix	7
3. Nhận xét	7
V. ỨNG DỤNG VÀ TÍCH HỢP THỰC TẾ	7
1. Front-end	7
1.1. Mục tiêu	7
1.2. Công nghệ sử dụng	7
1.3. Cấu trúc thư mục	7
1.4. Luồng hoạt động	8
1.5. Phân tích chi tiết các file	8
1.6. Giao diện người dùng	9
1.7. Định hướng mở rộng.....	11
2. Back-end.....	11
2.1. Kiến trúc.....	11
2.2. Công nghệ sử dụng	11
2.3. Cấu trúc thư mục.....	11
2.4. Luồng hoạt động	12
3. Docker	12



3.1. Dockerfile	12
3.2. docker-compose.yml.....	12
3.3. Quy trình khởi chạy:	13
3.4. Lợi ích:	13
V. CÁC THÀNH VIÊN CỦA NHÓM.....	13
TÀI LIỆU THAM KHẢO.....	13

I. GIỚI THIỆU ĐỀ TÀI

1. Bối cảnh và động lực

- Trong bối cảnh nhu cầu phổ cập tri thức ngày càng cao, việc đảm bảo nội dung văn bản dễ hiểu, phù hợp với đối tượng người đọc là yếu tố then chốt. Các nền tảng giáo dục, báo chí và hệ thống tư vấn tự động (chatbot, trợ lý ảo) ngày càng cần những công cụ có khả năng đánh giá độ dễ hiểu của văn bản để điều chỉnh nội dung theo trình độ nhận thức của người dùng.
- Tuy nhiên, hiện nay các nghiên cứu và công cụ đánh giá độ dễ hiểu văn bản chủ yếu tập trung vào tiếng Anh; trong khi đó, tiếng Việt – với đặc điểm ngôn ngữ giàu hình thái và cấu trúc ngữ pháp linh hoạt lại chưa có nhiều nghiên cứu hệ thống và ứng dụng hiệu quả trong lĩnh vực này

2. Mục tiêu của đề tài

- Mục tiêu chính của đề tài là xây dựng một mô hình tự động đánh giá độ dễ hiểu của văn bản tiếng Việt, bao gồm các mục tiêu cụ thể như sau:
 - Tiền xử lý tập dữ liệu: Thu thập bộ dữ liệu văn bản tiếng Việt có gán nhãn mức độ dễ hiểu, phục vụ cho huấn luyện và đánh giá mô hình.
 - Xây dựng mô hình học máy/ học sâu: Tinh chỉnh các mô hình hiện đại (như Transformer, đặc biệt là PhoBERT) để học biểu diễn ngữ nghĩa của văn bản và dự đoán độ dễ hiểu.
 - Đánh giá hiệu quả mô hình: Sử dụng các chỉ số đánh giá phù hợp (Accuracy, F1-score, Confusion Matrix...) để đánh giá mô hình.

3. Phạm vi thực hiện

- Đề tài được thực hiện trong phạm vi sau:

- Ngôn ngữ: Văn bản tiếng Việt hiện đại.
- Thể loại văn bản: Chủ yếu là văn bản ngắn đến trung bình.
- Mức độ dễ hiểu: Phân loại thành 4 mức độ: Rất dễ, Dễ, Trung bình, và Khó hiểu – dựa trên tiêu chí như độ dài câu, tần suất từ chuyên ngành, cấu trúc ngữ pháp, và mức độ phổ biến của từ vựng.
- Dữ liệu và mô hình: Sử dụng mô hình pre-trained như PhoBERT để trích xuất đặc trưng, kết hợp với các thuật toán phân loại để huấn luyện mô hình đánh giá.

MỤC LỤC

II. GIỚI THIỆU MÔ HÌNH VÀ BỘ DỮ LIỆU

1. Kiến trúc mô hình được sử dụng

- Trong đề tài này, mô hình được sử dụng là PhoBERT, một biến thể của BERT được huấn luyện riêng trên tập dữ liệu tiếng Việt. PhoBERT sử dụng kiến trúc Transformer Encoder tương tự như BERT nhưng được huấn luyện lại trên tập dữ liệu tiếng Việt quy mô lớn (gồm 20GB văn bản từ Wikipedia, tin tức và các nguồn web tiếng Việt).
- PhoBERT cung cấp hai phiên bản: PhoBERT-base và PhoBERT-large, trong đó phiên bản base được sử dụng trong đề tài do phù hợp với tài nguyên tính toán hiện có.

Mô hình được khai thác theo hai hướng:

- Hướng 1 – Fine-tuning (tinh chỉnh toàn bộ mô hình): tinh chỉnh toàn bộ tham số của PhoBERT cùng với lớp phân loại trên tập dữ liệu đánh giá độ dễ hiểu.
- Hướng 2 – Feature Extraction (trích xuất đặc trưng): Sử dụng PhoBERT như một bộ mã hóa ngôn ngữ tĩnh, trích xuất embedding (biểu diễn vector) của văn bản từ lớp cuối cùng, sau đó đưa vào một mô hình phân loại bên ngoài (softmax) thay vì huấn luyện lại toàn bộ PhoBERT.

2. Lý do lựa chọn mô hình

- Tối ưu cho tiếng Việt: PhoBERT là mô hình pre-trained đầu tiên được huấn luyện chuyên biệt cho tiếng Việt, do đó có khả năng biểu diễn ngữ nghĩa và cú pháp tiếng Việt chính xác hơn so với BERT hoặc các mô hình đa ngôn ngữ khác.
- Hiệu quả cao trên nhiều tác vụ NLP tiếng Việt: PhoBERT đã chứng minh hiệu năng vượt trội trong các tác vụ phân loại, gán nhãn và trích xuất thông tin trên tiếng Việt
- Hỗ trợ linh hoạt trong thiết kế mô hình: PhoBERT cho phép áp dụng cả hai chiến lược huấn luyện phổ biến trong NLP hiện đại:
 - Fine-tuning: giúp mô hình học đặc trưng ngữ nghĩa chuyên biệt cho tác vụ đánh giá độ dễ hiểu, đồng thời tận dụng kiến thức ngôn ngữ đã học trước.
 - Feature extraction: giảm chi phí tính toán và tăng khả năng kết hợp với các mô hình truyền thống, đồng thời giúp đánh giá ảnh hưởng của từng thành phần trong pipeline.
- Khả năng mở rộng: Việc sử dụng PhoBERT làm nền tảng giúp đề tài có khả năng mở rộng sang các tác vụ phân tích văn bản tiếng Việt khác trong tương lai, như tóm tắt, tái viết văn bản dễ hiểu hơn.

3. Giới thiệu bộ dữ liệu

- Bộ dữ liệu được sử dụng trong đề tài là Vietnamese Text Readability Dataset do thầy An Vinh Luong phát hành trên GitHub. Bộ dữ liệu bao gồm các đoạn văn bản ngắn được gán nhãn mức độ dễ hiểu theo 4 mức: Very Easy, Easy, Medium và Difficult. Nguồn dữ liệu được tổng hợp từ nhiều lĩnh vực đa dạng, bao gồm: giáo dục, tin tức, giải trí, chính trị – xã hội,... với mục tiêu phản ánh phong phú các cấu trúc ngôn ngữ trong tiếng Việt hiện đại.

Mỗi dòng dữ liệu bao gồm hai thông tin chính:

- text: đoạn văn bản tiếng Việt.
- label: nhãn độ dễ hiểu tương ứng (0: Rất dễ, 1: Dễ, 2: Trung bình, 3: Khó).

4. Đặc trưng và thống kê bộ dữ liệu

- Phân bố nhãn:

- Rất dễ (Very Easy): 809 mẫu
- Dễ (Easy): 453 mẫu
- Trung bình (Medium): 242 mẫu
- Khó (Difficult): 321 mẫu
- Tổng cộng: 1825 mẫu

- Độ dài văn bản:

- Ngắn nhất: 53 từ
- Dài nhất: 34709 từ

5. Tiền xử lý và chuẩn hóa dữ liệu

- Để đảm bảo chất lượng dữ liệu đầu vào cho mô hình, quy trình tiền xử lý và chuẩn hóa văn bản được thực hiện theo các bước sau:

- Chuyển văn bản về chữ thường: Tất cả ký tự được chuyển thành chữ thường để giảm độ phức tạp mô hình và tránh trùng lặp không cần thiết trong từ vựng.
- Chuẩn hóa mã Unicode: Sử dụng thư viện unicodedata để chuyển văn bản về chuẩn Unicode dạng NFC, giúp thống nhất cách biểu diễn ký tự tiếng Việt có dấu, tránh lỗi tokenizer hoặc sai biệt khi mã hóa.
- Loại bỏ ký tự không hợp lệ: Văn bản chỉ giữ lại các chữ cái tiếng Việt (bao gồm đầy đủ dấu), chữ cái Latinh, khoảng trắng, và một số dấu câu cơ bản như ., , !, ?. Các ký tự không mang ý nghĩa ngôn ngữ như ký hiệu đặc biệt, emoji, HTML tag,... sẽ bị loại bỏ.
- Chuẩn hóa khoảng trắng: Loại bỏ các khoảng trắng thừa, tab hoặc xuống dòng bằng cách thay thế toàn bộ chuỗi ký tự trắng liên tiếp thành một dấu cách duy nhất.
- Tách từ bằng VnCoreNLP: Do tiếng Việt không phân tách từ bằng khoảng trắng rõ ràng, bước tách từ được thực hiện bằng thư viện VnCoreNLP với bộ phân tích từ (wseg) giúp xác định đúng ranh giới giữa các từ tiếng Việt.
- Không loại bỏ từ hiếm hoặc stopwords: Vì trong bài toán đánh giá độ dễ hiểu, sự xuất hiện của từ hiếm, từ chuyên ngành hoặc cấu trúc ngữ pháp khó chính là tín hiệu cần thiết để mô hình nhận diện mức độ khó của văn bản.

Sau quá trình xử lý, văn bản được đưa vào tokenizer của PhoBERT để trích xuất đặc trưng ngôn ngữ phục vụ huấn luyện mô hình.

III. MÔ HÌNH GIẢI PHÁP ĐỂ XUẤT

1. Quy trình tổng quát

- Tiền xử lý và chuẩn hóa dữ liệu: Văn bản được làm sạch, chuẩn hóa Unicode, loại bỏ ký tự không cần thiết và tách từ bằng VnCoreNLP.
- Tách tập dữ liệu: Chia thành tập huấn luyện và kiểm tra theo tỷ lệ 80:20.
- Mã hóa văn bản bằng PhoBERT: Văn bản sau khi xử lý được tokenizer chuyển đổi thành tensor đầu vào cho mô hình.
- Xây dựng mô hình phân loại: Dựa trên kiến trúc PhoBERT và các lớp phân loại phía sau để dự đoán mức độ dễ hiểu.
- Huấn luyện mô hình: Tinh chỉnh toàn bộ mô hình với loss function và optimizer phù

hợp.

- Đánh giá mô hình: Sử dụng độ chính xác (accuracy), ma trận nhầm lẫn (confusion matrix) và F1-score để đánh giá hiệu năng.

2. Thiết kế pipeline mô hình

- Pipeline mô hình bao gồm các thành phần sau:

- **Tokenizer:** Sử dụng AutoTokenizer từ thư viện HuggingFace, với model vinai/phobert-base-v2, để chuyển văn bản thành chuỗi token có độ dài tối đa 256 token. Thêm token đặc biệt [CLS] và [SEP], padding và truncation được áp dụng để chuẩn hóa độ dài đầu vào.
- **Model Backbone - PhoBERT:**
 - Dựa trên kiến trúc BERT, PhoBERT là mô hình pretrained học biểu diễn ngôn ngữ tiếng Việt.
 - Đầu ra sử dụng hidden state của token [CLS] (vị trí đầu tiên) đại diện cho toàn bộ câu.
- **Classifier Head:**
 - Gồm 3 tầng fully-connected:
 - Linear($768 \rightarrow 256$) \rightarrow ReLU \rightarrow Dropout(0.3)
 - Linear($256 \rightarrow 128$) \rightarrow ReLU \rightarrow Dropout(0.3)
 - Linear($128 \rightarrow 4$) \rightarrow đầu ra là vector logits tương ứng với 4 lớp (Very Easy, Easy, Medium, Difficult).
 - Kết thúc bằng softmax để phân phối xác suất cho từng lớp.
- **Kỹ thuật huấn luyện:**
 - Tối ưu bằng AdamW, learning rate 2e-5.
 - Lịch trình learning rate dạng tuyến tính (linear scheduler).
 - Tăng tốc bằng Accelerator của thư viện accelerate.

3. Huấn luyện mô hình

- Cấu hình huấn luyện:

- Epochs: 3
- Batch size: 16
- Loss function: CrossEntropyLoss (phù hợp với bài toán phân loại đa lớp)
- Optimizer: AdamW
- Device: Tự động sử dụng GPU (nếu có) qua Accelerator

- Quy trình huấn luyện:

- Trong mỗi epoch, mô hình được huấn luyện trên train_dataloader bằng hàm train_model() với gradient descent.
- Sau mỗi epoch, mô hình được đánh giá trên tập validation (val_dataloader) thông qua hàm evaluate_model(), trả về độ chính xác và ma trận nhầm lẫn.
- F1-score được tính để theo dõi hiệu quả phân loại từng lớp.

- Đánh giá:

- Ma trận nhầm lẫn được trực quan hóa bằng heatmap để quan sát chi tiết khả năng phân loại từng mức độ dễ hiểu.
- Accuracy và F1-score giúp đo lường độ chính xác tổng thể và cân bằng giữa precision–recall ở mỗi lớp.

IV. ĐÁNH GIÁ MÔ HÌNH

1. Kết quả trên tập huấn luyện & kiểm tra

- Accuracy:
 - Train tăng đều từ 68.8% → 89.8% qua 5 epoch.
 - Validation duy trì ~73–81%, ổn định.
- Loss:
 - Train on Loss dao động ~11–13 → dấu hiệu không overfit nghiêm trọng.
- F1-score:
 - Train F1 tăng từ 0.55 → 0.84.
 - Valid F1 tăng từ 0.63 → 0.72–0.75 → mô hình giữ được cân bằng precision–recall giữa các lớp.

2. Confusion Matrix

- Các lớp Very Easy (149) và Easy (76) được phân loại khá tốt.
- Một số văn bản mức Medium dễ nhầm với Easy (33 mẫu nhầm sang Easy).
- Lớp Difficult nhận diện ổn (57/64) → ít nhầm.

→ Khó nhất là tách Medium/Easy, do ranh giới từ vựng, cấu trúc câu gần nhau.

3. Nhận xét

- Mô hình PhoBERT fine-tuning hoạt động tốt trên tiếng Việt.
- Độ chính xác và F1-score đều cải thiện qua các epoch.
- Lỗi chủ yếu do:
 - Dữ liệu chưa cân bằng (Medium ít hơn).
 - Một số câu trung tính khó gán mức độ chính xác.
- Mô hình đủ ứng dụng thực tế nhưng có thể nâng cao bằng:
 - Bổ sung dữ liệu.
 - Tuning hyperparameter chi tiết hơn.
 - Cân bằng số lượng mẫu giữa các lớp

V. ỨNG DỤNG VÀ TÍCH HỢP THỰC TẾ

1. Front-end

1.1. Mục tiêu

- Xây dựng giao diện web giúp người dùng nhập văn bản, nhận kết quả phân tích, hiển thị rõ ràng, dễ hiểu.

1.2. Công nghệ sử dụng

- Ngôn ngữ: Python
- Framework: Streamlit
- Các thư viện khác: streamlit

1.3. Cấu trúc thư mục

- Ngôn ngữ: Python
- Framework: Streamlit
- Các thư viện khác: streamlit

-  frontend
 - |-  api.py
 - |-  app.py
 - |-  components.py
 - |-  config.py
 - |-  README.md

1.4. Luồng hoạt động

Frontend hoạt động theo quy trình như sau:

- Người dùng truy cập website, nhập văn bản tiếng Việt cần đánh giá.
- Frontend gửi nội dung văn bản (và session ID nếu có) lên backend thông qua API (api.py).
- Backend xử lý, trả về các chỉ số đánh giá độ dễ đọc.
- Kết quả được hiển thị trực tiếp trên trang: độ khó, thống kê từ/câu/âm tiết, lịch sử phân tích.
- Giao diện có hỗ trợ chức năng:
 - o Đăng ký / đăng nhập
 - o Lưu lịch sử người dùng
 - o Xác nhận đăng xuất

1.5. Phân tích chi tiết các file

- app.py – Giao diện chính
 - o Thiết lập cấu hình trang: tiêu đề, icon, bộ cục.
 - o Quản lý trạng thái người dùng bằng st.session_state.
 - o Tổ chức luồng giao diện: đăng nhập → phân tích → hiển thị kết quả → lịch sử.
 - o Giao tiếp với backend bằng hàm từ api.py.
- api.py – Giao tiếp với backend
 - o Gửi các request đến API:
 - analyze_text(text): gửi văn bản để phân tích.
 - login_user(username, password): đăng nhập.
 - register_user(username, full_name, password): đăng ký tài khoản.
 - get_history(session_id): lấy lịch sử phân tích.
 - o Xử lý lỗi đơn giản, trả về message rõ ràng.
- components.py – Dựng các thành phần giao diện
 - o Tách rời giao diện thành nhiều component tái sử dụng:
 - render_title() – Tiêu đề chính.
 - render_input_area() – Ô nhập văn bản.
 - render_submit_button() – Nút “Phân tích độ khó”.
 - render_result() – Hiển thị kết quả phân tích.
 - render_login() / render_register() – Form đăng nhập / đăng ký.

- render_topbar() – Hiển thị tên người dùng, logout, xem lịch sử.
- render_history() – Hiển thị danh sách truy vấn cũ.
- config.py – Cấu hình API
 - Biến API_URL: lưu địa chỉ backend, có thể cấu hình động qua biến môi trường.

1.6. Giao diện người dùng

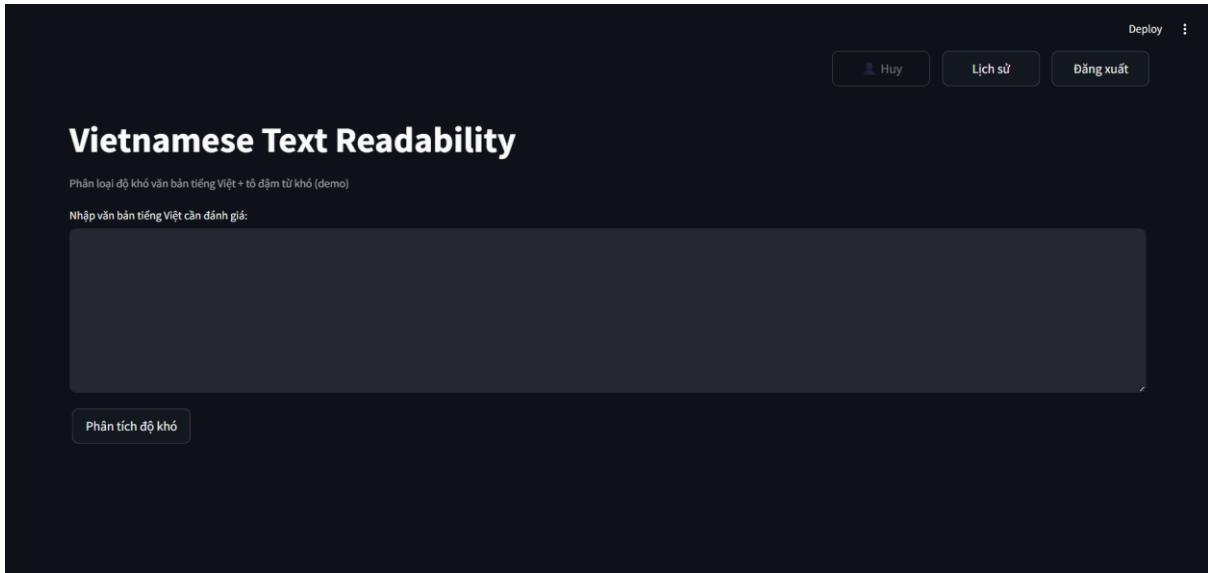
- Màn hình đăng nhập:

The screenshot shows a dark-themed login interface. At the top right are three buttons: 'Chưa đăng nhập' (Not logged in), 'Lịch sử' (History), and 'Đăng xuất' (Logout). Below these are two radio buttons for 'Tài khoản': 'Đăng nhập' (selected) and 'Đăng ký'. There are two input fields: 'Tên đăng nhập' (Login name) and 'Mật khẩu' (Password). A password visibility icon is located next to the password field. At the bottom is a large blue 'Đăng nhập' (Login) button.

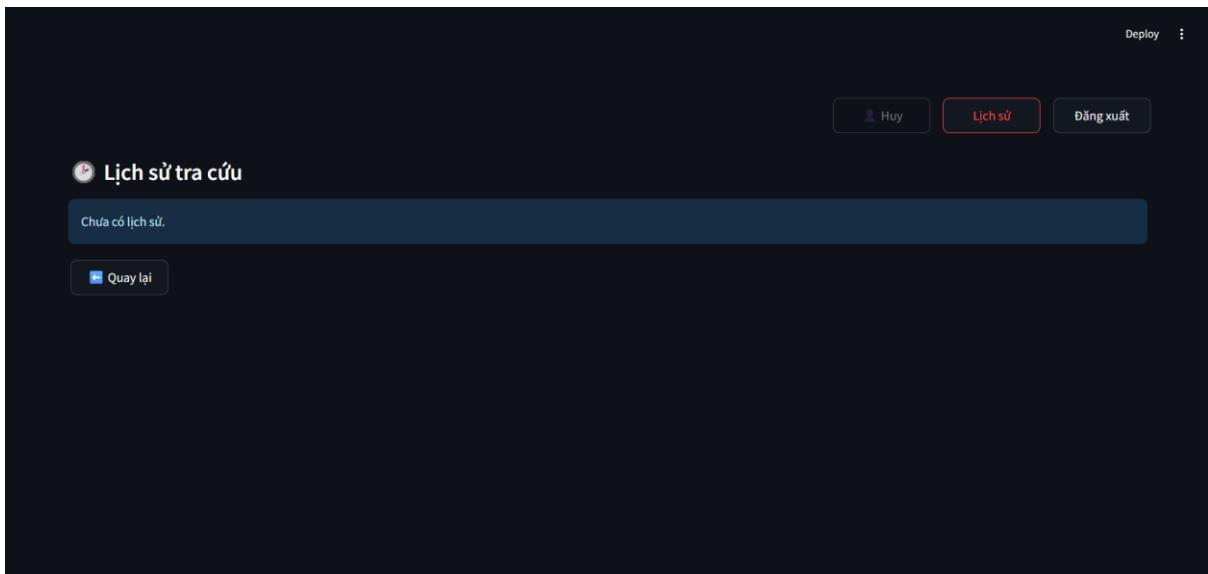
- Màn hình đăng ký:

The screenshot shows a dark-themed registration interface. At the top right are three buttons: 'Chưa đăng nhập' (Not logged in), 'Lịch sử' (History), and 'Đăng xuất' (Logout). Below these are two radio buttons for 'Tài khoản': 'Đăng nhập' and 'Đăng ký' (selected). There are four input fields: 'Tên đăng ký' (Registration name), 'Họ tên' (Name), and 'Mật khẩu đăng ký' (Registration password). A password visibility icon is located next to the password field. At the bottom is a large blue 'Đăng ký' (Register) button. Above the 'Đăng ký' button is a small checkbox labeled 'Đã đọc và đồng ý với các điều khoản' (I have read and agree to the terms and conditions).

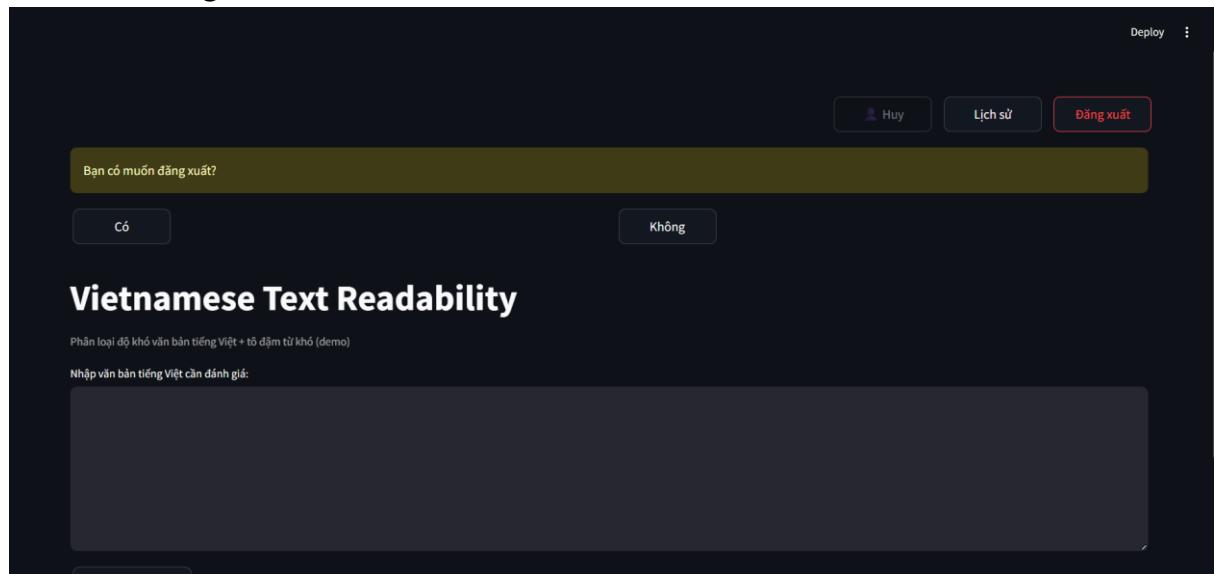
- Màn hình chính:



- Màn hình xem lịch sử:



- Màn hình đăng xuất:



1.7. Định hướng mở rộng

- Cho phép upload file .txt để phân tích.
- Hỗ trợ xuất kết quả ra PDF.
- Nâng cấp UI bằng streamlit.components.v1 để tùy biến sâu hơn.

2. Back-end

2.1. Kiến trúc

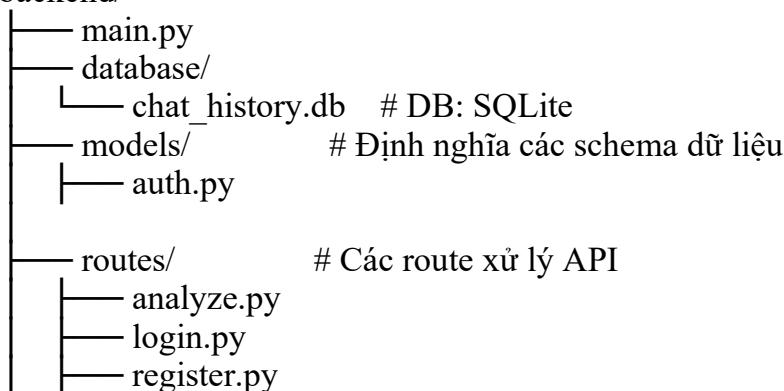
- Phần mềm sử dụng kiến trúc Client – Server.
- Client (giao diện người dùng) gửi yêu cầu thông qua HTTP request
- Server (backend) chịu trách nhiệm xử lý các yêu cầu, tương tác với cơ sở dữ liệu và mô hình AI, sau đó trả kết quả lại cho client dưới dạng json

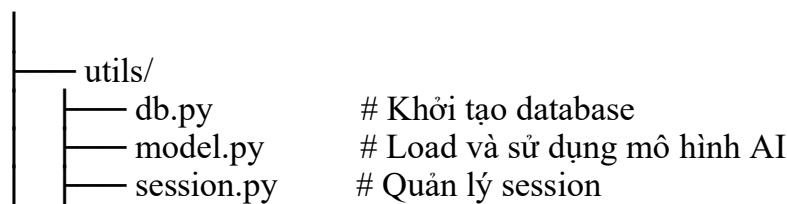
2.2. Công nghệ sử dụng

- Ngôn ngữ: python
- Framework: FastAPI
- Database: SQLite3

2.3. Cấu trúc thư mục

backend/





2.4. Luồng hoạt động

- Khi chạy tệp main.py chương trình thực hiện các bước sau:
 - o Khởi tạo cơ sở dữ liệu nếu chưa tồn tại
 - o Nạp các router từ module để cấu hình các API endpoint
 - o Sau khi hoàn tất, dịch vụ server sẽ được khởi động tại địa chỉ:
<http://localhost:8000>.
- Server đợi các request từ client.

3. Docker

3.1. Dockerfile

Mục Đích:

Dockerfile dùng để xây dựng một **image tích hợp cả mã nguồn, thư viện, và mô hình ML** để chạy backend (FastAPI) và frontend (Streamlit) trong môi trường độc lập, đảm bảo ứng dụng chạy ổn định trên mọi máy.

Nội dung chính:

- o Dựa trên image python:3.10-slim để tối ưu kích thước
- o Cài đặt tất cả thư viện cần thiết từ requirements.txt
- o Tải mô hình readability_model.pt từ Google Drive nếu chưa có bằng script download_model.py
- o Phân chia thành 2 stage:
 - o backend: chạy FastAPI (cổng 8000)
 - o frontend: chạy Streamlit (cổng 8501)

3.2. docker-compose.yml

Mục đích:

- o docker-compose.yml giúp **dựng nhiều container cùng lúc** (FE & BE), quản lý mạng, port, và volume dễ dàng, thuận tiện cho nhóm phát triển.

Cấu hình:

- o 2 services:
 - o backend: chạy FastAPI, expose cổng 8000
 - o frontend: chạy Streamlit, expose cổng 8501
- o Volumes:
 - o Mount file readability_model.pt từ máy host vào container

- Mount database chat_history.db để giữ lại dữ liệu người dùng
- Mạng nội bộ:
 - Hai service giao tiếp qua hostname backend trong Docker network

3.3. Quy trình khởi chạy:

- Build và chạy toàn bộ hệ thống
 - docker-compose up --build
- Mở trình duyệt:
 - API: http://localhost:8000/docs
 - Giao diện người dùng: http://localhost:8501

3.4. Lợi ích:

Mục tiêu	Đáp ứng
Đóng gói ứng dụng toàn diện	Có cả FE, BE, model, DB
Dễ deploy trên bất kỳ máy nào	Nhờ Docker
Tự động tải model khi chưa có	Dùng gdown
Dữ liệu người dùng không bị mất	Mount chat_history.db
Giao tiếp giữa FE ↔ BE ổn định	Dùng hostname nội bộ backend

V. CÁC THÀNH VIÊN CỦA NHÓM

Họ tên	MSSV	Vai trò	Nhiệm vụ
Trần Mạnh Hùng	22120117	Trưởng nhóm	Train model, lên kế hoạch, phân công
Mai Nhựt Huy	22120136	Thành viên	Frontend
Võ Nguyễn Song Huy	22120141	Thành viên	Backend, CSDL
Vy Quốc Huy	22120142	Thành viên	Tạo dockerfile, kết nối model
Nguyễn Hùng Việt	22120431	Thành viên	Chỉnh sửa, đảm bảo kết nối BE, FE

TÀI LIỆU THAM KHẢO

- ChatGPT
- Components • Streamlit
- <https://fastapi.tiangolo.com/features/#starlette-features>