

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN



HCMUTE

BÁO CÁO CUỐI KỲ
MÔN: CÁC CÔNG NGHỆ PHẦN MỀM MỚI
ĐỀ TÀI: XÂY DỰNG DESKTOP CLIENT CHO HỆ THỐNG AI NHẬN
DIỆN KHUÔN MẶT

GVHD: Huỳnh Xuân Phụng

Nhóm sinh viên thực hiện: Nhóm 29

Phạm Nguyễn Hải Dương	19110343
Nguyễn Tấn Hào	19110358
Trần Tiến Dũng	19110337

TP.Thủ Đức, tháng 12 năm 2022

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN



HCMUTE

BÁO CÁO CUỐI KỲ
MÔN: CÁC CÔNG NGHỆ PHẦN MỀM MỚI
ĐỀ TÀI: XÂY DỰNG DESKTOP CLIENT CHO HỆ THỐNG AI NHẬN
DIỆN KHUÔN MẶT

GVHD: Huỳnh Xuân Phụng

Nhóm sinh viên thực hiện: Nhóm 29

Phạm Nguyễn Hải Dương	19110343
Nguyễn Tấn Hào	19110358
Trần Tiến Dũng	19110337

TP.Thủ Đức, tháng 12 năm 2022

ĐH SƯ PHẠM KỸ THUẬT
TP.HCM
KHOA CNTT

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Họ và tên sinh viên 1: Phạm Nguyễn Hải Dương	MSSV: 19110343
Họ và tên sinh viên 2: Nguyễn Tấn Hào	MSSV: 19110358
Họ và tên sinh viên 3: Trần Tiến Dũng	MSSV: 19110337

Ngành: Công nghệ Thông tin

Tên đề tài: Xây dựng desktop client cho hệ thống AI nhận diện khuôn mặt

Họ và tên Giáo viên hướng dẫn: TS Huỳnh Xuân Phụng

NHẬN XÉT

1.Về nội dung đề tài & khối lượng thực hiện:

.....

.....

.....

2.Uưu điểm:

.....

.....

.....

3.Khuyết điểm:

.....

.....

.....

4. Đánh giá loại.....

5. Điểm:.....

Tp. Hồ Chí Minh, ngày tháng 12 năm 2021
Giáo viên hướng dẫn
(Ký & ghi rõ họ tên)

LỜI CẢM ƠN

Lời đầu tiên nhóm xin phép được gửi lời cảm ơn chân thành và sâu sắc nhất đến với Khoa Công Nghệ Thông Tin – Trường Đại Học Sư Phạm Kỹ Thuật Thành Phố Hồ Chí Minh đã tạo điều kiện cho nhóm chúng em được học tập, phát triển nền tảng kiến thức sâu sắc và thực hiện đề tài này.

Bên cạnh đó nhóm chúng em xin gửi đến thầy Huỳnh Xuân Phụng lời cảm ơn sâu sắc nhất. Trải qua một quá trình dài học tập và thực hiện đề tài trong thời gian qua. Thầy đã tận tâm chỉ bảo nhiệt tình nhóm chúng em trong suốt quá trình từ lúc bắt đầu cũng như kết thúc đề tài này.

Tuy nhiên lượng kiến thức là vô tận và với khả năng hạn hẹp chúng em đã rất cố gắng để hoàn thành một cách tốt nhất. Chính vì vậy việc xảy ra những thiếu sót là điều khó có thể tránh khỏi. Chúng em hi vọng nhận được sự góp ý tận tình của quý thầy (cô) qua đó chúng em có thể rút ra được bài học kinh nghiệm và hoàn thiện và cải thiện nâng cấp lại sản phẩm của mình một cách tốt đẹp nhất.

Cuối cùng một lần nữa chúng em xin gửi lời cảm ơn sâu sắc nhất đến với thầy Huỳnh Xuân Phụng và tập thể quý thầy (cô) Khoa Công Nghệ Thông Tin – Trường Đại Học Sư Phạm Kỹ Thuật Thành Phố Hồ Chí Minh. Chúc các thầy cô có sức khỏe thật tốt.

Chúng em xin chân thành cảm ơn!

MỤC LỤC

LỜI CẢM ƠN
CHƯƠNG 1. GIỚI THIỆU.....	1
1.1. Lý do chọn đề tài.....	1
1.2. Mục tiêu	1
1.3. Đối tượng và phạm vi	1
1.4. Bố cục của báo cáo.....	1
CHƯƠNG 2. XÁC ĐỊNH YÊU CẦU HỆ THỐNG.....	2
2.1. Danh sách yêu cầu hệ thống.....	2
2.1.1. Yêu cầu chức năng.....	2
2.1.2. Yêu cầu phi chức năng.....	2
2.2. Kiến trúc và công nghệ	3
2.2.1. Kiến trúc hệ thống.....	3
2.2.2. Công nghệ sử dụng	4
CHƯƠNG 3. CƠ SỞ LÝ THUYẾT	5
3.1. Face-api.js	5
3.1.1. Face Detection	5
3.1.2. Face Landmark Detection	5
3.1.3. Face Recognition	5
3.1.4. Face Expression Recognition.....	6
3.1.5. Age Estimation and Gender Recognition	6
3.2. React.....	6
3.3. Nodejs	7
3.4. Electron	8
CHƯƠNG 4. KẾT QUẢ ĐẠT ĐƯỢC	9
4.1. Sử dụng Electron để tạo ứng dụng desktop cho phần mềm.....	9
4.2. Controller	10

4.3. Server	11
4.3.1. Tạo Tensor	14
4.3.2. Nhận diện khuôn mặt	14
CHƯƠNG 5. KẾT LUẬN	17
5.1. Kết quả đạt được	17
5.2. Ưu điểm.....	17
5.3. Hạn chế	17
5.4. Hướng phát triển:	17
TÀI LIỆU THAM KHẢO.....	19

CHƯƠNG 1. GIỚI THIỆU

1.1. Lý do chọn đề tài

Ngày nay với sự phát triển của công nghệ thì việc nhận diện hình ảnh khuôn mặt thông qua camera đã ứng dụng vào rất nhiều trong cuộc sống chúng ta nhằm để phát hiện đối tượng, khách hàng để bảo mật, cảnh báo hay xác định. Cùng với sự phát triển của AI thì việc áp dụng các kỹ thuật machine learning vào các giải pháp bảo mật sinh trắc học (biometric security) là một trong những xu hướng AI mới nổi. Nhận thấy rằng đây là một trong những đề tài tiềm năng và có nhiều hướng phát triển, chúng em đã chọn đề tài: “Xây dựng desktop client cho hệ thống AI nhận diện khuôn mặt”

1.2. Mục tiêu

- Xây dựng 1 ứng dụng đa nền tảng. Có thể chạy trên tất cả các hệ điều hành khác nhau Linux, Windows, và MacOS.

- Yêu cầu về mặt phi chức năng: Ứng dụng có thể đáp ứng những yêu cầu cơ bản như khởi động nhanh, giao diện dễ nhìn, trực quan, dễ thao tác, và xử lý các yêu cầu của người dùng nhanh chóng.

- Yêu cầu về mặt chức năng: Xây dựng ứng dụng có thể xử lý những yêu cầu cơ bản của người dùng, phải đảm bảo về độ chính xác cao.

- Chức năng xây dựng ứng dụng có thể phát triển nhận dạng được những khuôn mặt người, có thể ước lượng được tuổi tác và nhận diện được cảm xúc của những khuôn mặt trong bức ảnh. Nâng cao hơn có thể tìm được khuôn mặt của một người nào đó trong bức ảnh. Tiếp theo không chỉ là thao tác trên hình ảnh, còn có các xử lý cho video, và webcam.

1.3. Đối tượng và phạm vi

- Đối tượng nghiên cứu: nghiên cứu nền tảng và công nghệ MERN để xây dựng ứng dụng

- Phạm vi nghiên cứu: thời gian thực hiện ứng dụng trong khuôn khổ thời gian học của học phần Các Công Nghệ Phần Mềm Mới.

1.4. Bố cục của báo cáo

CHƯƠNG 2. XÁC ĐỊNH YÊU CẦU HỆ THỐNG

2.1. Danh sách yêu cầu hệ thống

2.1.1. Yêu cầu chức năng

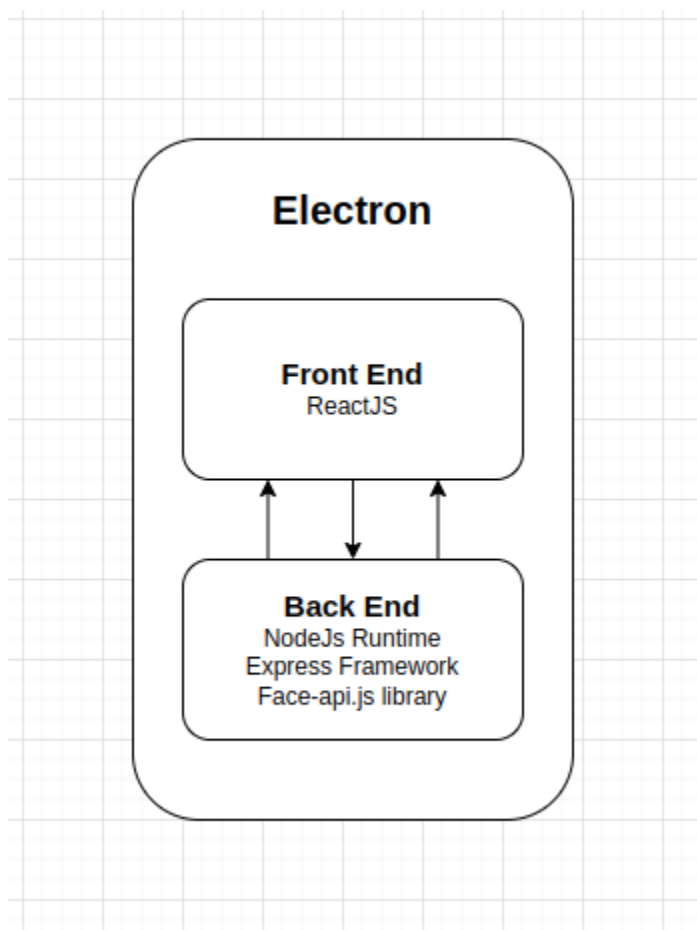
STT	Nội dung	Mô tả chi tiết	Ghi chú
1	Nhận diện khuôn mặt qua hình ảnh	Dựa theo hình ảnh được người dùng đưa lên, chương trình sẽ kẻ khung được khuôn mặt nhận diện và đưa ra cho người dùng.	
2	Xác định khuôn mặt được định sẵn	Người dùng sẽ lựa chọn một khuôn mặt mà họ muốn tìm, sau đó khi người dùng đưa lên một hình ảnh chứa nhiều khuôn mặt, chương trình sẽ chỉ kẻ khung khuôn mặt mà người dùng đã định sẵn.	
3	Dự đoán tuổi tác và giới tính qua khuôn mặt	Dựa theo hình ảnh được người dùng đưa lên, chương trình sẽ tính toán và đưa ra tuổi tác và giới tính gần nhất.	
4	Nhận diện đường nét của khuôn mặt	Dựa theo ảnh người dùng đưa lên, chương trình sẽ theo những điểm viền của khuôn mặt mà kẻ ra một đường đường nét.	
5	Nhận cảm xúc của khuôn mặt	Dựa theo hình ảnh được người dùng đưa lên, chương trình sẽ nhận diện cảm xúc của người dùng và hiển thị lên.	

2.1.2. Yêu cầu phi chức năng

STT	Nội dung	Tiêu chuẩn	Mô tả chi tiết	Ghi chú
1	Giao diện thân thiện, đơn giản.	Tiện dụng	Khách hàng ở mọi lứa tuổi đều có thể dễ dàng sử dụng.	Giao thân thiện, đơn giản.
2	Dễ thao tác.	Tiện dụng	Các thao tác nhập xuất chỉnh sửa đơn giản, gần gũi với người dùng.	Dễ thao tác.
3	Tốc độ phản hồi nhanh.	Hiệu quả	Tốc độ xử lý và phản hồi nhanh. Không tạo sự khó chịu cho người dùng.	Phản hồi dưới 0.5s.
4	Khả năng bảo trì, nâng cấp hệ thống.	Dễ dàng	Khả năng nâng cấp hệ thống dễ dàng trong tương lai.	

2.2. Kiến trúc và công nghệ

2.2.1. Kiến trúc hệ thống



2.2.2. Công nghệ sử dụng

Dependency: Python 2.7

Ngôn ngữ sử dụng: JavaScript

Runtime Environment: Nodejs

Framework: TensorFlow, React

Platform: Electron

CHƯƠNG 3. CƠ SỞ LÝ THUYẾT

3.1. Face-api.js

TensorFlow.js là một Thư viện JavaScript để đào tạo (training) và triển khai (deploying) các mô hình (model) học máy trong trình duyệt (browser) và trong Node.js.

Face-api là một thư viện giúp cho chúng ta thực hiện các công việc như phát hiện khuôn mặt và nhận diện khuôn mặt trên trình duyệt và nó được triển khai trên lõi của tensorflow.js

Hiện tại thì chúng ta có thể sử dụng được 5 model mà face-api cung cấp đó là:

3.1.1. Face Detection

SSD Mobilenet V1: Sử dụng để phát hiện khuôn mặt, bằng cách sử dụng SSD (Single Shot Multibox Detector) dựa trên MobileNetV1. Nó sẽ trả về một ô vuông giới hạn khuôn mặt và xác suất cho mỗi gương mặt mà nó phát hiện được. Model này giúp cho việc phát hiện khuôn mặt được nhanh hơn và có độ chính xác cao hơn. Kích cỡ của model khoảng 5.4 MB (ssd_mobilenetv1_model). Model này được "trained" bởi WIDERFACE dataset

Tiny Face Detector: hỗ trợ realtime face detector với thời gian nhanh hơn, kích thước nhỏ hơn, tốn ít tài nguyên hơn để so sánh với SSD Mobilenet V1 face detector, đổi lại nó hoạt động kém hơn khi phát hiện các khuôn mặt nhỏ. Model này cực kỳ thân thiện với model và web, phù hợp với các thiết bị hạn chế tài nguyên. Kích cỡ 190KB (tiny_face_detector_model).

3.1.2. Face Landmark Detection

Nó giúp phát hiện 68 điểm trên khuôn mặt của bạn một cách cực kỳ nhẹ và nhanh chóng mà lại còn chính xác nữa chứ. Kích cỡ 350KB (face_landmark_68_model) và model nhỏ chỉ 80KB (face_landmark_68_tiny_model). Cả 2 model được "trained" trong bộ dữ liệu ~35k hình ảnh khuôn mặt được gắn nhãn với 68 mốc khuôn mặt.

3.1.3. Face Recognition

Sử dụng cho việc nhận diện khuôn mặt, một kiến trúc gần giống như ResNet-34 được triển khai để tính toán một bộ mô tả khuôn mặt (một vector đặc trưng có 128 giá trị) từ các hình ảnh khuôn mặt đã được cung cấp, cái mà sử dụng để mô tả các nét đặc

trung của khuôn mặt. Model này không giới hạn ở bộ khuôn mặt được sử dụng để "training", có nghĩa là bạn có thể sử dụng nó để nhận dạng khuôn mặt của bất kỳ người nào, chẳng hạn như khuôn mặt của bạn. Bạn có thể xác định sự giống nhau của hai khuôn mặt tùy ý bằng cách so sánh các mô tả khuôn mặt của chúng. Ví dụ bằng cách tính khoảng cách euclide hoặc sử dụng bất kỳ bộ phân loại nào khác mà bạn chọn. Kích cỡ của model này là 6.2MB (face_recognition_model).

3.1.4. Face Expression Recognition

Model này giúp chúng ta có thể nhận dạng được những biểu cảm trên khuôn mặt một cách nhanh chóng và cực kỳ nhẹ cùng với độ chính xác cao. Nó được "training" bởi các nguồn được public trên mạng. Lưu ý rằng độ chính xác có thể giảm xuống nếu bạn đeo kính. Kích cỡ 310Kb

3.1.5. Age Estimation and Gender Recognition

Model này giúp chúng ta ước lượng tuổi tác cũng như nhận diện giới tính. Kích thước 420KB

3.2. React

Nó là một thư viện JavaScript được thiết kế để tạo các single-page applications với các thành phần giao diện người dùng (UI) có thể tái sử dụng (Reusable).

Sử dụng ReactJS, người dùng có thể:

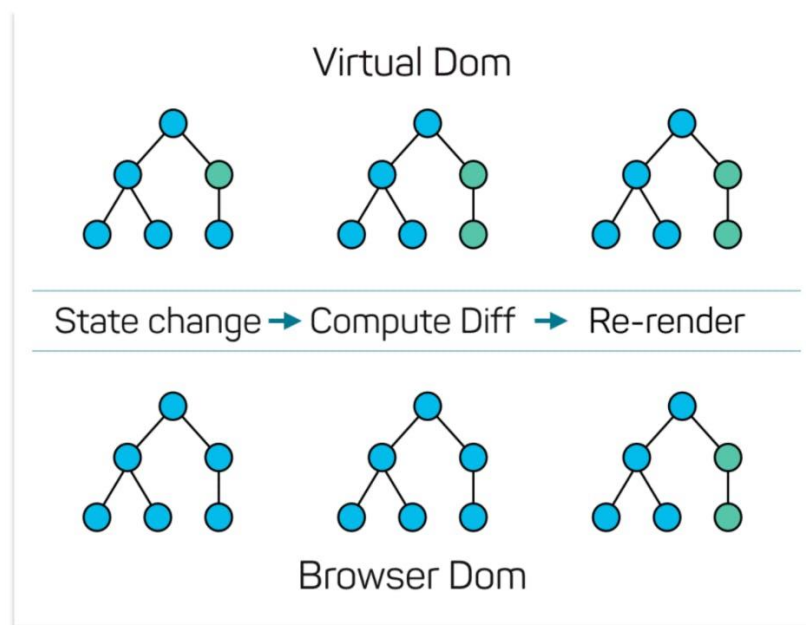
- Viết ứng dụng trực tiếp trên JavaScript.
- Phá vỡ những cấu tạo UI phức tạp và biến chúng thành những component độc lập.
- Chuyển các dữ liệu đã được tùy biến đến một UI component cụ thể.
- Thay đổi trạng thái cho nhiều component (child) trên ứng dụng nhưng không ảnh hưởng tới component gốc (parent) đang ở trạng thái Stateful.

Virtual DOM: Đây là một định dạng dữ liệu JavaScript nhẹ được dùng để thể hiện nội dung của DOM (Document Object Model, tạm dịch: Mô hình Đối tượng Tài liệu) tại một thời điểm nhất định nào đó. ReactJS là Framework sử dụng Virtual-DOM. Vừa là Model, vừa là View, tại DOM, mọi sự thay đổi trên Model đã kéo theo sự thay đổi trên View và ngược lại. Hiểu đơn giản, dù không tác động trực tiếp vào các phần tử DOM ở View, bạn vẫn sẽ thực hiện được cơ chế Data-binding. Nhờ vậy, tốc độ ứng

dùng tăng lên đáng kinh ngạc.

Component: Nếu các Framework khác dùng template thì ReactJS lại được xây dựng xung quanh các component. Sử dụng phương thức `createClass` (nhận một tham số mô tả đặc tính), bạn sẽ dễ dàng tạo ra một component sở hữu đầy đủ đặc tính đó.

- Cách hoạt động của React:



React lưu trữ thông tin của DOM bằng cách tạo virtual DOM trong bộ nhớ của nó. Trước khi render các DOM trên trình duyệt, nó sẽ kiểm tra các thay đổi giữa virtual DOM trong quá khứ và hiện tại. Nếu có thay đổi, nó sẽ cập nhật virtual DOM và sau đó hiển thị thành DOM thực trên trình duyệt.

3.3. Nodejs

Node.JS là một runtime mã nguồn mở JavaScript sử dụng công cụ JavaScript V8. Nó cho phép chạy JavaScript bên ngoài trình duyệt và cũng cung cấp một shell tương tác nơi bạn có thể thực thi mã JavaScript thô.

Node.js cung cấp kiến trúc hướng sự kiện (event-driven) và non-blocking I/O API, tối ưu hóa thông lượng của ứng dụng và có khả năng mở rộng cao

Mọi hàm trong Node.js là không đồng bộ (asynchronous). Do đó, các tác vụ đều được xử lý và thực thi ở chế độ nền (background processing)

Dự án Node.JS ban đầu được phát hành vào năm 2009 dưới dạng runtime đa nền tảng, open source để phát triển các ứng dụng phía máy chủ bằng JavaScript. Node.js đi

kèm với trình quản lý gói được gọi là npm, đây là hệ sinh thái lớn nhất của các thư viện nguồn mở.

3.4. Electron

Electron là Framework (open source bởi Github) cho phép viết desktop app chạy trên mọi nền tảng (Mac, Window, Linux) dựa trên công nghệ web (Nodejs, HTML và CSS).

Ứng dụng đầu tiên được xây dựng bằng Electron là trình soạn thảo Atom của Github. Bên cạnh đó còn rất nhiều sản phẩm nổi tiếng khác cũng được xây dựng bằng công nghệ Electron, 1 trong số đó phải kể đến Slack app trên Win và trên Mac.

CHƯƠNG 4. KẾT QUẢ ĐẠT ĐƯỢC

4.1. Sử dụng Electron để tạo ứng dụng desktop cho phần mềm

A screenshot of a code editor with a dark background and light-colored text. The code is written in JavaScript and uses Electron to create a desktop application. It includes comments in Vietnamese. The code is as follows:

```
1  const {app, BrowserWindow} = require('electron')
2  function createWindow () {
3      const mainWindow = new BrowserWindow({
4          width: 1600,
5          height: 900,
6          webPreferences: {
7              nodeIntegration: true
8          }
9      })
10
11     mainWindow.loadURL("http://localhost:3002")
12 }
13 app.whenReady().then(() => {
14     createWindow()
15     app.on('activate', function () {
16         if (BrowserWindow.getAllWindows().length === 0) createWindow()
17     })
18 })
19 app.on('window-all-closed', function () {
20     if (process.platform !== 'darwin') app.quit()
21 })
```

Đầu tiên là cũng là phần quan trọng nhất của ứng dụng, ở đây sẽ điều khiển các thành phần cũng như là tạo một cửa sổ window như một ứng dụng của desktop. Ở phần code này ở dòng 1 là khai báo những thành phần quan trọng nhất của nền tảng Electron.

Hàm được khai báo ở dòng số 2 là hàm cực kì quan trọng, trong hàm này có thêm một biến `mainWindow` để chỉnh những cấu hình của app như độ dài độ rộng. Ở dòng 11 sử dụng để import nội dung của web có địa chỉ là <http://localhost:3002>.

Tiếp tục là dòng 13 ở đây là phương thức để được gọi khi electron được gọi xong.

Tiếp theo là ở dòng 14 sẽ được gọi để tạo nên cửa sổ ứng dụng. Câu điều kiện ở dòng 16 dùng để chặn việc tạo thêm 1 cửa sổ mới ở trên hệ điều hành MacOS.

4.2. Controller

```
1
2 exports.faceDetection = async (req, res) => {
3     const {img} = req.files
4     const result = await faceApiService.face(img.data)
5     res.json({"result": result})
6 }
7
8 exports.landmarkDetection = async (req, res) => {
9     const {img} = req.files
10    const result = await faceApiService.landmark(img.data)
11    res.json({"result": result})
12 }
13
14 exports.ageAndGenderDetection = async (req, res) => {
15     const {img} = req.files;
16     const result = await faceApiService.ageGender(img.data)
17     res.json({"result": result})
18 }
19
20 exports.expressionDetection = async (req, res) => {
21     const {img } = req.files
22     const result = await faceApiService.expression(img.data)
23     res.json({"result": result})
24 }
```

Đây là code của 4 hàm khác nhau. Ở mỗi hàm lần lượt là khai báo tên hàm, tiếp theo là lấy file từ front-end gửi lên. Ở dòng tiếp theo là sử dụng các hàm trong service để thực hiện theo chức năng tương ứng. Ở dòng cuối cùng thì dùng để trả về client với dạng json.



```
1 exports.faceRecognition = async (req, res) => {
2   const {image1, image2} = req.files;
3   const result = await faceApiService.recognition(image1.data, image2.data)
4   res.json({"result": result})
5 }
```

Ở đây là hàm dùng để nhận diện khuôn mặt. Với đầu vào là hai hình ảnh, với image1 là hình ảnh gốc dùng để so sánh, image2 là hình ảnh dùng để kiểm tra.

4.3. Server



```
1 const express = require("express");
2 const cors = require("cors");
3 const path = require("path")
4 const bodyParser = require("body-parser")
5 const fileUpload = require("express-fileupload")
6
7 const imageRouter = require("../router/ImageRouter")
8
9 // config
10 const app = express();
11 const PORT = process.env.PORT || 3002;
12
13 //middleware
14 app.use(
15   cors({
16     origin: ["http://localhost:3000"],
17   })
18 );
19 app.use(fileUpload())
20 app.use(bodyParser.urlencoded({extended:true}))
21 app.use(bodyParser.json())
22
23 // view
24 app.use(express.static(path.resolve(__dirname, "../client/build")))
```


Ở phần code này gồm những phần riêng biệt như phần khai báo những module cần để sử dụng, khai báo router, cấu hình port, tường lửa, và phần view.



```
1 app.use(imageRouter)
2 app.listen(PORT, () => {
3   console.log(`App listen on port ${PORT}`);
4 });
```

Đây là dòng khai báo sử dụng imageRouter đã được import ở trên.

Tiếp theo là hàm listen. Theo mặc định thì web sẽ sử dụng 3002 làm port mặc định.



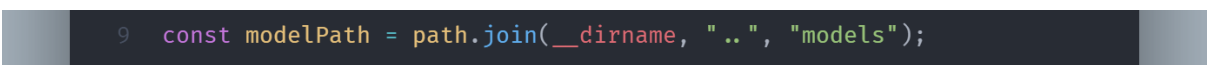
```
1 const tf = require("@tensorflow/tfjs-node");
2 const faceapi = require("@vladmandic/face-api/dist/face-api.node");
3 const canvas = require("canvas");
```

Từ dòng 1 - 3: khai báo những module cần thiết.



```
5 const { Canvas, Image, ImageData } = canvas;
```

5: destruction các thành phần của biến canvas



```
9 const modelPath = path.join(__dirname, "..", "models");
```

9: khai báo biến dẫn đến thư mục chứa những model đã được pre-train

```

11 Promise.all([
12   faceapi.nets.ssdMobilenetv1.loadFromDisk(modelPath),
13   faceapi.nets.faceRecognitionNet.loadFromDisk(modelPath),
14   faceapi.nets.faceLandmark68Net.loadFromDisk(modelPath),
15   faceapi.nets.faceLandmark68TinyNet.loadFromDisk(modelPath),
16   faceapi.nets.ageGenderNet.loadFromDisk(modelPath),
17   faceapi.nets.faceExpressionNet.loadFromDisk(modelPath),
18 ])
19 .then((val) => {
20   console.log(val);
21 })
22 .catch((err) => {
23   console.log(err);
24   process.exit(1);
25 });

```

11 - 25: sử dụng để các mạng tải các trọng số cũng như là giữ liệu về mô hình.
 Tổng thể:

```

1  const tf = require("@tensorflow/tfjs-node");
2  const faceapi = require("@vladmandic/face-api/dist/face-api.node");
3  const canvas = require("canvas");
4
5  const { Canvas, Image, ImageData } = canvas;
6  faceapi.env.monkeyPatch({ Canvas, Image, ImageData });
7
8  const path = require("path");
9  const modelPath = path.join(__dirname, "..", "models");
10
11 Promise.all([
12   faceapi.nets.ssdMobilenetv1.loadFromDisk(modelPath),
13   faceapi.nets.faceRecognitionNet.loadFromDisk(modelPath),
14   faceapi.nets.faceLandmark68Net.loadFromDisk(modelPath),
15   faceapi.nets.faceLandmark68TinyNet.loadFromDisk(modelPath),
16   faceapi.nets.ageGenderNet.loadFromDisk(modelPath),
17   faceapi.nets.faceExpressionNet.loadFromDisk(modelPath),
18 ])
19 .then((val) => {
20   console.log(val);
21 })
22 .catch((err) => {
23   console.log(err);
24   process.exit(1);
25 });

```

4.3.1. Tạo Tensor

```
1  const image = async (file) => {  
2    const decoded = tf.node.decodeImage(file);  
3    const casted = decoded.toFloat();  
4    const result = casted.expandDims(0);  
5    decoded.dispose();  
6    casted.dispose();  
7    return result;  
8  };  
9
```

Đây là chức năng dùng để tạo ra các tensor. Sử dụng các tensor này để nhận dạng.

4.3.2. Nhận diện khuôn mặt

```
1  const detectFace = async (img) => {  
2    const tensor = await image(img);  
3    const result = await faceapi.detectAllFaces(tensor);  
4    const canvasImg = await canvas.loadImage(img);  
5    const out = await faceapi.createCanvasFromMedia(canvasImg);  
6    await faceapi.draw.drawDetections(out, result);  
7  
8    return await out.toBuffer("image/png");  
9  };
```

Ví dụ với hàm này dùng để nhận dạng tất cả các khuôn mặt có trong một hình ảnh được gửi lên từ front-end. Các bước thực hiện như sau:

- Bước 1: tạo 1 tensor từ hình ảnh được gửi lên
- Bước 2: sẽ sử dụng lib để nhận dạng các khuôn mặt có trong tensor và kết quả sẽ được lưu vào biến result.
- Bước 3: sẽ tạo 1 canvas image
- Bước 4: từ canvas đã được tạo ở trên sẽ tạo một canvas
- Bước 5: từ biến canvas đã được tạo ở bước 4. Sử dụng thư viện faceapi để vẽ các khung của khuôn mặt được nhận dạng.
- Bước 6: trả về controller giá trị cuối cùng là một bức ảnh nhưng sẽ chuyển về dạng buffer



```
1 const data = new Uint8Array(res.data.result.data);
2 const blob = new Blob([data], { type: "image/jpeg" });
3 const file = URL.createObjectURL(blob);
```

Từ mảng các buffer được trả về ở trên thì client sẽ chuyển thành hình ảnh kiểu Blob và từ đó có thể show ra trên ứng dụng.

Những hàm còn lại sẽ từ mẫu này đi ra:

```
const detectExpression = async (img) => {
  const canvasImg = await canvas.loadImage(img);
  const out = await faceapi.createCanvasFromMedia(canvasImg);

  const tensor = await image(img);
  const result = await faceapi
    .detectAllFaces(tensor)
    .withFaceLandmarks()
    .withFaceExpressions();

  await faceapi.draw.drawDetections(out, result);
  await faceapi.draw.drawFaceExpressions(out, result, 0.5);

  return await out.toBuffer("image/png");
};
```

```
const detectAgeAndGender = async (img) => {
  const canvasImg = await canvas.loadImage(img);
  const out = await faceapi.createCanvasFromMedia(canvasImg);

  const tensor = await image(img);
  const result = await faceapi
    .detectAllFaces(tensor)
    .withFaceLandmarks()
    .withAgeAndGender();

  result.forEach((ele) => {
    let gender = ele.gender;
    let age = ele.age;
    let box = ele.detection.box;
    const drawBox = new faceapi.draw.DrawBox(box, {
      label: `Gender ${gender} - Age: ${age}`,
    });
    drawBox.draw(out);
  });

  return await out.toBuffer("image/png");
};
```

CHƯƠNG 5. KẾT LUẬN

Sau một thời gian thực hiện báo cáo dưới sự hướng dẫn của thầy Huỳnh Xuân Phụng báo cáo của em đã thực hiện tốt được các mục tiêu đề ra và đạt được những kết quả sau:

5.1. Kết quả đạt được

Khi nghiên cứu về hệ thống nhận dạng này chúng tôi không những nghiên cứu về thuật toán xử lý mà qua đó chúng tôi đã biết thêm nhiều về các công trình nghiên cứu về khuôn mặt người từ các lĩnh vực khác nhau và kết quả này thực sự có ý nghĩa trong việc nâng cao hiệu quả khả năng giao tiếp giữa con người và máy tính. Xây dựng được ứng dụng nhận diện khuôn mặt với các chức năng :

- Phát hiện khuôn mặt
- Phát hiện biểu cảm khuôn mặt
- Phát hiện đường nét khuôn mặt
- Dự đoán độ tuổi và giới tính.

5.2. Ưu điểm

- Phát triển được tracking face (Nhận diện khuôn mặt được định sẵn với độ chính xác khá lớn.
- Phát triển thêm được một chức năng dùng webcam để phát triển khuôn mặt.

5.3. Hạn chế

Với điều kiện thời gian cũng như kinh nghiệm hạn chế, khả năng dịch và hiểu tài liệu chưa tốt nên nội dung đồ án này còn nhiều thiếu sót, rất mong được sự chỉ bảo, góp ý của thầy để đồ án có thể phát triển trong tương lai cũng như kinh nghiệm để áp dụng vào thực tế.

5.4. Hướng phát triển:

Với mong muốn hoàn thành tốt và tìm hiểu rõ đề tài này, trong thời gian tới nhóm chúng em sẽ tiếp tục tìm hiểu, nghiên cứu sâu hơn các vấn đề của kiểm thử phần mềm và triển thêm một số tính năng như;

- Phát hiện khuôn mặt thông qua video.
- Áp dụng được vào những ứng dụng thực tiễn như bảo mật sinh trắc học hay xác định tội phạm trong đám đông qua

Đồ án có thể tiếp tục tìm hiểu sâu hơn về các vấn đề kiểm thử phần mềm để có thể ứng dụng kiểm thử thực tế cho các ứng dụng lớn sau này.

TÀI LIỆU THAM KHẢO

[1] GitHub - justadudewhohacks/face-api.js: *JavaScript API for face detection and face recognition in the browser and nodejs with tensorflow.js.*

<https://github.com/justadudewhohacks/face-api.js>

[2] Quang An, L. (2022, December 16). *Tìm hiểu thư viện face-api thông qua viết ứng dụng Face Recognition.* Viblo. <https://viblo.asia/p/tim-hieu-thu-vien-face-api-thong-qua-viet-ung-dung-face-recognition-4P856A1BIY3>

[3] Huong, N. X. (2022, December 16). *Xây dựng desktop app bằng Electron.* Viblo. <https://viblo.asia/p/xay-dung-desktop-app-bang-electron-maGK78wxZj2>