ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KĨ THUẬT MÁY TÍNH

BÁO CÁO BÀI TẬP LỚN

XỬ LÍ ẢNH SỐ VÀ THỊ GIÁC MÁY TÍNH

TÊN ĐỀ TÀI
HỆ THỐNG CHECK-IN TỰ ĐỘNG BẰNG NHẬN DIỆN KHUÔN MẶT

GVHD: Nguyễn Đức Dũng

SVTH1: Nguyễn Quốc Việt - 2115279
SVTH2: Cao Đức Vinh - 2115290

TP. HỒ CHÍ MINH, 1/2025

Mục lục

Danh sách hình vẽ

CHƯƠNG 1

Mở đầu

1.1 Đặt vấn đề

Trong bối cảnh hiện đại, việc đảm bảo an ninh và quản lý truy cập vào các khu vực quan trọng như hội trường, văn phòng, và các sự kiện lớn đang trở thành một thách thức lớn. Các phương pháp truyền thống như sử dụng thẻ từ, mật khẩu, hoặc danh sách điểm danh thủ công ngày càng bộc lộ nhiều hạn chế, từ việc dễ bị giả mạo, thất lạc, đến tốn thời gian và công sức quản lý. Những thách thức này không chỉ ảnh hưởng đến hiệu quả vận hành mà còn tạo ra những lỗ hổng nghiêm trọng về bảo mật.
Trước tình hình đó, công nghệ nhận diện khuôn mặt nổi lên như một giải pháp tiên tiến, giúp tự động hóa quy trình kiểm soát truy cập một cách chính xác và hiệu quả. Với khả năng nhận diện thời gian thực, công nghệ này không chỉ đảm bảo an ninh cao mà còn mang lại sự tiện lợi tối đa cho người dùng. Thay vì phải mang theo thẻ hoặc nhập mật khẩu, người dùng chỉ cần xuất hiện trước camera để hoàn tất quá trình xác thực.
Ứng dụng công nghệ nhận diện khuôn mặt trong hệ thống check-in tự động mang lại nhiều lợi ích vượt trội:

- Tăng cường bảo mật: Giảm thiểu rủi ro từ việc mất thẻ hoặc rò rỉ mật khẩu.
- Tiết kiệm thời gian: Loại bỏ các quy trình xác thực phức tạp, rút ngắn thời gian chờ đợi.
- Cá nhân hóa và hiện đại hóa: Tạo ra trải nghiệm người dùng chuyên nghiệp, phù hợp với xu hướng công nghệ 4.0.

Hệ thống này không chỉ phù hợp với các hội trường và văn phòng mà còn có tiềm năng lớn trong quản lý sự kiện, trường học, bệnh viện, và nhiều lĩnh vực khác. Bài toán đặt ra là làm thế nào để xây dựng một hệ thống check-in bằng nhận diện khuôn mặt đáp ứng các yêu cầu về tính chính xác, tốc độ xử lý, và khả năng triển khai thực tế.

1.2 Mục tiêu nghiên cứu

Mục tiêu của nghiên cứu này là xây dựng hệ thống check-in tự động bằng nhận diện khuôn mặt, với các yêu cầu:

- Đảm bảo an ninh và chính xác trong nhận diện.
- Tối ưu hóa thời gian xử lý, hoạt động trong thời gian thực.
- Dễ dàng triển khai trong môi trường cục bộ.

1.3 Phương pháp nghiên cứu

Từ đó nhóm đã đưa ra hướng đi cho bài tập lớn này như sau:

1. Tìm hiểu lý thuyết và phân tích các thuật toán nhận diện khuôn mặt.
2. Thiết kế mô hình nhận diện khuôn mặt dựa trên KNN.
3. Thực nghiệm trên dữ liệu thực tế để kiểm tra hiệu năng.

CHƯƠNG 2

Tìm hiểu tổng quan

2.1 Nguyên lí nhận diện khuôn mặt

Nhận diện khuôn mặt là một công nghệ máy học sử dụng các đặc điểm của khuôn mặt để xác định danh tính của một người. Các đặc điểm này có thể bao gồm hình dạng, kích thước, vị trí của các đặc điểm trên khuôn mặt, và các đặc điểm khác. Có nhiều cách khác nhau để thực hiện nhận diện khuôn mặt. Một cách phổ biến là sử dụng các đặc điểm đặc trưng của khuôn mặt. Các đặc điểm này có thể được xác định bằng cách sử dụng các kỹ thuật như phân tích hình học hoặc phân tích đặc trưng. Một cách khác để thực hiện nhận diện khuôn mặt là sử dụng các mô hình học máy. Các mô hình này được đào tạo trên một tập dữ liệu các hình ảnh khuôn mặt đã được dán nhãn. Các mô hình này có thể học cách phân biệt giữa các khuôn mặt khác nhau

2.2 Machine Learning

2.2.1 Định nghĩa

Là một lĩnh vực của trí tuệ nhân tạo liên quan đến việc nghiên cứu và xây dựng các kĩ thuật cho phép các hệ thống học tự động từ dữ liệu để giải quyết các vấn đề cụ thể. Ví dụ, các máy có thể học cách phân loại thư điện tử có phải thư rác hay không và tự động sắp xếp vào các thư mục tương ứng.
Machine Learning có liên quan đến thống kê, nhưng khác với thống kê, học máy tập trung vào sự phức tạp của các giải thuật trong việc thực thi tính toán.
Machine Learning được áp dụng rộng rãi trong máy truy tìm dữ liệu, máy phân tích thị trường chứng khoán, nhận dạng tiếng nói và chữ viết, và nhiều lĩnh vực khác.

2.2.2 Một số phương thức của Machine Learning

1. **Supervised learning (Học có giám sát)**

   Thuật toán dự đoán đầu ra của một dữ liệu mới (new input) dựa trên các cặp (input,

outcome) đã biết từ trước. Cặp dữ liệu này còn được gọi là (data, label), tức (dữ liệu, nhãn). Supervised learning là nhóm phổ biến nhất trong các thuật toán Machine Learning. Học có giám sát được phân thành 2 loại chính:

- **Classification (Phân lớp):** Quá trình phân loại một đối tượng dữ liệu vào một hay nhiều lớp đã được định trước thông qua một mô hình phân lớp(model). Mô hình này được xây dựng dựa trên một tập dữ liệu được xây dựng trước đó có gán nhãn (hay còn gọi là tập huấn luyện). Quá trình phân lớp là quá trình gán nhãn cho đối tượng dữ liệu.

  Có nhiều bài toán phân lớp như phân lớp nhị phân, phân lớp đa lớp, phân lớp đa trị. Trong đó phân lớp nhị phân là một loại phân lớp đặc biệt của phân lớp đa lớp.

  Ứng dụng của bài toán phân lớp được sử dụng rất nhiều và rộng rãi như nhận dạng khuôn mặt, nhận dạng chữ viết, nhận dạng giọng nói, phát hiện thư rác
- **Regression (Hồi quy):** Nếu không được chia thành các nhóm mà là một giá trị thực cụ thể. Đầu ra của một điểm dữ liệu sẽ bằng chính đầu ra của điểm dữ liệu đã biết.

2. **Unsupervised learning (Học không giám sát)**

   Là một kĩ thuật của máy học nhằm tìm ra một mô hình hay cấu trúc bị ẩn bởi tập dữ liệu không được gán nhãn cho trước. Unsupervised learning khác với Supervised learning là không thể xác định trước output từ tập dữ liệu huấn luyện được. Tùy thuộc vào tập huấn luyện kết quả output sẽ khác nhau. Trái ngược với Supervised learning, tập dữ liệu huấn luyện của Unsupervised learning không do con người gán nhãn, máy tính sẽ phải tự học hoàn toàn. Có thể nói, học không giám sát thì giá trị đầu ra sẽ phụ thuộc vào thuật toán Unsupervised learning. Ứng dụng lớn phổ biến của học không giám sát là bài toán phân cụm

3. **Semi-supervised learning (Học bán giám sát)**

   Các bài toán khi có một số lượng lớn dữ liệu nhưng chỉ một phần trong chúng được dán nhãn. Những bài toán này nằm giữa phương thức học giám sát và học không giám sát.

   Ngoài ra, còn một số cách tiếp cận khác về các phương thức của machine learning

2.3 Thuật toán K-Nearest Neighbors (KNN)

Thuật toán K-Nearest Neighbors (KNN) là một phương pháp máy học mạnh mẽ và trực giác được sử dụng để giải quyết các vấn đề phân loại và hồi quy. Bằng cách tận dụng

khái niệm về sự tương đồng, KNN dự đoán nhãn hoặc giá trị của một điểm dữ liệu mới bằng cách xem xét K láng giềng gần nhất của nó trong tập dữ liệu huấn luyện

2.3.1 Định nghĩa

Thuật toán K-Nearest Neighbors (KNN) là một trong những thuật toán phân loại cơ bản nhưng quan trọng nhất trong Machine Learning. Nó thuộc lĩnh vực học có giám sát và được ứng dụng mạnh mẽ trong việc nhận diện mẫu, khai thác dữ liệu và phát hiện xâm nhập. Ý tưởng của KNN là tìm ra output của dữ liệu dựa trên thông tin của những dữ liệu huấn luyện gần nó nhất.
KNN được rộng rãi sử dụng trong các tình huống thực tế vì nó là thuật toán phi tham số, có nghĩa là nó không đặt ra bất kỳ giả định cơ bản nào về phân phối của dữ liệu (ngược lại với các thuật toán khác như GMM, mà giả định về phân phối Gaussian của dữ liệu). Chúng ta được cung cấp một lượng dữ liệu trước đó (còn được gọi là dữ liệu huấn luyện), trong đó các điểm được phân loại vào các nhóm được xác định bởi một thuộc tính.
KNN không chỉ giữ vai trò quan trọng trong việc phân loại mà còn trong việc dự đoán giá trị của các điểm dữ liệu mới. Nó là một công cụ mạnh mẽ cho việc tìm kiếm sự tương đồng giữa các điểm dữ liệu và dự đoán dựa trên những điểm gần nhất. Điều này làm cho KNN trở thành một phương pháp linh hoạt và không đòi hỏi các giả định mạnh mẽ, làm cho nó phù hợp với nhiều ứng dụng thực tế khác nhau.

2.3.2 Các độ đo khoảng cách sử dụng trong thuật toán KNN

Như chúng ta biết, thuật toán KNN giúp chúng ta xác định các điểm gần nhất hoặc nhóm gần nhất cho một điểm truy vấn. Tuy nhiên, để xác định các nhóm gần nhất hoặc các điểm gần nhất cho một điểm truy vấn, chúng ta cần một số độ đo. Để đáp ứng mục đích này, chúng ta sử dụng các độ đo khoảng cách sau:

1. **Khoảng cách Euclidean**
   Khoảng Cách Euclidean là khoảng cách giữa hai điểm trong mặt phẳng/đa mặt. Nó cũng có thể được hình dung như độ dài của đường thẳng nối hai điểm đang xét.
   Đây là một độ đo giúp chúng ta tính toán sự di chuyển net giữa hai trạng thái của một đối tượng. Khoảng Cách Euclidean giữa một điểm truy vấn x và một điểm Xi trong không gian d chiều được xác định bởi công thức:

$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d} (x_j - X_{ij})^2}$$

   - x: Biểu diễn điểm truy vấn.

- X_i: Biểu diễn một điểm trong tập dữ liệu.
- d: Số chiều của không gian.
- x_j: Thành phần thứ j của điểm truy vấn x.
- X_{ij}: Thành phần thứ j của điểm X_i trong tập dữ liệu.

2. **Khoảng cách Manhattan**

Khoảng cách Manhattan thường được sử dụng khi quan tâm đến tổng quãng đường đã đi của đối tượng thay vì khoảng cách điểm đến điểm. Độ đo này được tính bằng cách cộng tổng giá trị tuyệt đối của sự chênh lệch giữa các tọa độ của các điểm trong không gian n chiều.

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

- x và y: Là các điểm cần tính khoảng cách.
- n: Là số chiều của không gian.
- x_i và y_i: Là các thành phần thứ i của điểm x và y, tương ứng.

3. **Khoảng cách Minkowski**

Khoảng cách Minkowski giữa hai điểm x và y trong không gian n chiều được xác định bởi:

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Các trường hợp đặc biệt:

- Khi $p = 2$: Công thức trở thành khoảng cách Euclidean.
- Khi $p = 1$: Công thức trở thành khoảng cách Manhattan.

Các độ đo khác: Ngoài các độ đo đã đề cập, còn có các độ đo khoảng cách khác như:

- **Khoảng cách Hamming:** Thường được sử dụng để so sánh trùng lặp giữa hai vectơ chứa giá trị boolean hoặc chuỗi.

4. **Cách Chọn Giá Trị của k cho Thuật Toán KNN**

Giá trị của k đóng vai trò rất quan trọng trong thuật toán KNN để xác định số lượng láng giềng trong thuật toán. Giá trị của k trong thuật toán k-nearest neighbors (k-NN) nên được chọn dựa trên dữ liệu đầu vào. Nếu dữ liệu đầu vào có nhiều ngoại lệ

hoặc nhiễu, giá trị của k lớn hơn sẽ tốt hơn. Đề xuất chọn giá trị lẻ cho k để tránh các trường hợp xung đột trong phân loại. Phương pháp kiểm tra chéo (cross-validation) có thể giúp chọn ra giá trị tốt nhất cho k dựa trên tập dữ liệu cụ thể.

5. **Hoạt động của Thuật toán K-Nearest Neighbors (KNN)**

   Thuật toán K-Nearest Neighbors (KNN) hoạt động dựa trên nguyên lý tương đồng, nơi nó dự đoán nhãn hoặc giá trị của một điểm dữ liệu mới bằng cách xem xét nhãn hoặc giá trị của K láng giềng gần nhất trong tập dữ liệu huấn luyện.

   Để thực hiện dự đoán, thuật toán tính toán khoảng cách giữa mỗi điểm dữ liệu mới trong tập dữ liệu kiểm thử và tất cả các điểm dữ liệu trong tập dữ liệu huấn luyện. Khoảng cách Euclidean là một độ đo khoảng cách phổ biến trong KNN, nhưng các độ đo khoảng cách khác như khoảng cách Manhattan hoặc khoảng cách Minkowski cũng có thể được sử dụng tùy thuộc vào vấn đề và dữ liệu.

   Khi khoảng cách giữa điểm dữ liệu mới và tất cả các điểm dữ liệu trong tập huấn luyện đã được tính, thuật toán tiếp tục tìm K láng giềng gần nhất dựa trên những khoảng cách này. Phương pháp cụ thể để chọn láng giềng gần nhất có thể thay đổi, nhưng một cách tiếp cận phổ biến là sắp xếp khoảng cách theo thứ tự.

6. **Ứng dụng của Thuật toán K-Nearest Neighbors (KNN)**

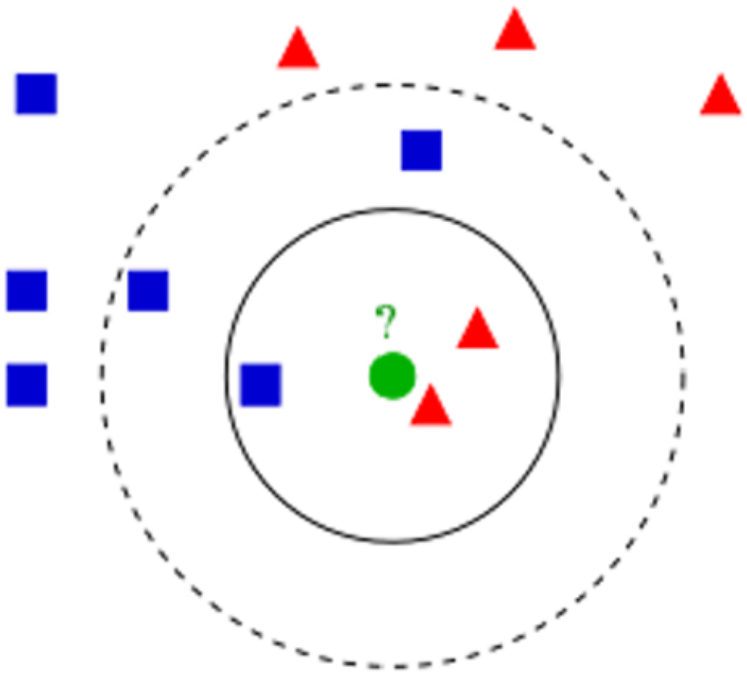
   Một số ứng dụng phổ biến của thuật KNN có thể kể đến như:

   - **Tiền Xử Lý Dữ Liệu (Data Preprocessing):** Khi xử lý bất kỳ vấn đề Máy Học nào, chúng ta thường thực hiện phần EDA (Exploratory Data Analysis). Nếu phát hiện rằng dữ liệu chứa giá trị thiếu, có nhiều phương pháp đền trái có sẵn. Một trong những phương pháp đó là KNN Imputer, một phương pháp hiệu quả và thường được sử dụng cho các phương pháp đền trái phức tạp.
   - **Nhận Diện Mẫu (Pattern Recognition):** Thuật toán KNN hoạt động rất tốt khi bạn đã huấn luyện một mô hình KNN bằng bộ dữ liệu MNIST và sau đó thực hiện quá trình đánh giá. Bạn có thể nhận thấy rằng độ chính xác là rất cao.
   - **Hệ Thống Gợi Ý (Recommendation Engines):** Nhiệm vụ chính của thuật toán KNN là gán một điểm truy vấn mới vào một nhóm đã tồn tại được tạo ra bằng một tập dữ liệu lớn. Điều này chính xác là những gì cần thiết trong các hệ thống gợi ý để gán mỗi người dùng vào một nhóm cụ thể và sau đó cung cấp gợi ý dựa trên sở thích của nhóm đó.

7. **Quy trình làm việc của KNN**

- Xác định tham số K, số láng giềng gần nhất.
- Tính khoảng cách đối tượng cần phân lớp với tất cả các đối tượng trong tập dữ liệu huấn luyện.
- Sắp xếp khoảng cách theo thứ tự tăng dần và xác định K láng giềng gần nhất.
- Lấy tất cả các lớp của K láng giềng gần nhất.
- Dựa vào phần lớn lớp của K để xác định lớp cho đối tượng cần phân lớp.

8. **Ví dụ về phân loại theo thuật toán KNN**

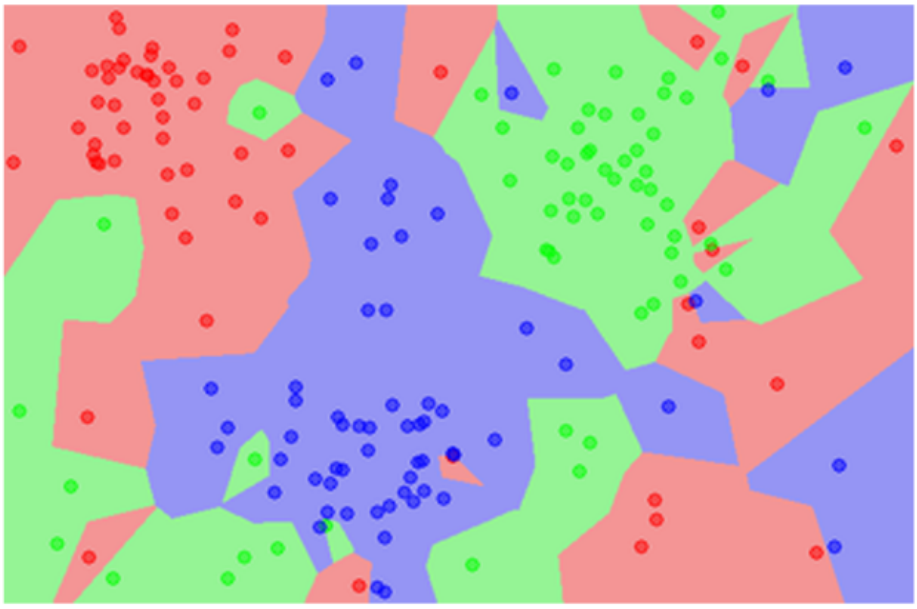


Hình 2.1: Ví dụ minh họa thuật toán KNN

Mẫu kiểm tra (điểm tròn màu xanh) sẽ được phân loại vào các hình vuông màu xanh hoặc các tam giác màu đỏ. Nếu K = 3 (vòng tròn đường kẻ đậm), nó sẽ được phân loại vào các tam giác màu đỏ vì có 2 tam giác và chỉ có 1 hình vuông nằm trong vòng tròn trong cùng. Nếu K = 5 (vòng tròn đường kẻ đứt), nó sẽ được phân loại vào các hình vuông màu xanh (3 hình vuông so với 2 tam giác nằm trong vòng tròn ngoài cùng). Có một vấn đề trong phương pháp trên khi trường hợp lấy K=4, ta nhận thấy sẽ có hai hình vuông xanh và hai hình tam giác đỏ gần đó nhất, đây là trường hợp có điểm bằng nhau, với trường hợp này KNN sẽ xử lý bằng cách so sánh tổng khoảng cách của các hình gần nhất với điểm ta đang xét.

Do xuất hiện trường hợp có điểm bằng nhau, vì vậy người ta thường chọn k là số lẻ. Đó cũng là ý tưởng của KNN.

9. **Ví dụ về KNN gây nhiễu với K =1**



Hình 2.2: Bản đồ minh họa KNN nhiễu với K = 1

Hình trên là bài toán phân lớp với ba lớp: đỏ, lam, lục. Mỗi điểm dữ liệu mới sẽ được gán nhãn theo màu của điểm đó mà nó thuộc về. Trong hình này, chú ý vùng khoanh tròn màu vàng, ta nhận thấy rằng điểm màu lục nằm giữa hai vùng lớn với nhiều dữ liệu đỏ và lam, điểm này rất có thể là nhiễu dẫn đến việc dữ liệu test nếu rơi vào vùng này sẽ có nhiều khả năng cho kết quả sai lệch.

10. **Ưu điểm, nhược điểm của thuật toán**

- **Ưu điểm:**
  - Dễ sử dụng, dễ cài đặt.
  - Việc dự đoán kết quả của dữ liệu mới dễ dàng.
  - Độ phức tạp tính toán nhỏ
- **Nhược điểm:**
  - Nhạy cảm với nhiễu và outliers: KNN có thể nhạy cảm với nhiễu và outliers, đặc biệt là khi số láng giềng là một số lớn và có sự không cân đối trong số lượng mẫu của từng lớp.
  - Cần thời gian lưu dữ liệu và tính toán khoảng cách, nếu dữ liệu quá lớn sẽ mất nhiều thời gian tính toán.

CHƯƠNG 3

Hiện thực

3.1 Hệ thống nhận diện khuôn mặt sử dụng KNN

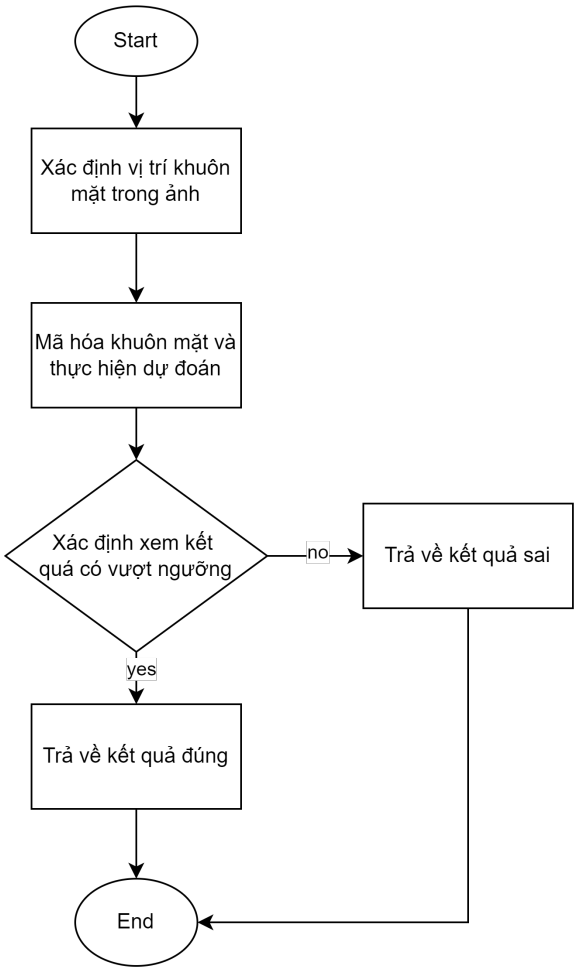
3.1.1 Cách giải thuật KNN hoạt động

Thuật toán K-Nearest Neighbors (KNN) là một thuật toán học có giám sát đơn giản, thường được sử dụng trong bài toán phân loại. Trong ngữ cảnh nhận diện khuôn mặt, KNN có thể được áp dụng như sau:

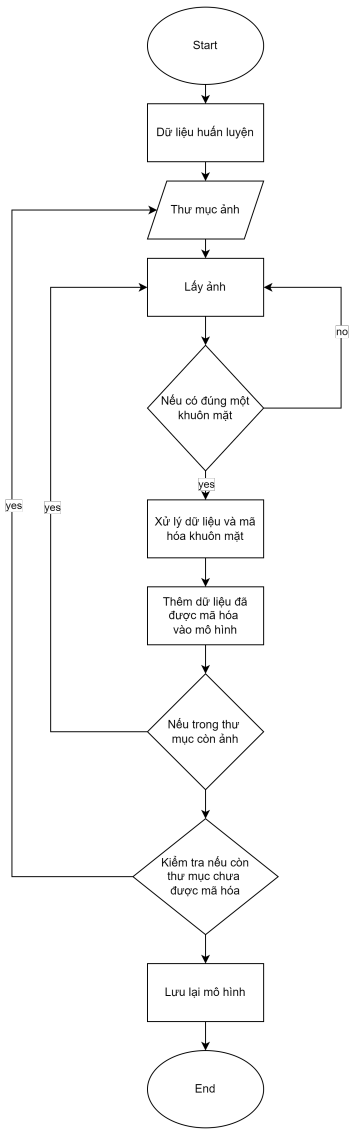
- **Huấn luyện mô hình (`train_function`):**
  1. **Xây dựng tập dữ liệu huấn luyện:**
     - Duyệt qua từng thư mục con trong thư mục huấn luyện (`train_dir`), mỗi thư mục đại diện cho một người.
     - Duyệt qua từng ảnh trong thư mục con và xác định vị trí khuôn mặt.
     - Nếu có đúng một khuôn mặt trong ảnh, mã hóa khuôn mặt đó và thêm vào danh sách đặc trưng (X) cùng với nhãn của người đó (Y).
  2. **Xác định số láng giềng (`k_neighbors`):**
     - Nếu số láng giềng không được xác định trước (`k_neighbors=None`), sẽ tự động chọn một giá trị dựa trên căn bậc hai của số lượng mẫu huấn luyện.
  3. **Tạo và huấn luyện mô hình KNN:**
     - Sử dụng thư viện `scikit-learn`, tạo một mô hình KNN với số láng giềng, thuật toán và trọng số cụ thể (chọn là `"distance"`).
     - Huấn luyện mô hình bằng cách sử dụng danh sách đặc trưng (X) và nhãn tương ứng (Y).
- **Nhận diện khuôn mặt (`predict_function`):**
  1. **Xác định vị trí khuôn mặt trong khung hình:**

- Sử dụng thư viện `face_recognition` để xác định vị trí khuôn mặt trong khung hình (`X_frame`).

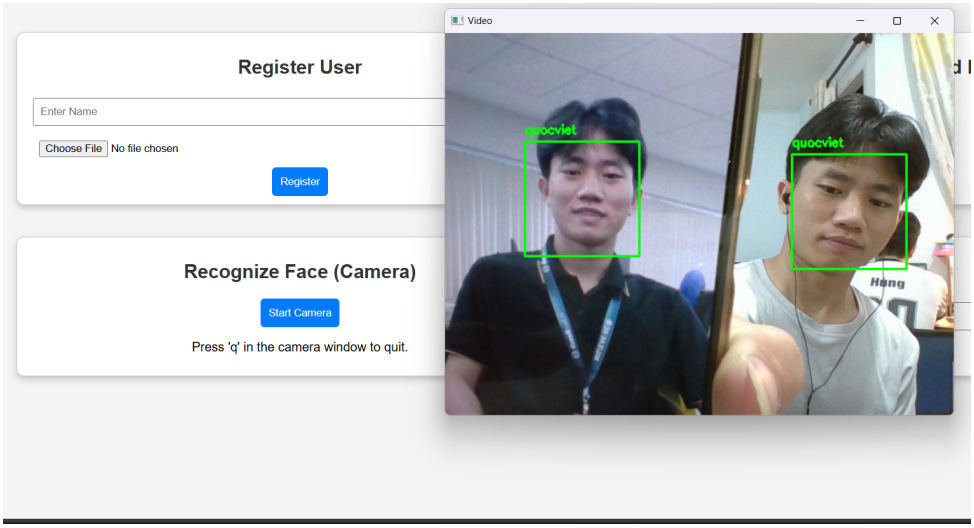
2. **Mã hóa khuôn mặt và dự đoán bằng mô hình KNN:**
   - Mã hóa đặc trưng của khuôn mặt trong khung hình bằng thư viện `face_recognition`.
   - Sử dụng mô hình KNN đã được huấn luyện để dự đoán tên của người trong khuôn mặt.
3. **Xác định xem kết quả có vượt qua ngưỡng khoảng cách không:**
   - Dùng ngưỡng khoảng cách để quyết định liệu kết quả dự đoán có đạt yêu cầu không.
4. **Trả về kết quả:**
   - Trả về một danh sách chứa tên và vị trí của khuôn mặt được nhận diện trong ảnh. Nếu không nhận diện được, trả về tên là `"Unknown"`.



Hình 3.1: Flowchart quá trình nhận diện

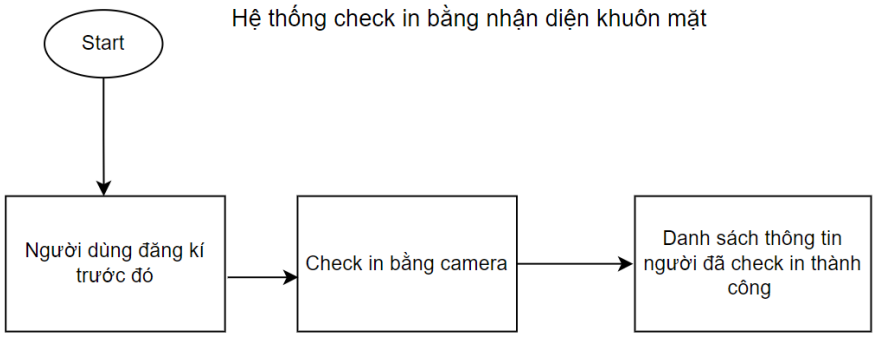


Hình 3.2: Flowchart quá trình xử lí dữ liệu



Hình 3.3: Nhận diện ảnh từ người thật và ảnh

3.2 Thiết kế hệ thống



Hình 3.4: Tổng quan thiết kế hệ thống

3.2.1 Quy trình hoạt động của hệ thống

- **Đăng ký người dùng:**
  - Người dùng nhập thông tin cá nhân (tên, ngày sinh, số điện thoại) và tải lên ảnh khuôn mặt.
  - Thông tin được lưu vào file `user_info.csv`, và ảnh được lưu trong thư mục `knn_folder/train/`.
- **Huấn luyện mô hình:**
  - Hệ thống trích xuất đặc trưng khuôn mặt từ ảnh đã đăng ký.
  - Mô hình KNN được huấn luyện và lưu vào file `trained_knn_model.clf`.
- **Check-in qua camera:**
  - Camera ghi lại khung hình, nhận diện khuôn mặt và so sánh với mô hình KNN.
  - Nếu khớp với một người dùng đã đăng ký, thông tin check-in (tên, ngày sinh, số điện thoại, thời gian) được ghi vào file `checkin_log.csv`.
- **Hiển thị danh sách check-in:**
  - Hệ thống cung cấp danh sách những người đã check-in qua giao diện web.

3.3 Hiện thực

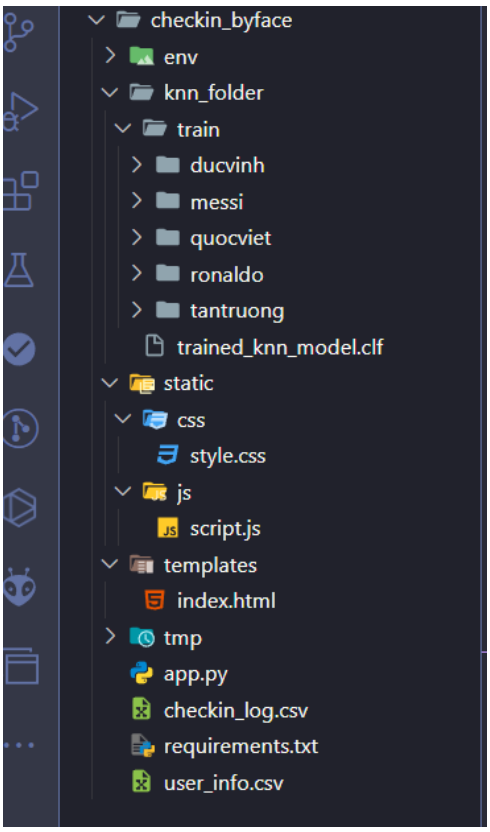
3.3.1 Cài đặt

Đảm bảo các thư viện cần thiết dưới đây đã được cài đặt và import vào dự án.

```
from flask import Flask, render_template, request, jsonify
from sklearn import neighbors
import cv2
import os
import time
import pickle
import face_recognition
import numpy as np
```

3.3.2 Cấu trúc thư mục dự án

1. **Xây dựng thư mục chưa ảnh được đăng kí**
    - Tạo một thư mục chính cho việc huấn luyện, chẳng hạn knn_folder/train.
    - Bên trong thư mục chính, tạo một thư mục cho mỗi người mà bạn muốn nhận diện và đặt tên thư mục theo tên của họ
2. **Xây dựng thư mục website quản lí**
    - Tạo thư mục templates/index.html để tạo giao diện của website đơn giản
    - Tạo thư mục statics/js và static/css để hỗ trợ định dạng web và lấy data từ lưu trữ cục bộ máy tính
3. **Tạo file app.py là file thực hiện chức năng chính của dự án**



Hình 3.5: Hình ảnh cấu trúc thư mục của dự án

3.3.3 Mô tả Source code

1. **Huấn luyện mô hình nhận diện khuôn mặt**

```python
# Paths for storing local files
TRAIN_DIR = "knn_folder/train/"
USER_INFO = "user_info.csv"
CHECKIN_LOG = "checkin_log.csv"
MODEL_PATH = "knn_folder/trained_knn_model.clf"

# Create directories and files if not exist
if not os.path.exists(TRAIN_DIR):
    os.makedirs(TRAIN_DIR)

if not os.path.exists(USER_INFO):
    with open(USER_INFO, "w") as f:
        f.write("name,dob,phone,img_path\n")

if not os.path.exists(CHECKIN_LOG):
    with open(CHECKIN_LOG, "w") as f:
        f.write("name,dob,phone,time\n")

# Train KNN model
def train_knn_model():
    X = []
    y = []

    for user in os.listdir(TRAIN_DIR):
        user_dir = os.path.join(TRAIN_DIR, user)
```

```
        if not os.path.isdir(user_dir):
            continue

        for img_path in os.listdir(user_dir):
            img_full_path = os.path.join(user_dir, img_path)
            image = face_recognition.load_image_file(img_full_path)
            face_locations = face_recognition.face_locations(image)

            if len(face_locations) == 1:
                X.append(face_recognition.face_encodings(image,
                    known_face_locations=face_locations)[0])
                y.append(user)

    knn_clf = neighbors.KNeighborsClassifier(n_neighbors=2, weights='
        distance')
    knn_clf.fit(X, y)

    with open(MODEL_PATH, 'wb') as f:
        pickle.dump(knn_clf, f)

    return knn_clf

# Load KNN model
def load_knn_model():
    if os.path.exists(MODEL_PATH):
        with open(MODEL_PATH, 'rb') as f:
            return pickle.load(f)
    return train_knn_model()

knn_model = load_knn_model()
```

Hệ thống sử dụng dữ liệu khuôn mặt đã đăng ký để huấn luyện mô hình KNN, phục vụ cho việc nhận diện khuôn mặt. Quá trình bao gồm hai bước chính:

- **Hàm `train_knn_model()`:**
  - Duyệt qua thư mục dữ liệu huấn luyện (`TRAIN_DIR`), nơi mỗi thư mục con đại diện cho một người dùng.
  - Đối với mỗi ảnh trong thư mục:
    * Tải ảnh và xác định vị trí khuôn mặt bằng thư viện `face_recognition`.
    * Nếu phát hiện chính xác một khuôn mặt, trích xuất đặc trưng khuôn mặt và lưu vào danh sách đặc trưng (X) và nhãn tương ứng (y).
  - Sử dụng thư viện `scikit-learn` để tạo và huấn luyện mô hình KNN với các tham số: $k = 2$ và trọng số `"distance"`.
  - Lưu mô hình đã huấn luyện vào file `trained_knn_model.clf` để sử dụng trong tương lai.
- **Hàm `load_knn_model()`:**
  - Tải mô hình KNN từ file `trained_knn_model.clf` nếu file tồn tại.
  - Nếu file không tồn tại, gọi hàm `train_knn_model()` để huấn luyện mô hình mới.

2. **Đăng ký người dùng mới (User Registration)**

```python
@app.route('/')
def index():
    return render_template('index.html')

# Register new user
@app.route('/register', methods=['POST'])
def register():
    name = request.form['name']
    dob = request.form['dob']
    phone = request.form['phone']
    file = request.files['file']

    if not file:
        return jsonify({"status": "error", "message": "No file uploaded"})

    # Save the image and user information
    user_dir = os.path.join(TRAIN_DIR, name)
    os.makedirs(user_dir, exist_ok=True)
    img_path = os.path.join(user_dir, f"{int(time.time())}.jpg")
    file.save(img_path)

    with open(USER_INFO, "a") as f:
        f.write(f"{name},{dob},{phone},{img_path}\n")

    # Retrain the model
    global knn_model
    knn_model = train_knn_model()

    return jsonify({"status": "success", "message": f"User {name}
       registered successfully."})
```

Giải thích cách người dùng mới được thêm vào hệ thống với thông tin cá nhân (tên, ngày sinh, số điện thoại) và ảnh khuôn mặt.

Và lưu thông tin vào file user_info.csv lưu ảnh vào thư mục knn_folder/train/.

3. **Nhận diện khuôn mặt và check-in (Face Recognition and Check-In)**

```python
        @app.route('/check_in', methods=['GET'])
def check_in():
    cap = cv2.VideoCapture(0)  #  M   camera
    if not cap.isOpened():
        return jsonify({"status": "error", "message": "Could not access the
            camera."})

    while True:
        ret, frame = cap.read()  #   c   khung h nh  t   camera
        if not ret:
            break


        small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
        rgb_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)


        face_locations = face_recognition.face_locations(rgb_frame)
        if face_locations:
            face_encodings = face_recognition.face_encodings(rgb_frame,
```

```
                face_locations)

            for encoding, location in zip(face_encodings, face_locations):
                closest_distances = knn_model.kneighbors([encoding],
                    n_neighbors=1)
                is_match = closest_distances[0][0][0] <= 0.4

                if is_match:
                    name = knn_model.predict([encoding])[0]


                    dob, phone = None, None
                    with open(USER_INFO, "r") as f:
                        for line in f.readlines()[1:]:
                            user_data = line.strip().split(",")
                            if user_data[0] == name:
                                dob, phone = user_data[1], user_data[2]
                                break


                    if dob is None or phone is None:
                        cap.release()
                        cv2.destroyAllWindows()
                        return jsonify({"status": "error", "message": f"
                            User {name} not found in user info."})


                    with open(CHECKIN_LOG, "a") as log_file:
                        log_file.write(f"{name},{dob},{phone},{time.
                            strftime('%Y-%m-%d %H:%M:%S')}\n")

                    top, right, bottom, left = location
                    top, right, bottom, left = top * 4, right * 4, bottom *
                         4, left * 4
                    cv2.rectangle(frame, (left, top), (right, bottom), (0,
                        255, 0), 2)
                    cv2.putText(frame, f"Checked In: {name}", (left, top -
                        10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
                    cv2.imshow("Camera", frame)
                    cv2.waitKey(2000)
                    cap.release()
                    cv2.destroyAllWindows()
                    return jsonify({"status": "success", "message": f"{name
                        } checked in successfully."})

        cv2.imshow("Camera", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
    return jsonify({"status": "error", "message": "No face detected. Please
         try again."})
```

Hệ thống sử dụng camera để nhận diện khuôn mặt và xác minh người dùng đã đăng ký. Quá trình hoạt động của API `/check_in` bao gồm các bước sau:

- **Mở camera và hiển thị luồng video:**
  - Hệ thống khởi tạo kết nối với camera và hiển thị khung hình trực tiếp.

  - Người dùng xuất hiện trước camera để bắt đầu quá trình nhận diện.

- **Trích xuất đặc trưng khuôn mặt:**
  - Sử dụng thư viện `face_recognition` để xác định vị trí khuôn mặt trong khung hình.
  - Trích xuất vector đặc trưng của khuôn mặt từ khung hình.
- **So sánh với mô hình KNN:**
  - Vector đặc trưng của khuôn mặt được so sánh với dữ liệu trong mô hình KNN đã huấn luyện.
  - Nếu khoảng cách nhỏ hơn ngưỡng cho phép, hệ thống xác định khuôn mặt thuộc về người dùng đã đăng ký.
- **Ghi thông tin check-in:**
  - Nếu nhận diện thành công, hệ thống ghi thông tin người dùng (tên, ngày sinh, số điện thoại, thời gian) vào file `checkin_log.csv`.
  - Nếu không nhận diện được, hệ thống chờ khuôn mặt khác hoặc người dùng nhấn `q` để thoát.
- **Hiển thị thông báo:**
  - Hệ thống hiển thị thông báo check-in thành công trên màn hình camera.
  - Nếu không nhận diện được, thông báo lỗi và yêu cầu người dùng thử lại.

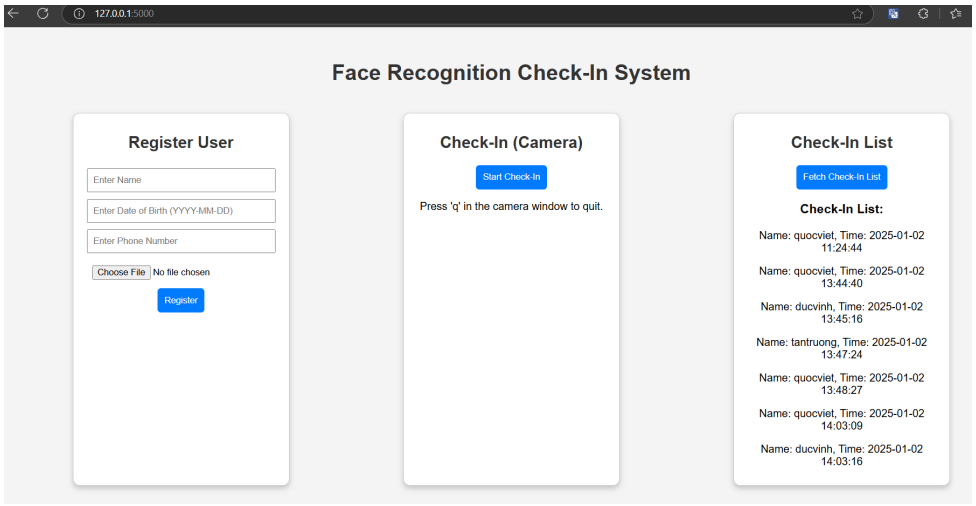
4. **Xem danh sách check-in (Check-In List)**

```
1
2  @app.route('/check_in_list', methods=['GET'])
3  def check_in_list():
4      with open(CHECKIN_LOG, "r") as log_file:
5          data = log_file.readlines()
6      check_in_data = [{"name": line.split(",")[0],
7                        "dob": line.split(",")[1],
8                        "phone": line.split(",")[2],
9                        "time": line.split(",")[3].strip()} for line in data
                             [1:]]
10     return jsonify({"status": "success", "data": check_in_data})
```

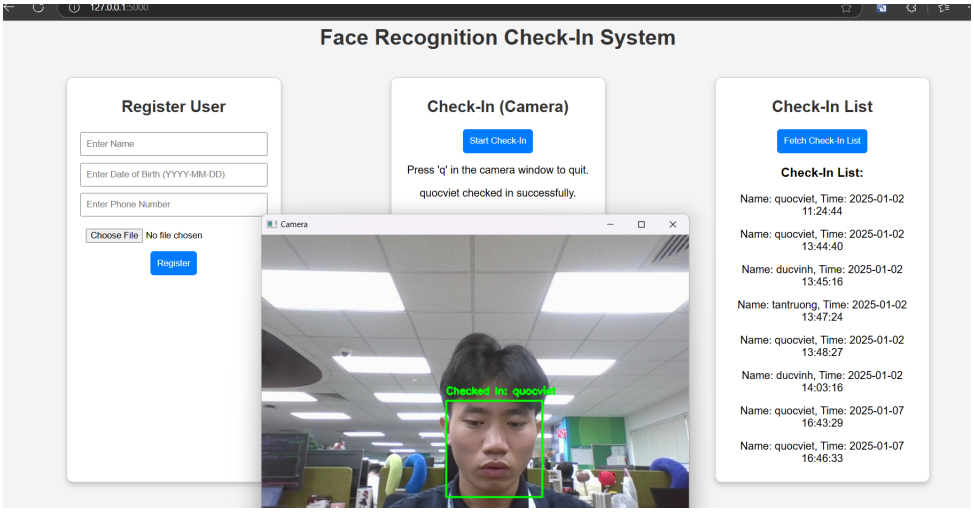
Hệ thống cung cấp danh sách những người đã check-in thông qua API `/check_in_list`. Quá trình hoạt động bao gồm:

- **Đọc dữ liệu từ file log:**
  - Hệ thống đọc nội dung từ file `checkin_log.csv`.
  - File log lưu trữ thông tin của mỗi lần check-in, bao gồm: tên, ngày sinh, số điện thoại, và thời gian check-in.

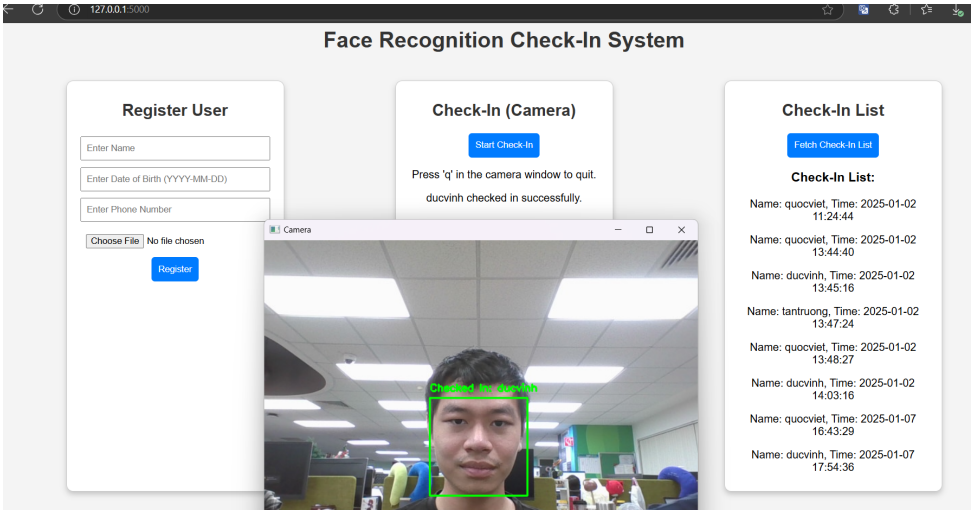
- **Xử lý và chuẩn bị dữ liệu:**
  - Hệ thống phân tích từng dòng dữ liệu trong file log để tạo danh sách các mục thông tin.
  - Các mục thông tin bao gồm:
    * Tên người dùng.
    * Ngày sinh (`dob`).
    * Số điện thoại (`phone`).
    * Thời gian check-in (`time`).
- **Trả về kết quả:**
  - Hệ thống trả danh sách check-in dưới dạng JSON.
  - Định dạng dữ liệu trả về đảm bảo dễ dàng tích hợp với giao diện người dùng hoặc các ứng dụng khác.



Hình 3.6: Trang web hiển thị thông tin đăng kí, danh sách check in



Hình 3.7: Hệ thống thông báo check in thành công và hiển thị danh sách đã check in



Hình 3.8: Hệ thống thông báo check in thành công và hiển thị danh sách đã check in

3.4 Link Source code

Source code và demo của dự án tại link:

`https://github.com/MrViet1502/Checkin_byFace.git`

CHƯƠNG 4

Tổng kết

4.1 Kết luận và hướng phát triển

4.1.1 Kết luận

Hệ thống đã hoàn thành các chức năng chính, bao gồm:

- Đăng ký người dùng (tên, ngày sinh, số điện thoại, ảnh khuôn mặt).
- Nhận diện khuôn mặt từ ảnh hoặc camera với độ chính xác cao.
- Ghi nhận thông tin check-in và quản lý log check-in.
- Hiển thị danh sách những người đã check-in trên trang web.

4.1.1.0.1 Đánh giá hệ thống:

- **Độ chính xác:** Hệ thống hoạt động hiệu quả trong điều kiện ánh sáng tốt, nhưng còn hạn chế khi ánh sáng yếu hoặc góc nhìn không thuận lợi.
- **Thời gian xử lý:** Hệ thống xử lý trong thời gian thực, với thời gian trung bình mỗi khung hình dưới $100ms$.
- **Khả năng triển khai:** Hệ thống hoạt động ổn định trên máy tính cục bộ, không yêu cầu tài nguyên tính toán lớn.

4.1.1.0.2 Hạn chế:

- Phụ thuộc vào chất lượng dữ liệu huấn luyện.
- Chưa hỗ trợ nhận diện nhiều khuôn mặt trong cùng một khung hình.
- Chưa tích hợp với cơ sở dữ liệu lớn hoặc các hệ thống đám mây.

4.1.2 Hướng phát triển

Để cải thiện và mở rộng ứng dụng, các đề xuất hướng phát triển bao gồm:

4.1.2.0.1 Cải thiện độ chính xác:

- Thu thập thêm dữ liệu huấn luyện từ nhiều góc độ và điều kiện ánh sáng khác nhau.
- Nâng cấp thuật toán nhận diện, sử dụng các mô hình như FaceNet hoặc DeepFace.

4.1.2.0.2 Tối ưu hóa hiệu suất:

- Tích hợp thuật toán xử lý ảnh để cải thiện nhận diện trong điều kiện ánh sáng yếu.
- Sử dụng GPU để tăng tốc độ xử lý trong thời gian thực.

4.1.2.0.3 Mở rộng ứng dụng:

- Tích hợp với hệ thống check-in trực tuyến cho các sự kiện lớn.
- Hỗ trợ nhận diện nhóm hoặc nhiều khuôn mặt trong một khung hình.
- Kết nối với các thiết bị mở cửa tự động.

4.1.2.0.4 Bảo mật và quyền riêng tư:

- Mã hóa dữ liệu khuôn mặt khi lưu trữ.
- Tuân thủ các quy định bảo mật như GDPR, đảm bảo quyền riêng tư của người dùng.

Tài liệu tham khảo

[1] GeeksforGeeks. *K-nearest neighbours.* https://www.geeksforgeeks.org/k-nearest-neighbours/.

[2] Wikipedia. *K-nearest neighbors algorithm.* https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.

[3] IBM. *K-nearest neighbors.* https://www.ibm.com/topics/knn.

[4] scikit-learn. *BallTree.* https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.BallTree.html.

[5] luci.vn. *Tìm hiểu về hệ thống nhận dạng khuôn mặt.* https://luci.vn/tim-hieu-ve-he-thong-nhan-dang-khuon-mat/.

[6] ResearchGate. *Face Detection & Face Recognition Using Open Computer Vision Classifies.* https://www.researchgate.net/publication/318900718_Face_Detection_Face_Recognition_Using_Open_Computer_Vision_Classifies.