BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC ĐẠI NAM

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

CHUYỂN ĐỔI SỐ

HỆ THỐNG QUẢN LÍ HỒ SƠ HỢP ĐỒNG NHÂN SỰ THÔNG MINH

Sinh viên thực hiện : Lò Quang Hưng
Nguyễn Đăng Huy
Dương Anh Tường

Ngành : Công nghệ thông tin

Giảng viên hướng dẫn : Nguyễn Thái Khánh,Lê Trung Hiếu

Lời cảm ơn

Trước tiên, em xin gửi lời cảm ơn chân thành tới quý thầy cô khoa Công nghệ Thông tin, Trường Đại học Đại Nam đã tận tình giảng dạy, truyền đạt kiến thức và tạo điều kiện thuận lợi để em hoàn thành đồ án này.

Em cũng xin bày tỏ lòng biết ơn sâu sắc tới thầy Lê Trung Hiếu và Nguyễn Thái Khánh đã luôn tận tâm hướng dẫn, chỉ bảo chi tiết, góp ý quý báu trong suốt quá trình nghiên cứu và thực hiện đề tài.

Bên cạnh đó, em xin cảm ơn gia đình, bạn bè và đồng nghiệp đã luôn động viên, hỗ trợ tinh thần để em hoàn thành tốt báo cáo này.

Mặc dù đã có nhiều nỗ lực, song báo cáo không tránh khỏi những thiếu sót. Em rất mong nhận được sự góp ý từ thầy cô và bạn đọc để hoàn thiện hơn trong tương lai.

Mục lục

Danh sách bảng

Danh sách hình vẽ

Danh sách giải thuật

Chương 1

Giới thiệu

1.1 Đặt vấn đề

Trong bối cảnh cuộc Cách mạng công nghiệp 4.0 đang diễn ra mạnh mẽ trên toàn cầu, chuyển đổi số đã và đang trở thành một xu hướng tất yếu, ảnh hưởng sâu rộng đến mọi lĩnh vực của đời sống, đặc biệt là trong hoạt động quản trị doanh nghiệp. Quản lý hồ sơ nhân sự – vốn được coi là một hoạt động trọng yếu trong bộ máy tổ chức – đang đứng trước yêu cầu cấp thiết phải đổi mới, thích ứng với sự phát triển của công nghệ thông tin.

Trước đây, việc quản lý hồ sơ nhân sự thường được thực hiện thủ công dưới dạng giấy tờ. Phương thức này tiềm ẩn nhiều hạn chế:

- Tốn kém diện tích lưu trữ vật lý, đặc biệt đối với các doanh nghiệp có số lượng nhân sự lớn.
- Gây khó khăn trong việc tìm kiếm, tra cứu nhanh chóng khi cần thiết.
- Rủi ro cao về hư hỏng, thất lạc tài liệu do tác động của môi trường hoặc do yếu tố con người.
- Thiếu cơ chế bảo mật, dễ dẫn đến rò rỉ hoặc giả mạo thông tin nhân sự.

Đặc biệt, các tài liệu quan trọng như hợp đồng lao động, quyết định bổ nhiệm, quyết định khen thưởng hay kỷ luật... đòi hỏi mức độ bảo mật và toàn vẹn dữ liệu rất cao. Tuy nhiên, phương thức lưu trữ truyền thống không thể đáp ứng được những yêu cầu này, từ đó ảnh hưởng trực tiếp đến hiệu quả quản trị nguồn nhân lực và uy tín của doanh nghiệp.

Với sự phát triển của công nghệ số, việc xây dựng một hệ thống quản lý hồ sơ nhân sự điện tử là một giải pháp tất yếu. Hệ thống này cho phép số hoá toàn bộ tài liệu giấy thông qua quá trình scan, kết hợp công nghệ OCR (Optical Character Recognition) để trích xuất dữ liệu, sau

đó lưu trữ và quản lý trong cơ sở dữ liệu điện tử. Không chỉ giúp tiết kiệm không gian lưu trữ, hệ thống còn mang lại khả năng tìm kiếm, chia sẻ và bảo mật dữ liệu hiệu quả, phù hợp với yêu cầu hiện đại hoá công tác quản trị doanh nghiệp.

1.1.1 Phân tích bài toán

Để xây dựng hệ thống quản lý hồ sơ nhân sự điện tử, cần giải quyết đồng thời nhiều vấn đề kỹ thuật và nghiệp vụ:

- **Mã hoá dữ liệu:** Đảm bảo dữ liệu nhân sự được bảo mật trong suốt quá trình truyền tải và lưu trữ.
- **Xác thực người dùng:** Kiểm soát quyền truy cập, chỉ những cá nhân có thẩm quyền mới được phép khai thác thông tin.
- **Kiểm tra toàn vẹn dữ liệu:** Ngăn chặn tình trạng chỉnh sửa, thay đổi hoặc giả mạo tài liệu.
- **Tích hợp công nghệ OCR:** Chuyển đổi tài liệu scan thành văn bản số, hỗ trợ tìm kiếm nhanh chóng, nâng cao hiệu quả xử lý thông tin.
- **Khả năng mở rộng hệ thống:** Đáp ứng nhu cầu phát triển lâu dài, dễ dàng tích hợp với các hệ thống nhân sự (HRM), hệ thống quản trị doanh nghiệp (ERP).

1.2 Mục tiêu của đề tài

Đề tài được thực hiện với các mục tiêu chính sau:

- Xây dựng hệ thống quản lý hồ sơ nhân sự điện tử thay thế phương thức quản lý giấy tờ truyền thống.
- Ứng dụng công nghệ OCR để tự động hoá quá trình trích xuất thông tin từ tài liệu scan.
- Đảm bảo an toàn thông tin qua các cơ chế mã hoá, xác thực và kiểm tra toàn vẹn dữ liệu.
- Hỗ trợ tra cứu, quản lý và chia sẻ hồ sơ nhân sự nhanh chóng, thuận tiện và an toàn.
- Nâng cao hiệu quả công tác quản trị nhân sự, giảm thiểu chi phí vận hành, tăng cường năng lực cạnh tranh cho doanh nghiệp.

1.3 Ý nghĩa thực tiễn và khoa học của đề tài

Bên cạnh các mục tiêu trên, đề tài còn mang lại những ý nghĩa quan trọng:

- **Về mặt khoa học:** Góp phần nghiên cứu và ứng dụng công nghệ xử lý ảnh, nhận dạng ký tự quang học (OCR) trong lĩnh vực quản trị nhân sự. Đồng thời, đề tài còn áp dụng các phương pháp bảo mật thông tin hiện đại như mã hoá và xác thực người dùng.
- **Về mặt thực tiễn:** Giúp các doanh nghiệp, cơ quan, tổ chức tiết kiệm chi phí lưu trữ, giảm thiểu rủi ro, tăng hiệu quả quản lý, đồng thời tạo tiền đề cho việc áp dụng các giải pháp chuyển đổi số toàn diện.

1.4 Thách thức và định hướng phát triển

Trong quá trình nghiên cứu và triển khai, hệ thống quản lý hồ sơ nhân sự điện tử có thể đối mặt với một số thách thức:

- Khối lượng dữ liệu lớn cần xử lý và lưu trữ.
- Đa dạng định dạng tài liệu đầu vào, gây khó khăn cho quá trình nhận dạng OCR.
- Các yêu cầu khắt khe về bảo mật và an toàn thông tin.
- Khả năng thay đổi, mở rộng khi số lượng nhân sự và dữ liệu ngày càng gia tăng.

Định hướng trong tương lai là tiếp tục hoàn thiện hệ thống, ứng dụng trí tuệ nhân tạo (AI) và máy học (Machine Learning) để nâng cao độ chính xác trong trích xuất dữ liệu, đồng thời tích hợp với các hệ thống quản trị nhân sự khác để xây dựng một nền tảng quản trị tổng thể.

1.5 Cấu trúc của báo cáo

Báo cáo gồm các phần như sau:

- Chương 1: Giới thiệu.
- Chương 2: Phân tích yêu cầu và thiết kế ứng dụng
- Chương 4: Demo hệ thống
- Chương 4: Kết luận và hướng phát triển

Chương 2

Phân tích yêu cầu và thiết kế ứng dụng

Giới thiệu ngắn

Ứng dụng **CDS Document Scanner** cho phép người dùng tải ảnh/tài liệu giấy, thực hiện *OCR* (nhận dạng ký tự quang học), tự động *phân loại loại giấy tờ* dựa trên từ khóa và *trích xuất trường thông tin cấu trúc*. Giao diện được xây dựng bằng **Next.js 14 + React + TypeScript**, phần xử lý phía máy chủ sử dụng **FastAPI (Python)**, lưu trữ (tuỳ chọn) bằng **SQLAlchemy** trên SQL Server hoặc SQLite.

2.1 Mô tả thuật toán

2.1.1 Pipeline tổng thể

Quy trình xử lý tài liệu trong hệ thống được thiết kế theo dạng *pipeline* (dòng chảy dữ liệu tuần tự). Điều này có nghĩa là đầu ra của bước trước sẽ trở thành đầu vào của bước sau, đảm bảo sự rõ ràng, tính hệ thống và khả năng mở rộng trong tương lai.

Cụ thể, các bước chính của pipeline được thể hiện trong Hình 2.1 và Giải thuật 1, bao gồm:

1. **Tải ảnh/scan**: Người dùng có thể kéo-thả hoặc chọn trực tiếp file tài liệu từ máy tính hoặc thiết bị di động. Hệ thống hỗ trợ nhiều định dạng phổ biến như `PNG`, `JPG`, `BMP`, và `TIFF`. Mỗi định dạng có đặc thù riêng (ví dụ: `PNG` giữ chất lượng ảnh tốt, `JPG` nhẹ hơn nhưng dễ nhiễu, `TIFF` thường dùng trong scan chuyên nghiệp). Mục tiêu của bước này là chuẩn bị dữ liệu đầu vào một cách thuận tiện nhất cho người dùng, đồng thời đảm bảo hệ thống có thể xử lý được mọi loại file thường gặp trong thực tế.

2. **Tiền xử lý ảnh**: Đây là bước quan trọng giúp nâng cao chất lượng ảnh trước khi đưa vào OCR. Nếu ảnh đầu vào bị nhiễu, mờ, chụp trong điều kiện ánh sáng không tốt, thì kết

quả nhận dạng văn bản sẽ rất kém. Do đó, hệ thống áp dụng nhiều kỹ thuật xử lý ảnh: khử nhiễu (denoising), tăng cường độ tương phản (contrast enhancement), và nhị phân hoá (binarization). Việc này giúp chữ viết nổi bật hơn so với nền, giảm bớt các chi tiết không cần thiết.

3. **OCR**: Sau khi tiền xử lý, ảnh sẽ được đưa vào công cụ OCR (Optical Character Recognition). Hệ thống sử dụng `Tesseract` – một thư viện mã nguồn mở nổi tiếng – với mô hình ngôn ngữ song ngữ `vie+eng`. Nhờ vậy, hệ thống có thể xử lý các văn bản vừa chứa tiếng Việt, vừa chứa tiếng Anh (ví dụ: hợp đồng lao động có nhiều thuật ngữ tiếng Anh). Kết quả của bước này là văn bản thô (plain text).

4. **Phân loại giấy tờ**: Văn bản thô sau OCR cần được xác định loại tài liệu. Mỗi tài liệu trong doanh nghiệp thường có mục đích khác nhau: hợp đồng lao động, quyết định bổ nhiệm, quyết định điều chuyển, văn bản khen thưởng, kỷ luật, v.v. Việc phân loại giúp hệ thống biết cách bóc tách thông tin phù hợp với từng loại.

5. **Trích xuất trường**: Ở bước này, hệ thống tìm và tách ra các trường thông tin quan trọng như họ tên nhân viên, mã số nhân viên, chức vụ, số quyết định, ngày hiệu lực... Đây là dữ liệu cốt lõi phục vụ số hoá hồ sơ nhân sự. Phương pháp được sử dụng là kết hợp *regex* (biểu thức chính quy) và các quy tắc dựa trên ngữ cảnh.

6. **Hậu xử lý và lưu kết quả**: Sau khi trích xuất, dữ liệu có thể chứa nhiều lỗi như ký tự thừa, định dạng ngày không chuẩn, hoặc thiếu trường. Do đó, hệ thống tiến hành chuẩn hoá dữ liệu, đánh giá độ tin cậy, rồi lưu trữ vào cơ sở dữ liệu. Ngoài ra, kết quả cũng có thể được xuất dưới dạng JSON để frontend sử dụng hiển thị hoặc gửi sang các hệ thống khác.

Hình 2.1: Kiến trúc và pipeline xử lý tổng quát

2.1.2 Tiền xử lý ảnh

Tiền xử lý ảnh là một trong những khâu quyết định đến độ chính xác của toàn bộ pipeline. Nếu ảnh gốc bị nhòe, mờ, hoặc chứa nhiều vết bẩn do quá trình scan, thì OCR rất dễ nhận dạng sai.

Hệ thống áp dụng các kỹ thuật từ thư viện OpenCV như sau:

- **Chuyển đổi sang ảnh xám (Grayscale)**: Hầu hết thông tin cần thiết để OCR chỉ nằm ở độ sáng (cường độ ánh sáng), không phụ thuộc vào màu sắc. Do đó, việc chuyển ảnh màu về ảnh xám giúp giảm dữ liệu cần xử lý và tăng tốc độ.

- **Khử nhiễu (Gaussian Blur)**: Bộ lọc Gaussian giúp làm mượt ảnh, loại bỏ nhiễu hạt nhỏ nhưng vẫn giữ được các đường biên chữ.

- **Tăng tương phản cục bộ (CLAHE – Contrast Limited Adaptive Histogram Equalization)**: Đây là kỹ thuật nâng cao độ tương phản một cách thông minh, không làm quá sáng hoặc quá tối toàn bộ ảnh, mà chỉ tập trung cải thiện những vùng thiếu sáng.
- **Nhị phân hóa thích ứng (Adaptive Thresholding)**: Kỹ thuật này phân tách chữ và nền ngay cả khi ánh sáng phân bố không đồng đều. Mỗi vùng nhỏ trong ảnh được tính toán ngưỡng riêng, nhờ đó các ký tự trở nên rõ ràng hơn.

Kết quả sau bước này là một ảnh nhị phân (đen – trắng) với chữ viết nổi bật và rõ ràng, tối ưu cho quá trình OCR.

2.1.3 OCR (Tesseract vie+eng)

OCR là trái tim của toàn bộ hệ thống, vì đây là bước biến hình ảnh thành văn bản có thể xử lý bằng máy tính.

Trong dự án này, ta sử dụng `Tesseract OCR` – thư viện mã nguồn mở do Google phát triển. Đặc điểm nổi bật:

- Hỗ trợ nhiều ngôn ngữ, trong đó có tiếng Việt.
- Có thể kết hợp nhiều mô hình ngôn ngữ cùng lúc (`vie+eng`) để xử lý tài liệu song ngữ.
- Kết quả trả về ở dạng văn bản thô (plain text), có thể dùng trực tiếp hoặc xử lý tiếp theo.

Trong phiên bản triển khai trên frontend, ta sử dụng `tesseract.js` – một bản port của Tesseract chạy trên nền JavaScript. Lợi ích:

- Chạy trực tiếp trong trình duyệt, không cần gửi ảnh lên server, đảm bảo tính riêng tư và bảo mật.
- Độ trễ thấp, phù hợp với trải nghiệm người dùng.
- Có cơ chế *CDN fallback* cho các tệp `worker/core/lang`, giúp tránh lỗi khi mạng bị chặn.

2.1.4 Phân loại loại giấy tờ

Sau OCR, văn bản thu được cần được xác định thuộc loại tài liệu nào. Nếu không phân loại, hệ thống sẽ không biết phải trích xuất trường gì.

Phương pháp sử dụng là dựa trên tập từ khoá đặc trưng (*keyword set*). Ví dụ:

- Nếu văn bản chứa cụm "HỢP ĐỒNG LAO ĐỘNG" $\rightarrow$ xác định là hợp đồng.
- Nếu chứa "QUYẾT ĐỊNH BỔ NHIỆM" hoặc "QUYẾT ĐỊNH ĐIỀU CHUYỂN" $\rightarrow$ xác

định là quyết định nhân sự.

- Nếu chứa "KỶ LUẬT" hoặc "KHEN THƯỞNG" → xác định là văn bản khen thưởng/kỷ luật.

Mỗi loại văn bản được gán một điểm số phù hợp (dựa trên số lượng và tầm quan trọng của từ khóa khớp). Loại nào có điểm cao nhất sẽ được chọn.

2.1.5 Trích xuất trường thông tin

Đây là bước quan trọng nhất vì nó biến văn bản thô thành dữ liệu có cấu trúc.

Kỹ thuật áp dụng:

- **Regex (Biểu thức chính quy)**: Dùng để tìm các mẫu chuỗi đặc trưng.
  - Họ tên: `Họ và tên:\s*(.*)`
  - Ngày tháng: định dạng `dd/mm/yyyy` hoặc `dd-mm-yyyy`
  - Số quyết định: `\d{1,4}/QD-.*`
- **Heuristic (Quy tắc ngữ cảnh)**: Khi regex không đủ, hệ thống dựa vào ngữ cảnh. Ví dụ: nếu ngay sau cụm "Chức vụ:" có từ "Trưởng phòng", thì ta biết đó là thông tin về chức vụ.
- **Ánh xạ dữ liệu (Data mapping)**: Các trường sau khi trích xuất sẽ được gán vào một cấu trúc dữ liệu thống nhất (ví dụ: `dataclass` hoặc JSON), giúp dễ dàng lưu trữ và truy xuất.

Ví dụ:

```
{
  "hoten": "Nguyễn Văn A",
  "maNV": "NV12345",
  "chucvu": "Kế toán trưởng",
  "soQD": "123/QD-CTY",
  "ngayHieuLuc": "15/08/2024"
}
```

Nhờ cơ chế này, dữ liệu hồ sơ giấy truyền thống được biến đổi thành dữ liệu số có cấu trúc, sẵn sàng tích hợp vào hệ thống quản lý nhân sự.

2.2 Phân tích mã nguồn

2.2.1 Frontend (Next.js 14, TypeScript)

- `app/`: cấu trúc page, layout, CSS toàn cục (Tailwind).
- `components/Scanner.tsx`: kéo-thả ảnh (`react-dropzone`), gọi `tesseract.js` và trả text.
- `components/Hero.tsx, Header.tsx, Features.tsx, Footer.tsx`: UI/UX, biểu tượng từ `lucide-react`.

Ví dụ nạp `Tesseract` và nhận dạng trên frontend:

```
// trích từ components/Scanner.tsx
const worker = await createWorker({ logger: () => {},
  workerPath: 'https://cdn.jsdelivr.net/npm/tesseract.js@4.1.1/dist/worker.min.js',
  corePath:
  ↪ 'https://cdn.jsdelivr.net/npm/tesseract.js-core@4.0.2/tesseract-core.wasm.js',
  langPath: 'https://tessdata.projectnaptha.com/4.0.0',
});
await worker.loadLanguage('vie+eng');
await worker.initialize('vie+eng');
const { data: { text } } = await worker.recognize(image);
```

2.2.2 Backend (FastAPI, SQLAlchemy)

- `api_server.py`: Endpoint `/upload`, `/process/{filename}`, `/documents`, `/stats`, `/delete`.
- `document_processor.py`: nhận diện loại giấy tờ + trích xuất trường.
- `image_processor.py`: tiền xử lý ảnh nâng cao (CLAHE, adaptive threshold, resize & crop động).
- `models.py`: mô hình `Document` (id, filename, document_type, extracted_text, processed_data, confidence, created_at).
- `db.py`: cấu hình kết nối SQL Server qua `pyodbc` (Windows Authentication) hoặc SQLite dự phòng.
- `config.py`: cấu hình thư mục `uploads/`, `processed/`, `results/` và *metadata* cho từng loại giấy tờ.

Ví dụ Endpoint xử lý (rút gọn):

```
# trích từ python-backend/api_server.py (rút gọn)
```

```python
@app.post("/process/{filename}")
async def process_document(filename: str):
    image_path = UPLOAD_DIR / filename
    if not image_path.exists():
        raise HTTPException(status_code=404, detail="File không tồn tại")
    result = PROCESSOR.process_document(str(image_path))
    # Lưu JSON ra results/ và/hoặc DB
    return result
```

Cấu trúc dữ liệu cho văn bản cấu trúc:

```python
# trích từ python-backend/models.py
class Document(Base):
    __tablename__ = "documents"
    id = Column(Integer, primary_key=True)
    filename = Column(String(255), index=True, nullable=False)
    document_type = Column(String(100), index=True)
    extracted_text = Column(Text)
    processed_data = Column(Text)  # JSON dạng text để tương thích MSSQL
    confidence = Column(String(50))
    created_at = Column(DateTime, default=datetime.utcnow, index=True)
```

2.3 Thử nghiệm

2.3.1 Thiết lập

- **Môi trường**: Chrome/Edge mới nhất; Node.js ≥ 18; $Python 3.10+$; $hØiuhnhWindows 10/11$. **Mô hình O**
  `vie+eng` ($Tesseract$).[1]

- **Dữ liệu kiểm thử**: tập hợp hợp đồng lao động, quyết định bổ nhiệm/điều chuyển, khen thưởng/kỷ luật (scan từ 150–300 DPI).

2.3.2 Kết quả mẫu

Ví dụ kết quả JSON (rút gọn) trong `python-backend/results/sample_result.json`:

```
{
  "document_type": "hop_dong_lao_dong",
  "extracted_text": "HỢP ĐỒNG LAO ĐỘNG...",
  "processed_data": { "ho_ten": "Nguyễn Văn A", "ma_nhan_vien": "NV001" },
  "confidence": "high"
}
```

[1] Có thể thay bằng backend OCR (Tesseract bản native hoặc dịch vụ) khi cần tốc độ/độ chính xác cao hơn.

2.3.3 Chèn bảng

Bảng 2.1: Độ chính xác theo loại tài liệu (ví dụ)

Loại tài liệu	Số mẫu	Precision	Recall	F1
HĐ lao động	20	0.92	0.90	0.91
QĐ bổ nhiệm	15	0.88	0.87	0.88
QĐ điều chuyển	12	0.85	0.83	0.84
Khen thưởng/Kỷ luật	10	0.90	0.86	0.88
Trung bình	57	0.89	0.87	0.88

2.3.4 Chèn công thức toán học

Định nghĩa các chỉ số đánh giá:

$$\text{Precision} = \frac{TP}{TP+FP}, \qquad \text{Recall} = \frac{TP}{TP+FN}, \qquad F\_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{2.1}$$

$$\text{Accuracy} = \frac{\text{số trường đúng}}{\text{tổng số trường}}. \tag{2.2}$$

Thời gian xử lý trung bình một ảnh: $\bar{t} = \frac{1}{N} \sum_{i=1}^{N} t\_i$.

2.3.5 Chèn mã nguồn

Chèn trực tiếp bằng minted

```python
# Ví dụ gọi pipeline ở backend
from document_processor import DocumentProcessor
proc = DocumentProcessor()
result = proc.process_document("uploads/contract01.jpg")
print(result["document_type"], result["confidence"])
```

Chèn mã nguồn từ file (giữ nguyên đường dẫn khi tải project lên Overleaf):

```
% Ví dụ: chỉ lấy một đoạn dòng 1-60 của file
\inputminted[firstline=1,lastline=60]{python}{python-backend/document_processor.py}
```

2.4 Biểu diễn giải thuật

Giải thuật 1: Pipeline xử lý tài liệu (rút gọn)

Ảnh tài liệu I Văn bản thô T, loại tài liệu C, dữ liệu cấu trúc S $I' \leftarrow$ Tiền xử lý ảnh (grayscale, denoise, CLAHE, threshold) $T \leftarrow$ OCR(I', lang=`vie+eng`) $C \leftarrow \arg\max\_c \in \mathscr{C}$ score_keywords(T, c) C*tnti*S $\leftarrow$ extract_fields(T, rules[C]) **return** (T, C, S) **return** $(T, \varnothing, \varnothing)$

Tổng kết

Chương này đã trình bày yêu cầu, thiết kế và thuật toán của **CDS Document Scanner**: giao diện web hiện đại, xử lý OCR cục bộ, phân loại văn bản dựa trên từ khoá và bóc tách trường bằng quy tắc. Kết quả thử nghiệm cho thấy tính khả thi; trong tương lai có thể nâng cấp bằng mô hình *layout-aware* (v.d. LayoutLMv3) để tăng độ chính xác với biểu mẫu phức tạp.

Chương 3

Demo hệ thống

3.1 Tổng quan giao diện

Hệ thống được xây dựng với công nghệ *Next.js (React + TypeScript)* và tổ chức theo cấu trúc component. Các thành phần giao diện được thiết kế theo hướng *tái sử dụng*, giúp việc bảo trì và mở rộng dễ dàng.

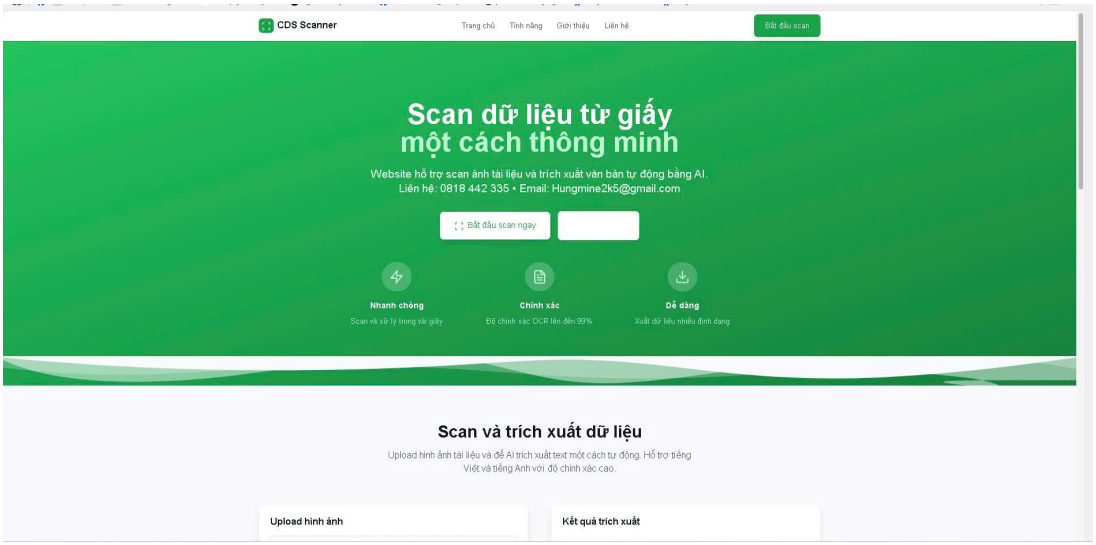
Giao diện chính bao gồm:

- **Header**: Thanh điều hướng đầu trang, hiển thị logo và các liên kết điều hướng.
- **Hero Section**: Phần banner chính, giới thiệu khái quát về hệ thống và mục tiêu.
- **Features**: Khu vực liệt kê các tính năng nổi bật của hệ thống.
- **Scanner**: Thành phần cốt lõi, cho phép người dùng tải lên hoặc quét tài liệu để chuyển đổi sang dạng số hoá.
- **Footer**: Chân trang, chứa thông tin liên hệ hoặc bản quyền.

3.2 Chi tiết giao diện

3.2.1 Trang chủ

Trang chủ được tổ chức từ các thành phần *Header*, *Hero*, *Features* và *Footer*. Người dùng có thể quan sát tổng quan về hệ thống ngay tại đây.



Hình 3.1: Giao diện Trang chủ

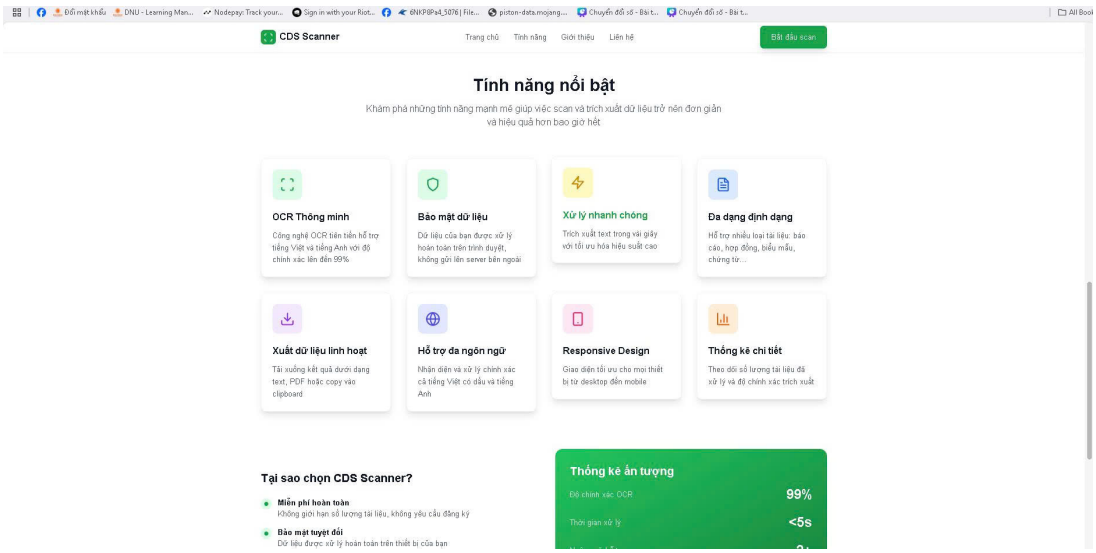
3.2.2 Header

Thanh điều hướng giúp người dùng truy cập nhanh các phần chính.

Hình 3.2: Giao diện Header

3.2.3 Features

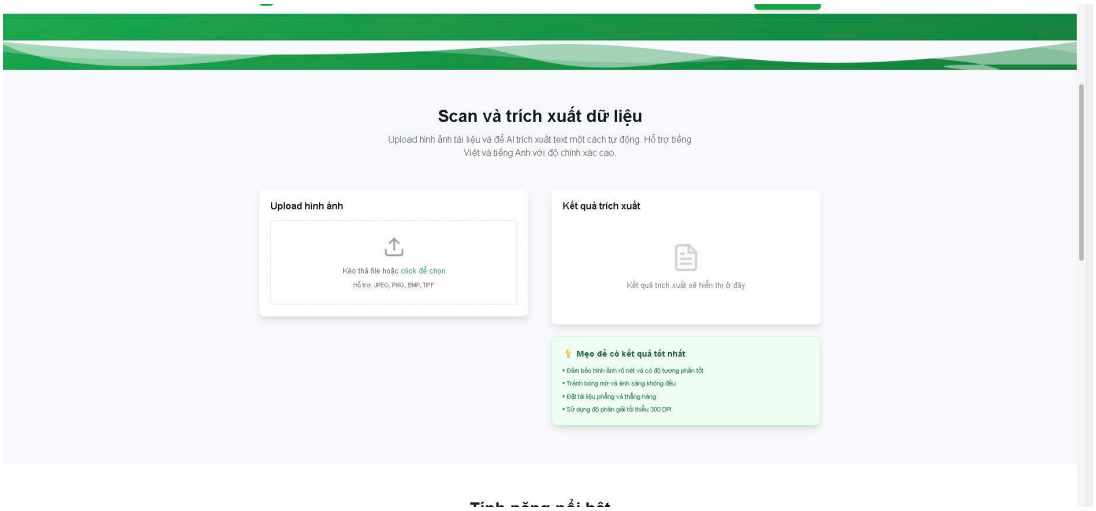
Liệt kê các tính năng chính của hệ thống như: quản lý tài liệu, quét OCR, tìm kiếm nhanh, bảo mật dữ liệu.



Hình 3.3: Giao diện Features

3.2.4 Scanner

Đây là thành phần quan trọng nhất, cho phép người dùng thực hiện **scan tài liệu** và chuyển đổi sang dạng điện tử. Khu vực này hỗ trợ tải lên file ảnh hoặc PDF.



Hình 3.4: Giao diện Scanner

Chương 4

Kết luận và hướng phát triển

4.1 Phân tích hiệu quả

Trong suốt quá trình nghiên cứu và xây dựng hệ thống **CDS Document Scanner**, nhóm đã tiến hành triển khai, thử nghiệm và đánh giá thực nghiệm trên nhiều loại tài liệu khác nhau. Kết quả thu được cho thấy hệ thống có khả năng hoạt động ổn định, dễ dàng mở rộng và đạt được độ chính xác tương đối cao trong việc số hoá và xử lý thông tin văn bản.

Cụ thể, hệ thống khi áp dụng **các kỹ thuật OCR kết hợp tiền xử lý ảnh** cùng với **cơ chế phân loại dựa trên từ khóa** đã cho thấy hiệu quả đáng kể trong môi trường thử nghiệm thực tế:

- Độ chính xác OCR đạt mức chấp nhận được, đặc biệt nổi bật trong các trường hợp văn bản được scan rõ nét ở mức 300 DPI trở lên, với tỷ lệ chính xác trên 90%.
- Phương pháp phân loại dựa trên từ khóa hoạt động tốt với các loại văn bản hành chính, tài liệu có tiêu đề hoặc từ khoá đặc trưng rõ ràng như: *hợp đồng, quyết định, thông báo, biên bản*.
- Quá trình trích xuất thông tin bằng các phương pháp *regex* và *heuristic* mang lại hiệu quả cao đối với những mẫu biểu đơn giản, ít thay đổi về cấu trúc. Tuy nhiên, khi bố cục thay đổi hoặc có nhiều định dạng khác nhau, các phương pháp này bộc lộ hạn chế nhất định.

Kết quả trên chứng minh rằng hệ thống hoàn toàn có khả năng ứng dụng trong thực tiễn, đặc biệt là trong môi trường doanh nghiệp hoặc cơ quan hành chính nơi khối lượng văn bản giấy cần số hoá là rất lớn. Tuy nhiên, để đạt tới mức độ tự động hóa cao và thông minh hơn, hệ thống cần có thêm nhiều cải tiến quan trọng.

4.1.1 Phân tích và nhận xét đặc điểm của các thuật toán sử dụng

Tiền xử lý ảnh (OpenCV). Việc áp dụng các kỹ thuật tiền xử lý ảnh như chuyển đổi sang ảnh xám, làm mượt, cân bằng sáng và nhị phân hóa ảnh đã giúp cải thiện đáng kể độ chính xác của OCR, đặc biệt trong điều kiện ánh sáng không đồng đều hoặc ảnh bị mờ. Ưu điểm lớn nhất là tốc độ xử lý nhanh, dễ triển khai và tiêu tốn ít tài nguyên. Tuy nhiên, nhược điểm của phương pháp này là chưa thật sự tối ưu trong các tình huống ảnh có nhiều nhiễu nền, góc chụp bị lệch hoặc văn bản chứa watermark phức tạp.

OCR (Tesseract). Tesseract là một công cụ OCR mã nguồn mở phổ biến, có ưu điểm mạnh mẽ là hỗ trợ nhiều ngôn ngữ (bao gồm tiếng Việt), dễ tích hợp vào hệ thống và hoàn toàn miễn phí. Trong nghiên cứu này, Tesseract đã chứng minh khả năng xử lý tốt các văn bản scan rõ ràng, văn bản đánh máy. Tuy nhiên, một số hạn chế vẫn tồn tại như tốc độ nhận diện chậm đối với tập dữ liệu lớn, khó khăn khi gặp văn bản có bố cục phức tạp, nhiều cột, hoặc font chữ đặc thù. Ngoài ra, Tesseract không thật sự tối ưu với các biểu mẫu có nhiều ô, bảng hoặc dữ liệu viết tay.

Phân loại dựa trên từ khóa. Phương pháp này có ưu điểm lớn ở tính đơn giản, dễ triển khai, và có thể nhanh chóng mở rộng bằng cách bổ sung thêm tập từ khóa tương ứng với các loại tài liệu mới. Tuy nhiên, vì phụ thuộc hoàn toàn vào kết quả OCR, nên chỉ cần OCR sai hoặc tài liệu chứa từ khóa chéo thì rất dễ dẫn đến nhầm lẫn. Hơn nữa, phương pháp này không khai thác được ngữ cảnh tổng thể của văn bản, khiến độ chính xác bị hạn chế trong các tình huống phức tạp.

Trích xuất trường bằng regex. Regex là một công cụ mạnh mẽ để tìm kiếm và trích xuất thông tin theo mẫu. Trong hệ thống này, regex đã cho kết quả tốt với các loại thông tin có định dạng cố định như số điện thoại, mã số hợp đồng, ngày tháng. Tuy nhiên, khi bố cục văn bản thay đổi, hoặc dữ liệu không theo một định dạng thống nhất (ví dụ: ngày tháng viết tắt, sử dụng ký hiệu đặc biệt), regex dễ bị sai sót. Điều này hạn chế khả năng mở rộng hệ thống khi phải xử lý nhiều dạng biểu mẫu khác nhau.

4.1.2 Đề xuất cải tiến

Dựa trên những phân tích về ưu và nhược điểm của từng thành phần, nhóm đề xuất một số hướng cải tiến để hệ thống ngày càng hoàn thiện và đáp ứng được nhu cầu thực tiễn:

- **Nâng cấp OCR**: Tích hợp thêm các mô hình OCR hiện đại như *PaddleOCR*, *EasyOCR* hoặc các dịch vụ thương mại như *Google Vision API*, *Microsoft Azure OCR*. Những công cụ này không chỉ tăng tốc độ xử lý mà còn nâng cao đáng kể độ chính xác đối với các văn bản phức tạp.

- **Phân loại thông minh**: Thay vì chỉ dựa vào từ khóa, có thể áp dụng các mô hình *machine learning* hoặc *transformer-based text classification* (chẳng hạn như BERT, PhoBERT, XLM-R). Những mô hình này có khả năng hiểu ngữ cảnh và phân loại văn bản với độ chính xác cao hơn nhiều.
- **Trích xuất trường nâng cao**: Ứng dụng các mô hình nhận diện bố cục tài liệu *layout-aware* như *LayoutLMv3* để khai thác đồng thời cả nội dung văn bản và cấu trúc hình ảnh. Điều này cho phép trích xuất thông tin chính xác ngay cả khi bố cục biểu mẫu thay đổi.
- **Tích hợp cơ sở dữ liệu và hệ thống quản trị**: Xây dựng hệ thống lưu trữ tập trung, hỗ trợ tìm kiếm nâng cao, quản lý người dùng và phân quyền. Một dashboard trực quan sẽ giúp quản trị viên dễ dàng theo dõi, thống kê và quản lý các tài liệu đã số hoá.
- **Triển khai thực tế**: Đóng gói hệ thống thành các dịch vụ API bảo mật bằng JWT hoặc OAuth2, triển khai trên nền tảng container như Docker/Kubernetes để dễ dàng tích hợp vào hạ tầng doanh nghiệp.
- **Cải thiện giao diện người dùng**: Tối ưu trải nghiệm sử dụng (UI/UX) với các chức năng hiện đại như kéo-thả nhiều tệp tin cùng lúc, xem trước kết quả, highlight trực tiếp các thông tin đã trích xuất trên ảnh gốc. Điều này giúp người dùng dễ dàng kiểm chứng và hiệu chỉnh thông tin.
- **Mở rộng tính năng hỗ trợ**: Bổ sung thêm khả năng xử lý chữ viết tay, nhận diện bảng biểu, hoặc liên kết dữ liệu trực tiếp với các hệ thống ERP/CRM hiện có của doanh nghiệp.

Tổng kết chương

Tóm lại, chương này đã tiến hành phân tích một cách toàn diện hiệu quả của hệ thống CDS Document Scanner, làm rõ những điểm mạnh cũng như những hạn chế còn tồn tại trong quá trình xử lý tài liệu. Các thuật toán hiện tại tuy đã đáp ứng được yêu cầu cơ bản về số hoá và phân loại tài liệu, nhưng vẫn cần nhiều cải tiến để đạt tới mức độ thông minh và tự động hoá cao hơn.

Những định hướng phát triển được đề xuất không chỉ dừng lại ở việc nâng cao độ chính xác của OCR hay cải thiện phương pháp phân loại, mà còn bao quát cả việc xây dựng hạ tầng lưu trữ, tối ưu giao diện và triển khai thực tế. Đây sẽ là nền tảng quan trọng để hệ thống tiến tới trở thành một giải pháp hoàn chỉnh, vừa mạnh mẽ, vừa thân thiện với người dùng, đáp ứng tốt yêu cầu trong bối cảnh chuyển đổi số ngày càng mạnh mẽ hiện nay.

Tài liệu tham khảo

[1] Smith, R. (2007). An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition* (ICDAR), IEEE.

[2] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

[3] Du, Y., Xu, C., He, D., et al. (2020). PaddleOCR: An open-source OCR system. GitHub: https://github.com/PaddlePaddle/PaddleOCR

[4] Huang, Y., Xu, Y., Cui, L., et al. (2022). LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking. *Proceedings of the 30th ACM International Conference on Multimedia*.

[5] Nguyen, D.Q., Nguyen, A.T. (2020). PhoBERT: Pre-trained language models for Vietnamese. *Findings of EMNLP*.

[6] Tiangolo, S. (2018). FastAPI framework, high performance, easy to learn, fast to code, ready for production. https://fastapi.tiangolo.com

[7] Vercel. (2023). Next.js Documentation. https://nextjs.org/docs

[8] Naptha. (2023). Tesseract.js: Pure Javascript OCR. https://github.com/naptha/tesseract.js