**ĐẠI HỌC ĐÀ NẴNG**
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**
**KHOA CÔNG NGHỆ THÔNG TIN**

# ĐỒ ÁN TỐT NGHIỆP

## NGÀNH: CÔNG NGHỆ THÔNG TIN
## CHUYÊN NGÀNH: CÔNG NGHỆ THÔNG TIN

**ĐỀ TÀI:**

# JOBSWIPE - ỨNG DỤNG AI TRONG HỆ THỐNG TÌM VIỆC

Người hướng dẫn: **ThS. MAI VĂN HÀ**
Sinh viên thực hiện: **HUỲNH HẢI ĐĂNG**
Số thẻ sinh viên: **102210341**
Lớp**:** **21TCLC_Nhật 1**

**Đà Nẵng, 06/2025**

**ĐẠI HỌC ĐÀ NẴNG**
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**
**KHOA CÔNG NGHỆ THÔNG TIN**

# ĐỒ ÁN TỐT NGHIỆP

## NGÀNH: CÔNG NGHỆ THÔNG TIN
## CHUYÊN NGÀNH: CÔNG NGHỆ THÔNG TIN

**ĐỀ TÀI:**

# JOBSWIPE - ỨNG DỤNG AI TRONG HỆ THỐNG TÌM VIỆC

Người hướng dẫn: **ThS. MAI VĂN HÀ**
Sinh viên thực hiện: **HUỲNH HẢI ĐĂNG**
Số thẻ sinh viên: **102210341**
Lớp**:** **21TCLC_Nhật 1**

**Đà Nẵng, 06/2025**

# 概要

**課題名**: JobSwipe - AIを活用した求人マッチングシステムの開発
**実施する学生**: Huỳnh Hải Đăng (フイン・ハイ・ダン)
**学生証番号**: 102210341 **クラス**: 21TCLC_Nhật1

本プロジェクトは、今日の競争が激化する労働市場において、求職者と企業双方が効率的かつ的確にマッチングできるソリューションを提供することを目指しています。このシステムは、直感的なスワイプ機能を採用し、AI技術を活用した高度なレコメンデーションを通じて、最適なマッチングを実現します。特に、双方向の推薦アルゴリズムとユーザーの相互作用に基づくパーソナライズドマトリックスを活用することで、マッチングの結果を最適化します。これにより、求職者は興味のある求人に簡単にアクセスでき、企業は適切な人材を迅速に見つけ出すことが可能となります。

技術面では、以下の先進的な技術スタックを採用しました：

- **フロントエンド**: Flutterを用いて、iOSとAndroid両対応のモバイルアプリを開発。スワイプやチャットなど、直感的で使いやすいインターフェースを提供。
- **バックエンド**: Spring Boot（Java）によるRESTful APIで、データ管理、認証、リアルタイム通信を効率的に処理。
- **データベース**: PostgreSQLで堅牢なデータ管理を実現し、Redisをキャッシュとして活用してパフォーマンスを最適化。
- **AIレコメンデーション**: ベクトル埋め込みを用いた双方向のレコメンドアルゴリズムが、ユーザーの行動や好みを学習し、パーソナライズされたマッチングを提供します。
- **クラウドインフラ**: AWS（EC2、S3）を活用し、スケーラビリティと信頼性を確保。

本プロジェクトは、以下の4章で構成されています：

- **第1章: 理論的背景と使用技術**
  RESTful API、JWT、Flutter、Spring Boot、PostgreSQL、AI技術（ベクトルエンベディング、類似度計算）など、システムの基盤技術を解説します。
- **第2章: システムの分析と設計**
  ユースケース図、シーケンス図、データベース設計を通じて、機能要件とシステム設計を詳細に説明します。

- **第3章: システムの開発**
  開発環境の構築、AIモデルの実装、フロントエンドとバックエンドの統合など、具体的な開発プロセスを紹介します。
- **第4章: システムのデプロイと結果**
  AWS上でのデプロイ、実際の画面例を交えたデモンストレーションを行い、成果を総括します。

本プロジェクトは、ユーザーが自身のスキルや適性を最大限に活かし、理想的な仕事あるいは人材との出会いを実現できるよう支援することを目的としています。双方向の推薦アルゴリズムおよびユーザーの行動データに基づくパーソナライズドマトリックスを活用することで、求人と求職者間のマッチング精度を向上させ、従来の採用プロセスに比べて、より効率的かつ効果的な人材マッチングを可能とする新しいアプローチを提案します。本システムは、AI技術と直感的なユーザーインターフェースを融合させた先進的な求人マッチングプラットフォームの一例として、実用性と発展性の両面において有望な成果を示しています。

# TÓM TẮT

**Tên đề tài**: JobSwipe - Ứng dụng AI trong hệ thống tìm việc
**Sinh viên thực hiện**: Huỳnh Hải Đăng
**Số thẻ SV**: 102210341
**Lớp**: 21TCLC_NHAT 1

Dự án JobSwipe là ứng dụng di động kết nối nhà tuyển dụng và người tìm việc qua cơ chế vuốt kiểu ứng dụng Tinder giúp đơn giản hóa quy trình tuyển dụng và tăng trải nghiệm người dùng. Giao diện được xây dựng bằng Flutter, backend dùng Spring Boot, áp dụng kiến trúc RESTful API và triển khai trên AWS để đảm bảo tính ổn định, bảo mật và dễ mở rộng.

Người dùng có thể tạo, chỉnh sửa hoặc quét CV từ file PDF, hệ thống sẽ tự động trích xuất thông tin như kỹ năng, học vấn, kinh nghiệm để điền vào hồ sơ. Khi hai bên cùng quan tâm, tính năng chat trực tiếp được mở, rút ngắn quá trình trao đổi thông tin tuyển dụng.

Điểm nổi bật của hệ thống là mô hình đề xuất thông minh. Thay vì dùng thuật toán truyền thống, JobSwipe ứng dụng vector embedding và lưu trữ trên ChromaDB. Mỗi người dùng và công ty đều có vector riêng, được điều chỉnh liên tục thông qua ma trận cá nhân hóa dựa trên feedback (swipe phải/trái). Cơ chế này giúp hệ thống hiểu rõ hơn về sở thích thực tế của người dùng và từ đó đề xuất các công việc phù hợp nhất. Đồng thời, phía công ty cũng có thể nhận được đề xuất về các ứng viên tiềm năng, giúp tăng hiệu quả kết nối hai chiều trong quá trình tuyển dụng.

JobSwipe là một dự án độc lập hoàn chỉnh từ thiết kế, lập trình đến triển khai, tích hợp AI recommendation và vận hành Cloud. Đây không chỉ là sản phẩm thử nghiệm mà còn là nền tảng vững chắc cho các hệ thống tuyển dụng thông minh trong tương lai.

**ĐẠI HỌC ĐÀ NẴNG**
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**
**KHOA CÔNG NGHỆ THÔNG TIN**

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

# NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Huỳnh Hải Đăng Số thẻ sinh viên: 102210341

Lớp: 21TCLC_NHAT 1 Khoa: Công Nghệ Thông Tin

Ngành: Công nghệ thông tin (Chất lượng cao - tiếng Nhật)

1. *Tên đề tài đồ án:* Ứng dụng AI trong hệ thống tìm việc.
2. *Đề tài thuộc diện:* ☐ *Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*
3. *Các số liệu và dữ liệu ban đầu:*
4. *Nội dung các phần thuyết minh và tính toán:*

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT
CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG
CHƯƠNG 3: PHÁT TRIỂN HỆ THỐNG.
CHƯƠNG 4: TRIỂN KHAI HỆ THỐNG VÀ KẾT QUẢ

5. *Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ):*
   - Sơ đồ quan hệ
   - Biểu đồ tuần tự
   - Sơ đồ use-case
6. *Họ tên người hướng dẫn: ThS. Mai Văn Hà*
7. *Ngày giao nhiệm vụ đồ án: 06/2025*
8. *Ngày hoàn thành đồ án: 06/2025*

*Đà Nẵng, ngày ... tháng 06 năm 2025*

**Trưởng Bộ môn** **Người hướng dẫn**

**PGS.TS, Nguyễn Tấn Khôi** **ThS. Mai Văn Hà**

# LỜI NÓI ĐẦU

Đề tài *"JobSwipe – Ứng dụng AI trong hệ thống tìm việc"* là kết quả của quá trình học tập, rèn luyện và tích lũy kinh nghiệm trong suốt thời gian theo học tại Khoa Công nghệ Thông tin – Trường Đại học Bách khoa – Đại học Đà Nẵng. Đây cũng là đề tài đồ án tốt nghiệp, mang ý nghĩa quan trọng trong việc đánh dấu bước trưởng thành về cả kiến thức chuyên môn lẫn tư duy hệ thống.

Xin trân trọng gửi lời cảm ơn đến Ban Giám hiệu Nhà trường, quý thầy cô trong Khoa Công nghệ Thông tin đã tạo điều kiện học tập, trang bị nền tảng kiến thức vững chắc và truyền cảm hứng suốt quá trình học đại học.

Trong quá trình thực hiện đề tài để có thể hoàn thành tốt, em xin bày tỏ lòng biết ơn sâu sắc đến thầy Mai Văn Hà, người đã tận tình hướng dẫn, luôn theo sát tiến độ và sẵn sàng đưa ra những góp ý quý báu trong suốt quá trình thực hiện đề tài. Sự hỗ trợ của thầy là yếu tố quan trọng giúp em vượt qua những khó khăn, hoàn thiện sản phẩm một cách hiệu quả và thực tế hơn.

Mặc dù đã cố gắng nghiên cứu và phát triển đề tài với tinh thần trách nhiệm cao nhất, nhưng do hạn chế về thời gian, kinh nghiệm và kiến thức, chắc chắn đề tài không tránh khỏi những thiếu sót nhất định. Kính mong nhận được những ý kiến đóng góp từ quý thầy cô và các bạn sinh viên để có thể hoàn thiện hơn trong tương lai.

Cuối cùng em xin kính chúc quý thầy cô mạnh khỏe, thành đạt, bình an và hạnh phúc.

# LỜI CAM ĐOAN

**Em xin cam đoan:**

1. Những nội dung trong báo cáo này là do em tự thực hiện dưới sự hướng dẫn trực tiếp của giảng viên hướng dẫn ThS. Mai Văn Hà.
2. Các số liệu sử dụng trong báo cáo có nguồn gốc rõ ràng. Các kết quả nghiên cứu trong báo cáo do em tự tìm hiểu, phân tích một cách trung thực, khách quan, cam đoan về liêm chính học thuật.
3. Nếu có những sao chép không hợp lệ vi phạm quy chế đào tạo, vi phạm liêm chính học thuật, em xin chịu hoàn toàn trách nhiệm.

*Đà Nẵng, ngày…tháng 06 năm 2025*

**Họ và tên sinh viên**
(Ký và ghi rõ họ tên)

Huỳnh Hải Đăng

# DANH SÁCH BẢNG BIỂU

# DANH SÁCH HÌNH ẢNH

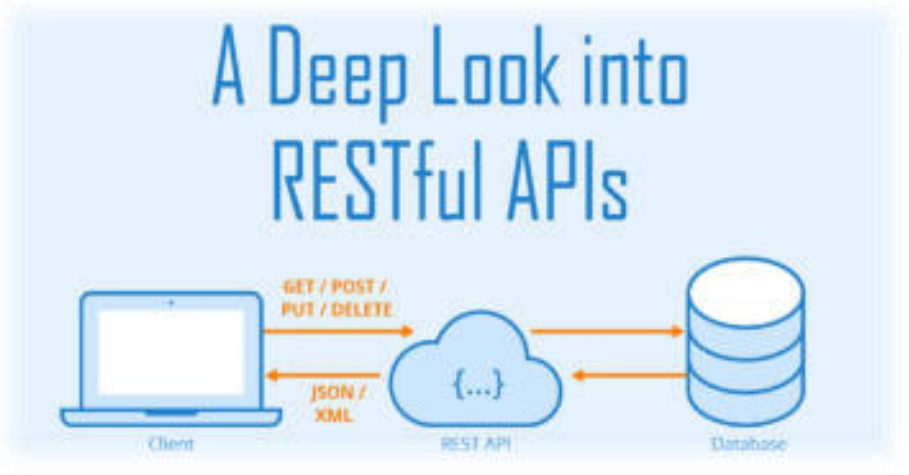
# DANH SÁCH CÁC KÍ HIỆU, CHỮ VIẾT TẮT

| Số thứ tự | Từ viết tắt | Mô tả |
|---|---|---|
| 1 | ACID | Atomicity, Consistency, Isolation, Durability |
| 2 | AI | Artificial Intelligence |
| 3 | AOT | Ahead-of-Time |
| 4 | API | Application Programming Interface |
| 5 | AWS | Amazon Web Services |
| 6 | BRIN | Block Range Index |
| 7 | CDN | Content Delivery Network |
| 8 | CLI | Command Line Interface |
| 9 | CSDL | Cơ sở dữ liệu (Database) |
| 10 | DR | Disaster Recovery |
| 11 | EC2 | Elastic Compute Cloud |
| 12 | GIN | Generalized Inverted Index |
| 13 | GiST | Generalized Search Tree |
| 14 | HPC | High-Performance Computing |
| 15 | HTTP | Hypertext Transfer Protocol |
| 16 | JIT | Just-in-Time |
| 17 | JSON | JavaScript Object Notation |
| 18 | JVM | Java Virtual Machine |
| 19 | JWT | JSON Web Token |
| 20 | MVC | Model-View-Controller |
| 21 | MVVM | Model-View-ViewModel |
| 22 | RBAC | Role-Based Access Control |
| 23 | REST | Representational State Transfer |
| 24 | S3 | Simple Storage Service |
| 25 | UI | User Interface |
| 26 | URL | Uniform Resource Locator |
| 27 | UX | User Experience |
| 28 | VPC | Virtual Private Cloud |
| 29 | WAL | Write-Ahead Logging |

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT VÀ CÔNG CỤ SỬ DỤNG

## 1.1 Kiến thức cơ bản

### *1.1.1 RESTful API*




Hình 1.1: Tổng quan RESTful API

RESTful API (Representational State Transfer Application Programming Interface) là một kiến trúc phổ biến dùng để xây dựng các dịch vụ web. Nó hoạt động dựa trên giao thức HTTP, cung cấp giao diện nhất quán để trao đổi dữ liệu giữa client (ứng dụng web, di động) và server.

Trong dự án này, RESTful API đóng vai trò cầu nối giữa các thành phần frontend và backend. Nó định nghĩa các endpoint rõ ràng cho phép client gửi yêu cầu và nhận phản hồi từ server theo cách có cấu trúc, dễ hiểu.

RESTful API sử dụng các phương thức HTTP tiêu chuẩn:

- GET: Lấy thông tin tài nguyên
- POST: Tạo mới tài nguyên
- PUT: Cập nhật toàn bộ tài nguyên
- PATCH: Cập nhật một phần tài nguyên
- DELETE: Xóa tài nguyên

Các tài nguyên được định danh bằng URI, ví dụ:

- GET /api/v1/users: lấy danh sách người dùng
- GET /api/v1/users/123: lấy thông tin người dùng có ID 123
- POST /api/v1/users: tạo người dùng mới
- PUT /api/v1/users/123: cập nhật toàn bộ thông tin người dùng
- PATCH /api/v1/users/123: cập nhật một phần (ví dụ: email)
- DELETE /api/v1/users/123: xóa người dùng

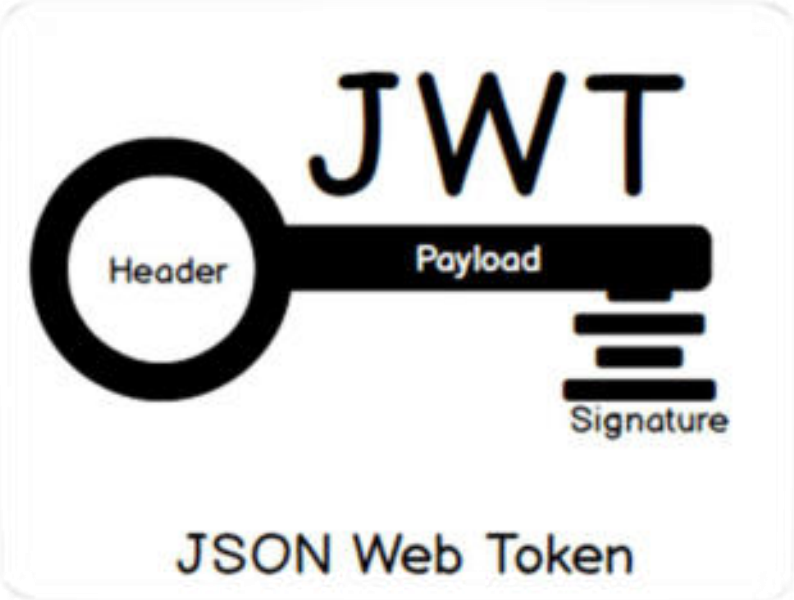
Dữ liệu thường được trao đổi dưới dạng JSON do tính đơn giản, dễ phân tích và được hỗ trợ rộng rãi.

Lợi ích của RESTful API:

- Đơn giản, dễ sử dụng: dựa trên HTTP và định dạng phổ biến như JSON.
- Mở rộng tốt: dễ bổ sung tính năng mới.
- Tính độc lập cao: client và server phát triển tách biệt.
- Bảo mật tốt: hỗ trợ các cơ chế xác thực như JWT.

Việc áp dụng RESTful API giúp hệ thống linh hoạt, dễ bảo trì, đồng thời đảm bảo khả năng tích hợp và mở rộng trong tương lai.

### *1.1.2 JSON Web Token (JWT)*




Hình 1.2: Tổng quan JSON Web Token

JSON Web Token (JWT) là tiêu chuẩn mở (RFC 7519) dùng để truyền thông tin an toàn giữa client và server dưới dạng JSON. JWT gồm ba phần, phân tách bằng dấu chấm (.):

Header: Xác định loại token và thuật toán ký.

Ví dụ: {
"alg": "HS256",
"typ": "JWT"
}

Payload: Phần payload chứa các claims (thông tin về đối tượng). Các claims có thể là registered, public, hoặc private.

Ví dụ: {
"sub": "77f4ea91-ec36-43bb-a8e2-0e00dc7893bb ",

```
  "name": "ptvinh203",
  "iat": 1516239022
}
```

Signature: Ký phần header và payload bằng secret key (HMAC) hoặc cặp khóa công khai/riêng tư (RSA/ECDSA) để đảm bảo tính toàn vẹn và xác thực.

Quy trình sử dụng trong dự án:

- Sau khi đăng nhập, server tạo JWT chứa ID và vai trò của người dùng, rồi trả về client.
- Client lưu token (local storage/cookie) và gửi kèm theo mỗi yêu cầu.
- Server kiểm tra chữ ký và các claim để xác thực và phân quyền trước khi xử lý.

Lợi ích:

- Bảo mật: Chữ ký số ngăn giả mạo và đảm bảo dữ liệu không bị sửa đổi.
- Stateless: Server không cần lưu trữ phiên, giảm tải và dễ mở rộng.
- Di động: Token có thể truyền qua header HTTP hoặc nhiều cơ chế khác, tương thích đa nền tảng.

Việc áp dụng JWT giúp hệ thống an toàn hơn, linh hoạt khi mở rộng và giảm phụ thuộc vào quản lý phiên lưu trữ phía server.

### 1.1.3 Java


Hình 1.3: Logo của ngôn ngữ lập trình Java

Java là ngôn ngữ hướng đối tượng mạnh mẽ với triết lý "Write Once, Run Anywhere" nhờ Java Virtual Machine (JVM), được ưa chuộng trong phát triển ứng dụng doanh nghiệp quy mô lớn.

Hệ sinh thái Spring phong phú hỗ trợ phát triển nhanh chóng và chuyên nghiệp, cung cấp:

- Tự động cấu hình thông minh: Thiết lập sẵn các thành phần theo quy ước, đồng thời cho phép tùy chỉnh linh hoạt.
- Quản lý phụ thuộc hiệu quả: Tự động giải quyết và tối ưu phiên bản thư viện, giảm xung đột.
- Xây dựng RESTful API: Hỗ trợ đầy đủ HTTP, xử lý request/response linh hoạt, tích hợp validation và JSON serialization.
- Hệ sinh thái phong phú: Kết hợp liền mạch với Spring Data, Spring Security, Spring Cloud cho ứng dụng phân tán.
- Bảo mật đa lớp: Áp dụng Spring Security với xác thực, phân quyền, mã hóa và chống tấn công web.

Ưu điểm chính của Java và Spring Boot trong dự án:

- Tính an toàn: Hệ thống kiểu tĩnh phát hiện lỗi sớm.
- Tính ổn định và mở rộng: Mô hình kiến trúc rõ ràng, dễ bảo trì.
- Hiệu suất cao: JVM tối ưu runtime; Spring Boot tối giản cấu hình.
- Phát triển nhanh: Các công cụ và thư viện hỗ trợ đắc lực, rút ngắn thời gian đưa sản phẩm vào vận hành.

### 1.1.4 Java Spring Boot


Hình 1.4: Logo của framwork Java Spring Boot

**Java Spring Boot** là framework mã nguồn mở xây dựng trên Spring, giúp đơn giản hóa phát triển ứng dụng Java, đặc biệt là web, nhờ auto-configuration và tích hợp sẵn nhiều tính năng:

- Auto-configuration: Tự cấu hình các thành phần theo thư viện có trong classpath, giảm mã boilerplate và cấu hình XML.
- Embedded Servers: Nhúng Tomcat/Jetty/Undertow, triển khai dễ dàng mà không cần cài máy chủ riêng.
- Spring Data: Hỗ trợ truy vấn và thao tác với cơ sở dữ liệu SQL (MySQL, PostgreSQL…) và NoSQL (MongoDB, Cassandra…).
- Spring Security: Bảo mật ứng dụng với xác thực, phân quyền và chống tấn công.
- Actuator: Cung cấp các endpoint để giám sát và quản lý ứng dụng.
- CLI: Tạo và chạy nhanh ứng dụng qua dòng lệnh mà không cần mã mẫu.

**Ứng dụng trong dự án:**

- Xây RESTful API giao tiếp với client.
- Kết nối PostgreSQL lưu trữ dữ liệu, Redis xử lý cache và session.
- Triển khai xác thực, ủy quyền qua JWT.
- Giám sát hệ thống bằng Actuator.

Nhờ Spring Boot, việc phát triển, cấu hình và triển khai web service trở nên nhanh chóng, ổn định và dễ mở rộng, giúp tập trung vào logic nghiệp vụ cốt lõi.

### *1.1.5 Tổng quan về ngôn ngữ Dart và Flutter Framwork*

#### *1.1.5.1 Dart*


Hình 1.5: Logo của ngôn ngữ lập trình Dart

Dart là một ngôn ngữ lập trình đa năng, hướng đối tượng được phát triển bởi Google. Ban đầu được thiết kế để thay thế JavaScript như ngôn ngữ lập trình web chính, Dart đã phát triển thành một ngôn ngữ linh hoạt được sử dụng chủ yếu trong phát triển ứng dụng di động thông qua framework Flutter.

**Các đặc điểm nổi bật của Dart:**

- Cú pháp rõ ràng và dễ học: Dart có cú pháp tương tự với C++ và Java, giúp các lập trình viên dễ dàng tiếp cận. Ngôn ngữ này cung cấp các tính năng hiện đại như null safety, async/await và collection literals.
- Hỗ trợ AOT và JIT compilation: Dart có thể được biên dịch trước (AOT) để tạo mã máy native hiệu suất cao, hoặc biên dịch ngay lúc chạy (JIT) để hỗ trợ hot reload trong quá trình phát triển.
- Garbage Collection tự động: Dart tự động quản lý bộ nhớ thông qua garbage collection, giúp lập trình viên tập trung vào logic ứng dụng mà không phải lo lắng về việc quản lý bộ nhớ thủ công.
- Type System mạnh mẽ: Hệ thống kiểu dữ liệu của Dart vừa linh hoạt vừa an toàn, hỗ trợ cả static typing và type inference, giúp phát hiện lỗi sớm trong quá trình phát triển.
- Hỗ trợ lập trình bất đồng bộ: Dart cung cấp các công cụ mạnh mẽ để xử lý các tác vụ bất đồng bộ thông qua Future và Stream, kết hợp với async/await syntax để code dễ đọc và bảo trì.
- Ecosystem phong phú: Pub.dev - kho package chính thức của Dart - cung cấp hàng nghìn package và plugin có sẵn, giúp tăng tốc quá trình phát triển.

**Ứng dụng chính của Dart:**

- Phát triển ứng dụng di động: Kết hợp với Flutter framework, Dart là công cụ chính để xây dựng ứng dụng di động đa nền tảng với hiệu suất cao và giao diện người dùng đẹp.
- Phát triển web: Dart có thể được biên dịch sang JavaScript để chạy trên trình duyệt, cho phép xây dựng ứng dụng web động với AngularDart hoặc các framework khác.
- Ứng dụng máy chủ: Dart có thể được sử dụng để viết các ứng dụng máy chủ và microservices thông qua frameworks như Aqueduct và Angel.

***a) Flutter***


Hình 1.6: Logo của Framework Flutter

Flutter là framework phát triển ứng dụng di động đa nền tảng do Google phát triển. Framework này cho phép lập trình viên tạo ra các ứng dụng chất lượng cao cho iOS và Android từ một codebase duy nhất sử dụng ngôn ngữ Dart.

**Các đặc điểm nổi bật của Flutter:**

- Hot Reload: Cho phép xem thay đổi ngay lập tức trong quá trình phát triển mà không cần build lại ứng dụng, giúp tăng tốc quá trình phát triển và debugging.
- Widget tùy chỉnh: Flutter sử dụng hệ thống widget phong phú và linh hoạt, cho phép tạo giao diện người dùng phức tạp với hiệu suất cao và khả năng tùy chỉnh không giới hạn.
- Hiệu năng cao: Ứng dụng Flutter được biên dịch thành mã máy native, cho phép chạy mượt mà với tốc độ 60fps trên cả hai nền tảng iOS và Android.
- Material Design và Cupertino: Framework cung cấp sẵn các widget theo thiết kế Material (Android) và Cupertino (iOS), giúp tạo giao diện phù hợp với từng nền tảng.

**Kiến trúc của Flutter:**

- Framework Layer: Chứa các widget và công cụ cấp cao để xây dựng ứng dụng.
- Engine Layer: Xử lý rendering, animations, và tương tác với platform-specific APIs.
- Platform Layer: Tương tác trực tiếp với hệ điều hành và các tính năng native.

**Ưu điểm của Flutter trong phát triển ứng dụng:**

- Giảm thời gian và chi phí phát triển: Một codebase duy nhất cho nhiều nền tảng.
- Cộng đồng lớn mạnh: Nhiều package và plugin có sẵn trên pub.dev.
- Tương thích ngược tốt: Các ứng dụng Flutter có thể chạy trên các phiên bản iOS và Android cũ.

### 1.1.6 Tổng quan về Python


Hình 1.7: Logo của ngôn ngữ lập trình Python

Python là ngôn ngữ đa năng, mã nguồn mở, phát triển từ 1991, sử dụng rộng rãi trong khoa học dữ liệu, AI và phát triển web.

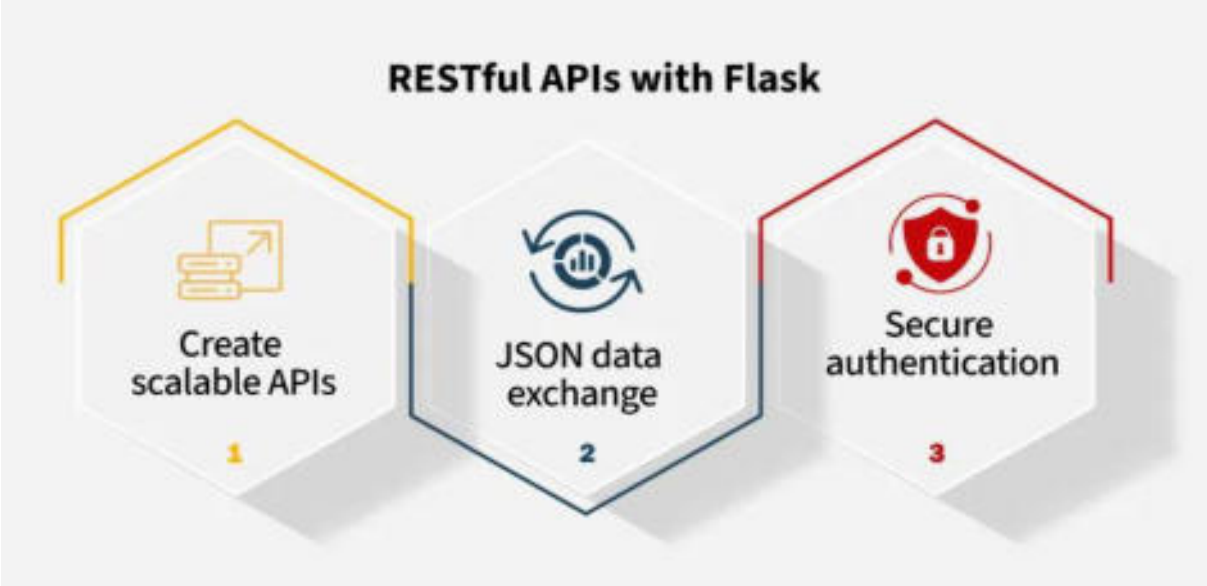
Đặc trưng chính:

- Cú pháp đơn giản, dễ đọc: Giúp viết và bảo trì mã hiệu quả.
- Đa nền tảng: Chạy mượt trên Windows, Linux, macOS…
- Thư viện phong phú: NumPy, Pandas, Matplotlib, Scikit-learn… hỗ trợ xử lý và phân tích dữ liệu.
- Tương tác đa ngôn ngữ: Kết hợp với C/C++/Java để tái sử dụng mã và thư viện.
- Đa mô hình lập trình: Hướng đối tượng, hàm, thủ tục.
- Hỗ trợ AI/ML: TensorFlow, Keras, PyTorch…

Nhờ tính linh hoạt và hệ sinh thái mạnh, Python là lựa chọn hàng đầu cho nghiên cứu dữ liệu, trí tuệ nhân tạo và phát triển ứng dụng.

### *1.1.7 Tổng quan về Flask framework*

Flask là loại framework web phổ biến được viết bằng trình lập ngôn ngữ Python. Công nghệ thường được sử dụng để xây dựng trang web từ những ứng dụng đơn giản đến những hệ thống phức tạp hơn.




Hình 1.8: Tổng quan về Flask framework

Flask Framework sở hữu một số tính năng quan trọng mà nhà phát triển thường sử dụng để xây dựng hiệu ứng web. Dưới đây là một số tính năng chính của Flask:

- Nhẹ và dễ sử dụng: Công nghệ có cấu trúc nhẹ nhàng và mã nguồn dễ đọc, giúp người phát triển dễ dàng tiếp cận và tùy chỉnh theo nhu cầu cụ thể của họ.
- Định tuyến linh hoạt: Flask cung cấp cơ chế hoạt động định tuyến, cho phép người phát triển xác định các mẫu URL và phân bổ chúng cho các hàm xử lý tương ứng. Điều này giúp quản lý và xử lý yêu cầu HTTP một cách hiệu quả.
- Công cụ mẫu: Flask tích hợp Jinja2, đây là một loại trình biên dịch mẫu mạnh mẽ cho phép tạo ra các giao diện người dùng.
- Được mở rộng rộng rãi: Mặc dù mang đặc điểm rút gọn nhưng Flask vẫn có khả năng mở rộng mạnh mẽ thông qua việc sử dụng các tiện ích và thư viện của cộng đồng. Người dùng có thể phân tích các tính năng như xác thực, đăng nhập, điều hướng, cơ sở dữ liệu tương tác và nhiều tính năng khác.
- Máy chủ phát triển tích hợp: Flask cung cấp máy chủ phát triển hợp đồng, giúp người phát triển dễ dàng kiểm tra và phát triển ứng dụng mà không cần cấu hình bổ sung.
- Gửi yêu cầu RESTful: Flask hỗ trợ xây dựng API và các ứng dụng RESTful theo cách hoạt động và hiệu quả.

### *1.1.8 Tổng quan về hệ quản trị cơ sở dữ liệu MySQL*


Hình 1.9: Logo của hệ quản trị cơ sở dữ liệu PostgreSQL

PostgreSQL là hệ quản trị CSDL quan hệ mã nguồn mở, nổi bật về độ ổn định, tính năng và khả năng mở rộng.

**Đặc điểm chính:**

- ACID: Bảo đảm tính nguyên tử, nhất quán, cô lập và bền vững.
- Kiểu dữ liệu đa dạng: Hỗ trợ JSON, XML, hstore, địa lý…
- Mở rộng: Table inheritance, partitioning, custom data types.
- Hiệu suất cao: Parallel query, index-only scans, query planner tối ưu.

**Tính năng nâng cao:**

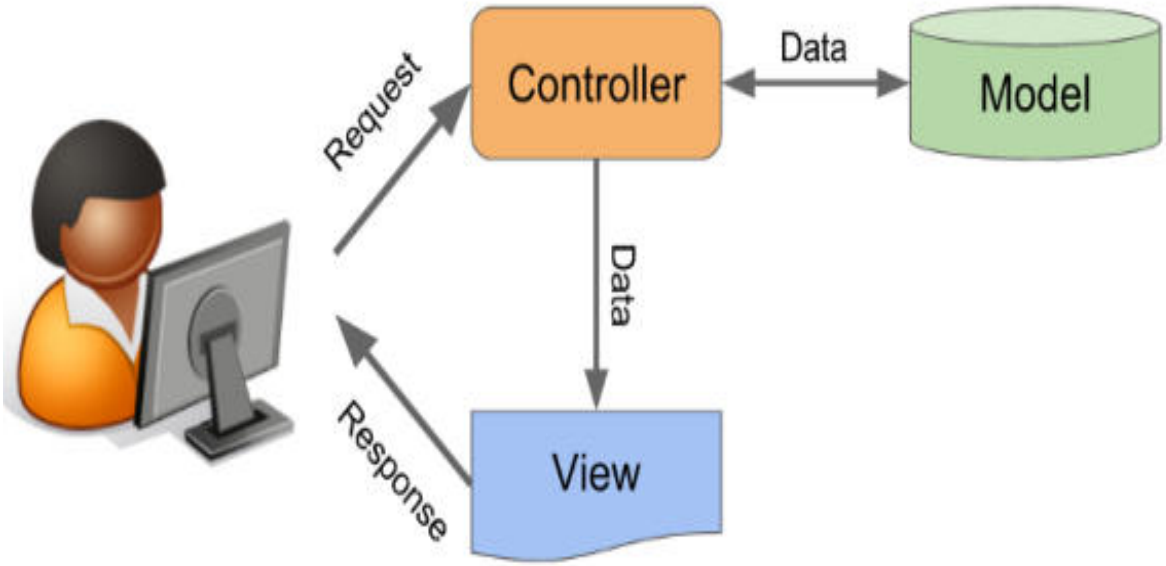
- Stored Procedures/Functions: PL/pgSQL, PL/Python, PL/Perl…
- Advanced Indexing: B-tree, GIN, GiST, BRIN…
- Materialized Views: Lưu kết quả truy vấn để tăng tốc.
- Full-text Search: Tìm kiếm văn bản đa ngôn ngữ.

**Bảo mật và tin cậy:**

- RBAC: Phân quyền chi tiết.
- SSL/TLS: Kết nối mã hóa.
- WAL: Write-Ahead Logging đảm bảo phục hồi.

PostgreSQL phù hợp cho ứng dụng doanh nghiệp yêu cầu cao về độ tin cậy, tính năng và khả năng mở rộng.

### *1.1.9 Tổng quan về Mô hình MVC*




Hình 1.10: Mô hình MVC

**MVC (Model-View-Controller)** chia ứng dụng thành ba thành phần riêng biệt:

- **Model**: Quản lý dữ liệu và logic nghiệp vụ. Ví dụ: đối tượng "Công việc" với phương thức thêm, sửa, xóa.
- **View**: Hiển thị giao diện và nhận tương tác người dùng, lấy dữ liệu từ Model.
- **Controller**: Nhận yêu cầu từ View, điều phối với Model và cập nhật View.

**Luồng hoạt động:**

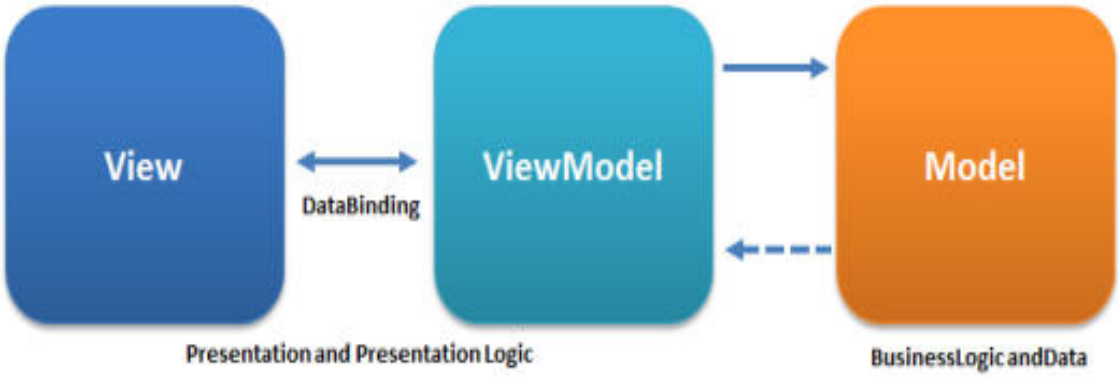
1. Người dùng tương tác View
2. View gửi yêu cầu đến Controller
3. Controller xử lý và gọi Model
4. Model cập nhật dữ liệu và thông báo View
5. View render giao diện mới

**Lợi ích:**

- Tách biệt trách nhiệm: Dễ bảo trì, mở rộng
- Tái sử dụng mã: Các thành phần độc lập dùng cho nhiều nơi
- Dễ kiểm thử: Kiểm thử riêng lẻ từng thành phần

MVC giúp phát triển ứng dụng web có cấu trúc rõ ràng, dễ quản lý và kiểm thử

### *1.1.10 Tổng quan về Mô hình MVVM*




Hình 1.11: Mô hình MVVM

**MVVM (Model-View-ViewModel)** tách biệt rõ giao diện và logic nghiệp vụ, phù hợp với cơ chế data binding.

- **Model**: Quản lý dữ liệu và logic nghiệp vụ (ví dụ: đối tượng Người dùng và phương thức truy xuất CSDL).
- **View**: Giao diện người dùng, binding với ViewModel để hiển thị và nhận tương tác.
- **ViewModel**: Lớp trung gian chuyển đổi dữ liệu từ Model thành dạng View cần, chứa property và command, xử lý validation, quản lý trạng thái.

**Luồng hoạt động**:

1. View bind đến các property/command của ViewModel.
2. Người dùng thao tác, gọi command trong ViewModel.
3. ViewModel tương tác Model để lấy hoặc cập nhật dữ liệu.
4. Model thay đổi dữ liệu và thông báo ViewModel.
5. ViewModel cập nhật property; View tự động render nhờ binding.

**Lợi ích**:

- **Data binding hai chiều**: Đồng bộ dữ liệu tự động.
- **Separation of Concerns**: Giao diện và logic tách biệt.
- **Dễ kiểm thử**: ViewModel test độc lập không cần UI.
- **Tái sử dụng code**: ViewModel dùng chung cho nhiều View.

MVVM giúp xây dựng ứng dụng dễ mở rộng, bảo trì và test, thường dùng trong WPF, Xamarin và các framework JavaScript như Vue.js, Angular.

## 1.2 Tổng quan về Trí tuệ nhân tạo

### *1.2.1 Machine learning*


Hình 1.12: Học máy

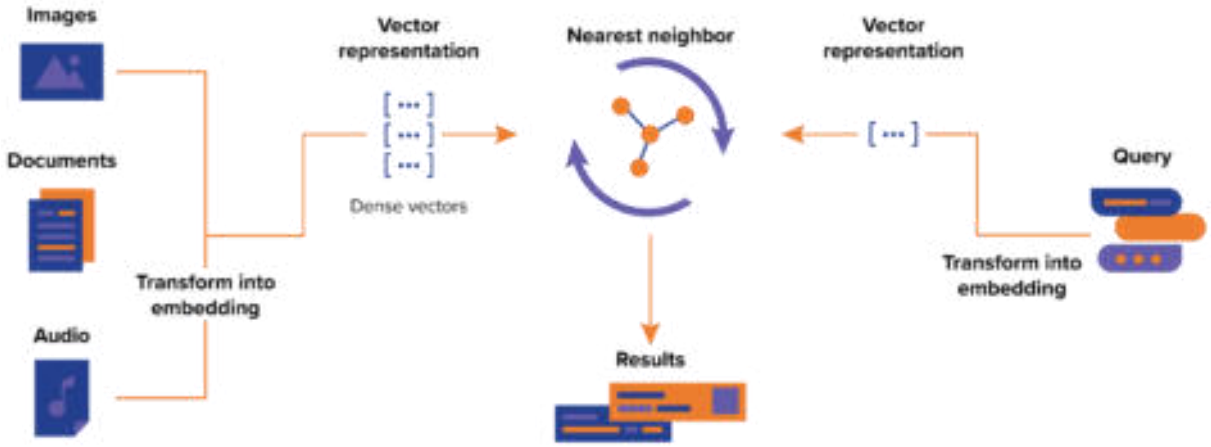
Học máy (machine learning) là một phần của trí tuệ nhân tạo, bao gồm việc huấn luyện các thuật toán để học các mẫu trong dữ liệu và đưa ra dự đoán hoặc quyết định. Nó bao gồm nhiều kỹ thuật khác nhau, bao gồm học có giám sát (supervised learning), học không giám sát (unsupervised learning ) và học củng cố (reinforcement learning). Học máy được sử dụng trong nhiều ứng dụng khác nhau, từ nhận dạng hình ảnh và giọng nói đến phát hiện gian lận và xe tự lái và đóng một vai trò rất lớn trong việc cải thiện đời sống con người. Để có thể có một mô hình học máy tốt, nó phụ thuộc rất nhiều vào các tập dữ liệu lớn và các nguồn tài nguyên tính toán mạnh, bao gồm GPU và các nền tảng tính toán đám mây. Khi lĩnh vực này tiếp tục phát triển, học máy sẽ có vai trò ngày càng quan trọng trong nhiều lĩnh vực và ngành công nghiệp. (Machine Learning Tutorial)

### *1.2.2 Deep learning*

Học sâu (deep learning) là một phần của học máy sử dụng mạng neural với nhiều lớp để học các mẫu phức tạp trong dữ liệu. Nó đã cách mạng hóa các lĩnh vực như thị giác máy tính, xử lý ngôn ngữ tự nhiên và nhận dạng giọng nói. Các mô hình deep learning được huấn luyện trên các bộ dữ liệu lớn và yêu cầu các tài nguyên tính toán mạnh, bao gồm GPU và tính toán phân tán. Các kỹ thuật quan trọng trong deep learning bao gồm mạng neural tích chập (convolutional neural networks), mạng neural tái phát (recurrent neural networks)và mạng sinh đối địch (generative adversarial networks). Khi lĩnh vực

này tiếp tục phát triển, deep learning có khả năng đẩy mạnh sự tiến bộ trong trí tuệ nhân tạo và học máy. (Introduction to Deep Learning)

### *1.2.3 Vector embedding*

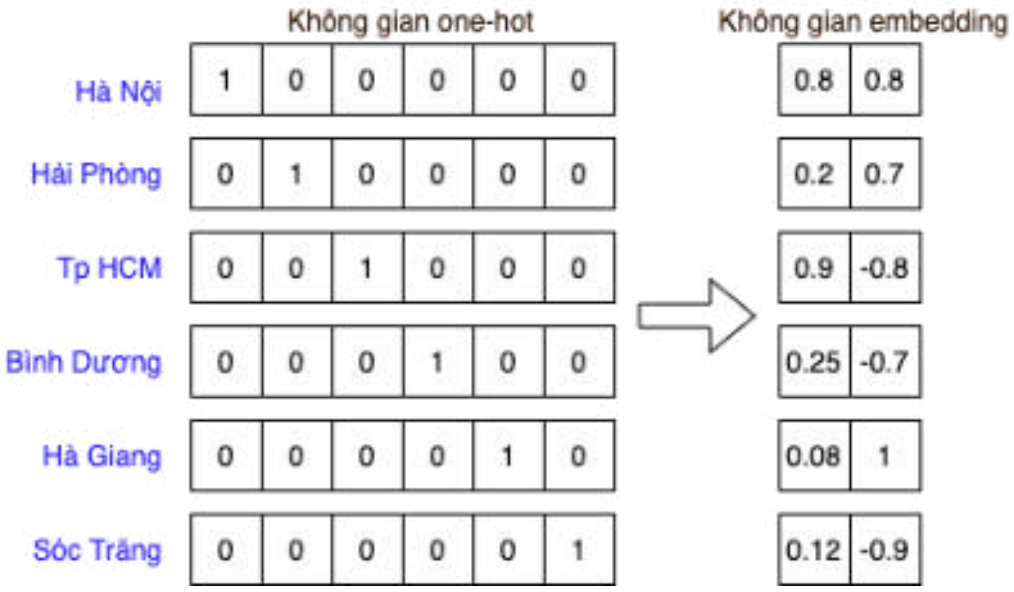



Hình 1.13: Nguyên lí Vector Embedding

Embedding là một kỹ thuật đưa một vector có số chiều lớn, thường ở dạng thưa, về một vector có số chiều nhỏ, thường ở dạng dày đặc. Phương pháp này đặc biệt hữu ích với những đặc trưng hạng mục có số phần tử lớn ở đó phương pháp chủ yếu để biểu diễn mỗi giá trị thường là một vector dạng one-hot. Một cách lý tưởng, các giá trị có ý nghĩa tương tự nhau nằm gần nhau trong không gian embedding.

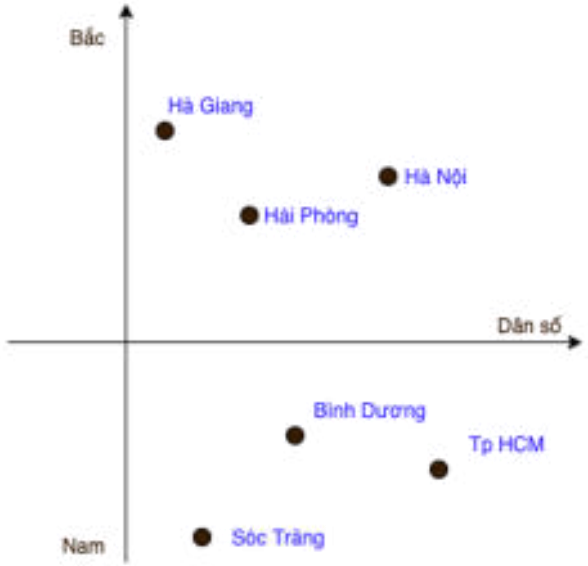
Ví dụ nổi bật nhất là biểu diễn các từ trong một bộ từ điển dưới dạng số. Khi từ điển có hàng triệu từ, biểu diễn các từ dưới dạng one-hot vector dẫn tới số chiều vô cùng lớn. Hơn nữa, các từ này sẽ có khoảng cách đều nhau tới mọi từ khác (căn bậc hai của 2), dẫn đến việc thiếu thông tin giá trị cho việc huấn luyện mô hình machine learning. Chẳng hạn, một cách biểu diễn tốt các từ tiếng Việt cần mô tả tốt sự liên quan giữa cặp từ (vua, hoàng hậu) và (chồng, vợ) vì chúng có ý nghĩa gần nhau.

Giả sử một từ điển nào đó chỉ có sáu giá trị (Hà Nội, Hải Phòng, Tp HCM, Bình Dương, Lào Cai, Sóc Trăng). Thể hiện cách biểu diễn của các giá trị này trong không gian one-hot và không gian embedding hai chiều.




Hình 1.14: Biểu diễn các giá trị hạng mục dưới dạng one-hot vector và embedding

Ở đây, các giá trị trong không gian embedding được lấy ví dụ bằng tay với chiều thứ nhất thể hiện dân số và chiều thứ hai thể hiện vĩ độ đã chuẩn hóa của mỗi giá trị. Vị trí của mỗi vector embedding trong không gian hai chiều được minh hoạ. Trong không gian này, Hà Nội, Hải Phòng và Hà Giang gần nhau về vị trí địa lý. Nếu chúng ta có một bài toán nào đó mà dân số có thể là một đặc trưng tốt, ta chỉ cần co trục tung và giãn trục hoành là có thể mang những tỉnh thành có dân số giống nhau gần với nhau hơn.

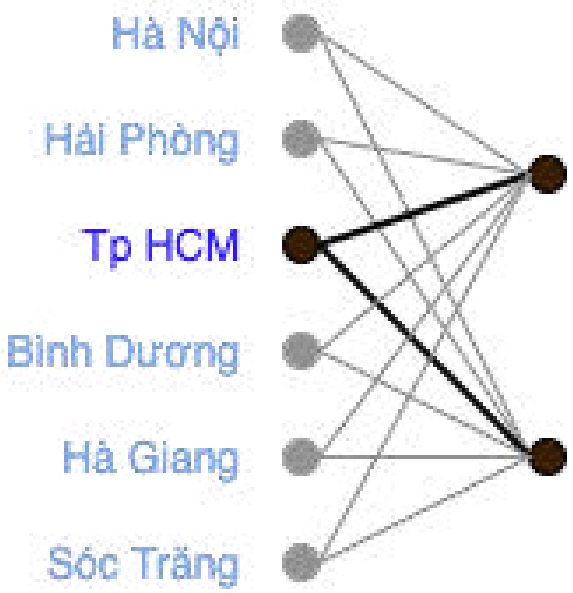



Hình 1.15: Biểu diễn các vector embedding trong không gian

Với một từ điển bất kỳ với $N$ từ $(w_1, w_2, \ldots, w_N)$. Giả sử số chiều của không gian embedding là $\boldsymbol{d}$, ta có thể biểu diễn toàn bộ các embedding cho $N$ từ này dưới dạng một ma trận $\mathbf{E} \in \mathbb{R}^{N \times k}$ với hàng thứ  là biểu diễn embedding cho từ $w_i$.

Nếu vector $\mathbf{o}_i \in \mathbb{R}^{N \times 1}$ là biểu diễn one-hot của từ $w_i$, ta có ngay $\mathbf{e} = \mathbf{o}_i^T \mathbf{E} \in \mathbb{R}^{1 \times k}$ là biểu diễn của từ đó trong không gian embedding.

Cách biểu diễn các tỉnh thành trong ví dụ trên đây chỉ là một ví dụ minh họa khi chúng ta có thể định nghĩa các trục một cách cụ thể dựa vào kiến thức nhất định đã có về dữ liệu. Cách làm này không khả thi với những dữ liệu vô cùng nhiều chiều và không có những ý nghĩa từng trục rõ ràng như trên. Việc tìm ra ma trận **E** cần thông qua một quá trình "học" dựa trên mối quan hệ vốn có của dữ liệu. Ta có thể thấy rằng ma trận **E** có thể được coi là một ma trận trọng số của một tầng tuyến tính trong một mạng neural nhân tạo như trong hình




Hình 1.16: Biểu diễn các vector embedding trong không gian

Như vậy, ma trận này cũng có thể được xây dựng bằng cách đặt nó vào một mạng neural với một hàm mất mát nào đó. Trong Tensorflow (phiên bản 2.5), tầng embedding có thể được khai báo bởi tf.keras.layers.Embedding. Trong Pytorch 1.8.1, ta có thể dùng torch.nn.Embedding.

### *1.2.4 Độ tương tự giữa hai embedding*

Quay lại với mục đích chính của việc tạo embedding là đưa các giá trị hạng mục về một không gian số sao cho embedding của những giá trị tương tự nằm gần nhau trong không gian. Vậy khoảng cách này thường được tính như thế nào.

Có ba phép đo thường được sử dụng để tính khoảng cách giữa hai embedding là khoảng cách Euclid, tích vô hướng của hai vector, và độ tương tự cosine.

**Công thức tính khoảng cách Euclid giữa hai vector embedding:**

$$[d_1(e_1, e_2) = |e_1 - e_2| = \sqrt{|e_1|^2 + |e_2|^2 - 2e_1^{\top e_2}}]$$

Khoảng cách này không âm và càng nhỏ thì hai vector embedding càng gần nhau.

Ở đây, $||\mathbf{e}|| = \sqrt{\sum_{i=1}^{d} e_i^2}$ là độ dài của vector $\mathbf{e} \in R^d$.

Để giảm sự phức tạp khi tính căn bậc hai, bình phương khoảng cách Euclid thường được sử dụng. Việc lấy bình phương không ảnh hưởng tới việc so sánh khoảng cách vì bình phương là một hàm đơn điệu.

$$[d_2(\boldsymbol{e_1}, \boldsymbol{e_2}) = |\boldsymbol{e_1} - \boldsymbol{e_2}|^2 = |\boldsymbol{e_1}|^2 + |\boldsymbol{e_2}|^2 - 2\boldsymbol{e_1}^{\top \boldsymbol{e_2}}]$$

**Công thức tính độ tương tự theo tích vô hướng (dot product) giữa hai vector embedding:**

$$[\text{similar\_dot}(e_1, e_2) = e_1^{\top e_2}]$$

Tích vô hướng giữa hai vector càng cao thì hai vector embedding càng gần nhau. Giá trị này lớn nhất nếu góc giữa hai vector nhỏ và có độ dài lớn.

**Công thức tính tương tự cosine để đo độ tương tự giữa hai vector:**

$$[\text{similar\_cosine}(e_1, e_2) = \frac{e_1^{\top e_2}}{|e_1||e_2|}]$$

Góc giữa hai vector càng nhỏ thì độ tương tự cosine càng cao. Độ tương tự cosine nhỏ nhất bằng $-1$ nếu hai vector này trái dấu nhau.

Trong ba độ đo trên đây, tích vô hướng có công thức đơn giản nhất và thường được sử dụng trong các bài toán quy mô lớn. Tương tự cosine không quan tâm tới độ lớn của hai vector mà chỉ xét tới góc giữa chúng, phép đo này phù hợp với các bài toán cần tìm sự tương đồng về ý nghĩa giữa các từ. Nếu các vector embedding có cùng độ dài, ba phép đo này có ý nghĩa như nhau.

**Tìm embedding gần nhất (Nearest Neighbor Search):**

Embedding được dùng nhiều trong bài toán tìm kiếm các điểm trong cơ sở dữ liệu (item embeddings) gần nhất với một embedding truy vấn (query embedding) nào đó.

Giả sử $v \in R^{N\times d}$ và $q \in R^{\mathbb{1}\times d}$ lần lượt là ma trận embedding của các giá trị trong cơ sở dữ liệu và vector truy vấn.

Với khoảng cách Euclid, khoảng cách giữa $q$ và một embedding $e_i$ trong $D$ được tính bởi:

$$[d_2(q, e_i) = |q|^2 + |e_i|^2 - 2q^{\top}e_i]$$

Chỉ số của embedding gần $\mathbf{q}$ được tính bởi:

$$[\arg\min_i d_2(q, e_i) = \arg\min_i (|e_i|^2 - 2q^{\top}e_i)]$$

Với độ tương tự tích vô hướng, chỉ số của embedding gần $q$ được tính bởi:

$$[\arg\max_i q^{\top}e_i = \arg\min_i (-q^{\top}e_i)]$$

Với độ tương tự cosine, chỉ số của embedding gần $q$ được tính bởi:

$$[\arg\max_i \frac{\boldsymbol{q}^{\top}\boldsymbol{e_i}}{|\boldsymbol{q}||\boldsymbol{e_i}|} = \arg\min_i \left(-\frac{\boldsymbol{q}^{\top}\boldsymbol{e_i}}{|\boldsymbol{e_i}|}\right)]$$

## 1.3 Dịch vụ AWS EC2


Hình 1.17: Logo của dịch vụ web AWS EC2

Amazon Elastic Compute Cloud (EC2) là dịch vụ cho thuê máy chủ ảo linh hoạt, mở rộng trên đám mây, cho phép tùy chỉnh CPU, bộ nhớ, lưu trữ và mạng theo nhu cầu.

Đặc trưng nổi bật

- Linh hoạt: Nhiều loại instance với cấu hình khác nhau.
- Mở rộng: Tự động tăng/giảm tài nguyên để tối ưu hiệu suất và chi phí.
- Đa nền tảng: Chạy Linux, Windows và mọi phần mềm cần thiết.
- Bảo mật: Tích hợp VPC, firewall, kiểm soát truy cập mạng.
- Tiết kiệm: Trả theo mức sử dụng, hỗ trợ On-Demand, Reserved, Spot.
- Nhất quán: Môi trường chạy ổn định, tự khôi phục khi lỗi.
- Tích hợp: Kết nối dễ dàng với S3, RDS, Lambda,…

Ứng dụng

- Web & doanh nghiệp: Triển khai trang web, hệ thống quản lý.
- Big Data: Chạy Hadoop, Spark cho phân tích dữ liệu lớn.
- Dev/Test: Môi trường phát triển, kiểm thử nhanh gọn.
- Storage & CDN: Lưu trữ, phân phối nội dung số.
- HPC: Tính toán hiệu năng cao cho mô phỏng, phân tích.
- Backup/DR: Giải pháp dự phòng và khôi phục thảm họa.

Nhờ tính linh hoạt và tích hợp sâu với AWS, EC2 giúp xây dựng, vận hành và mở rộng hệ thống hiệu quả, an toàn và tiết kiệm.

## 1.4. Dịch vụ AWS S3

Amazon Simple Storage Service (S3) là một dịch vụ lưu trữ đám mây được cung cấp bởi Amazon Web Services (AWS). Nó cho phép người dùng lưu trữ và truy xuất các đối tượng (object) dữ liệu như tệp tin, ảnh, video và các loại dữ liệu khác trong một kho lưu trữ đám mây.


Hình 1.18: Logo của dịch vụ lưu trữ AWS S3

*Những đặc trưng nổi bật của AWS S3:*

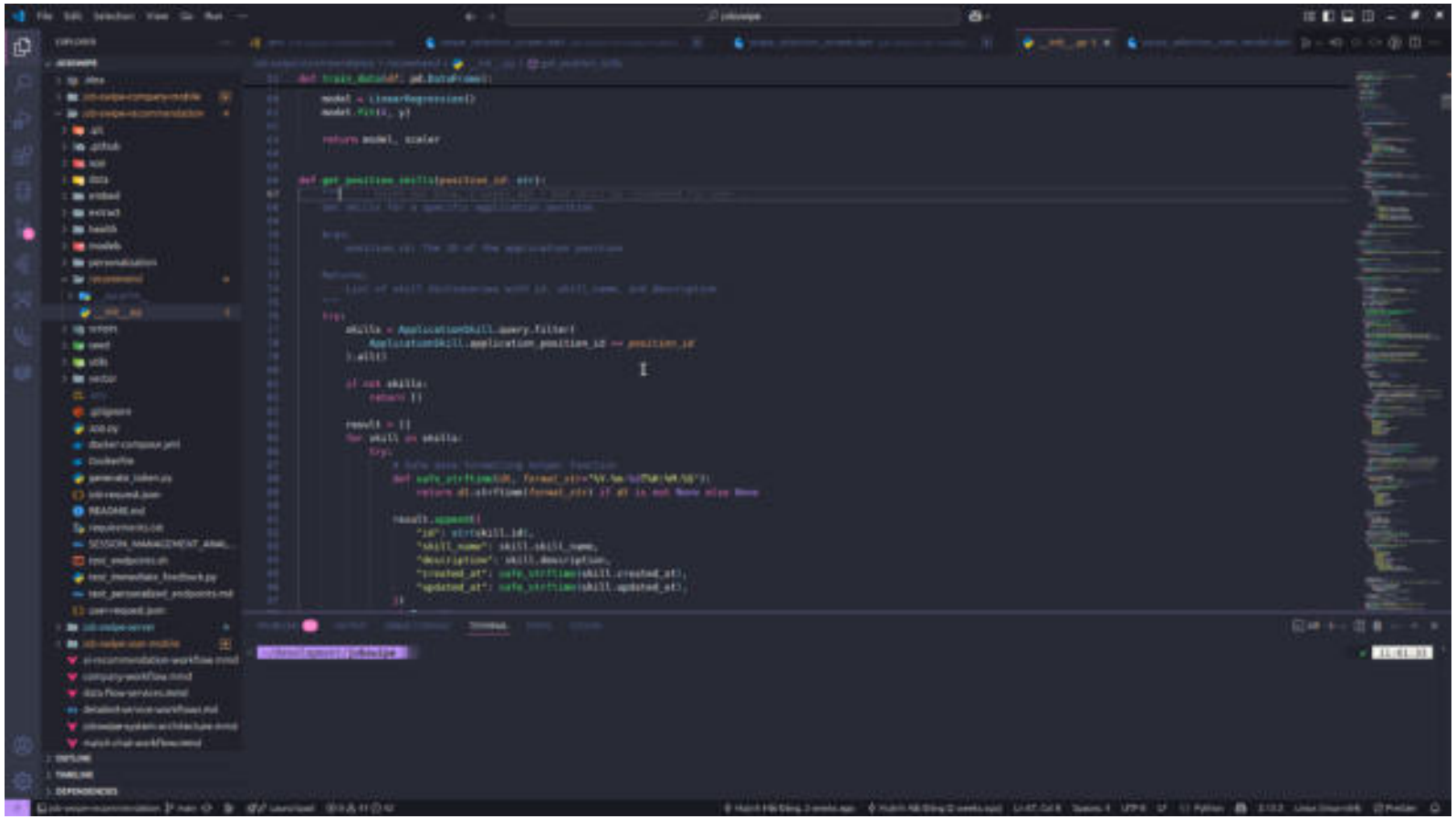
a) Khả năng lưu trữ các đối tượng lớn: Amazon S3 cho phép lưu trữ các đối tượng lên đến 5 terabyte.
b) Khả năng mở rộng: Amazon S3 có khả năng mở rộng đến hàng trăm tỷ đối tượng và hàng trăm petabyte dữ liệu.
c) Khả năng bảo mật cao: Amazon S3 cung cấp các tính năng bảo mật như mã hóa dữ liệu và quản lý quyền truy cập.
d) Khả năng truy cập dữ liệu từ mọi nơi: Amazon S3 cho phép truy cập dữ liệu từ bất kỳ nơi đâu trên thế giới thông qua giao thức HTTP hoặc HTTPS.
e) Khả năng sao lưu dữ liệu: Amazon S3 cho phép sao lưu dữ liệu tự động và thường xuyên thông qua tính năng Lifecycle.
f) Tính năng versioning: Amazon S3 cho phép lưu trữ nhiều phiên bản của cùng một đối tượng, giúp người dùng phục hồi dữ liệu nhanh chóng.
g) Tính năng truy xuất dữ liệu tốc độ cao: Amazon S3 cho phép truy xuất dữ liệu với tốc độ cao thông qua tính năng Amazon S3 Transfer Acceleration.
h) Khả năng tích hợp với các dịch vụ khác: Amazon S3 tích hợp tốt với các dịch vụ khác của AWS, giúp lập trình viên dễ dàng tích hợp với các ứng dụng khác nhau.

Về tổng quan, Amazon S3 là một dịch vụ lưu trữ đám mây linh hoạt, với các tính năng như khả năng lưu trữ các đối tượng lớn, khả năng mở rộng, khả năng bảo mật cao, khả năng truy cập dữ liệu từ mọi nơi, tính năng sao lưu dữ liệu, tính năng versioning
tính năng truy xuất dữ liệu tốc độ cao và khả năng tích hợp với các dịch vụ khác. Trong dự án này thì em đã sử dụng Filebase - một dịch vụ lưu trữ đám mây miễn phí được xây dựng dựa trên nền tảng AWS S3.

## 1.5. Công cụ sử dụng

### *1.5.1. Visual Studio Code*

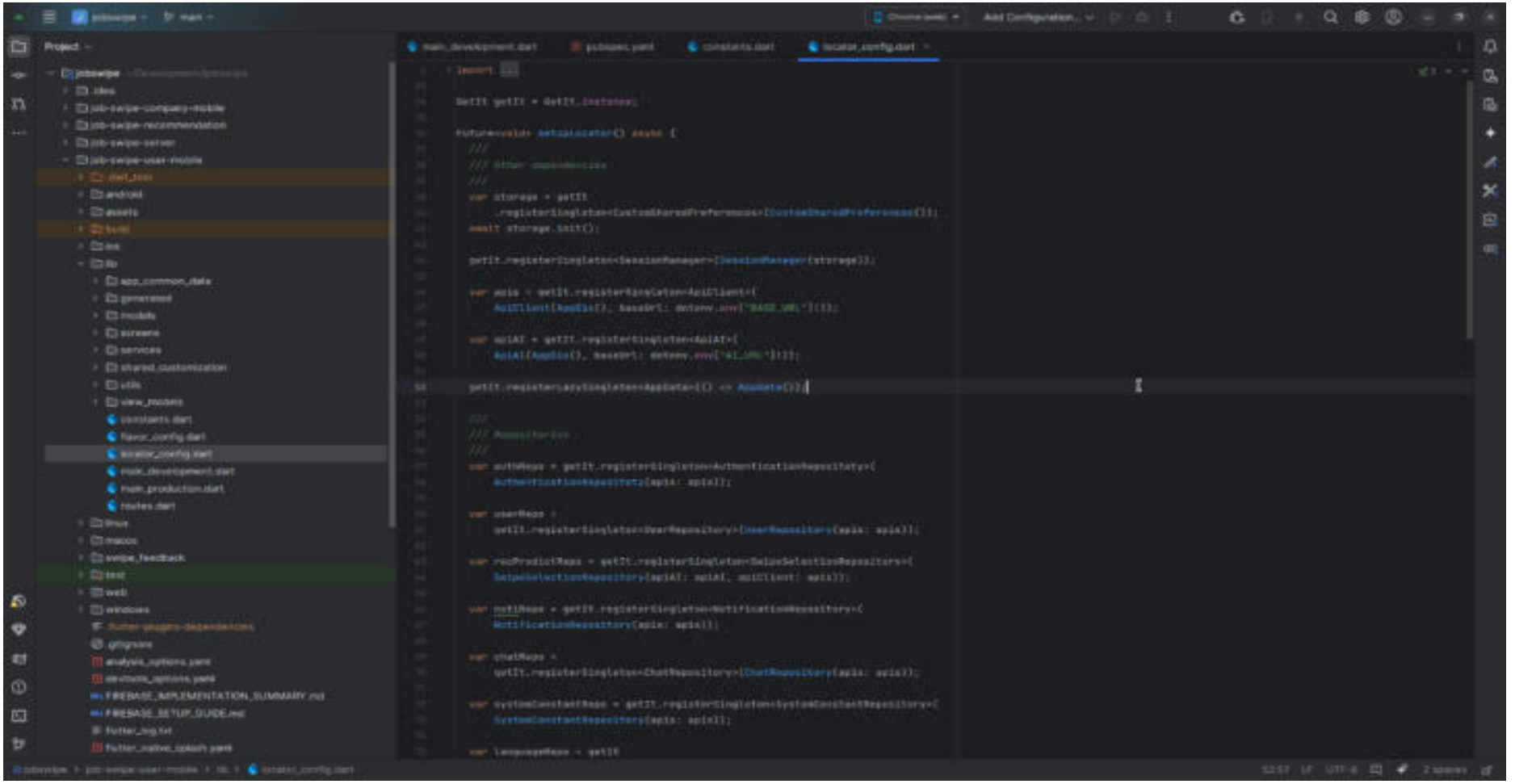
Xây dựng mã chương trình sử dụng công cụ VSCode, đặc điểm nổi bật là đơn giản, gọn nhẹ, dễ dàng cài đặt, hỗ trợ nhiều hệ điều hành cũng như nhiều ngôn ngữ lập trình khác nhau.



Hình 1.19: Sử dụng công cụ Visual Studio Code

### *1.5.2. Công cụ Android Studio*

Quá trình phân tích và thiết kế UML thì công cụ **draw.io** (link truy cập trên website: https://www.draw.io/) được sử dụng. Công cụ này giúp chúng ta có thể dễ dàng thao tác và sử dụng để tạo các sơ đồ  Use Case, sơ đồ hoạt động,….



*Hình 1.20: Sử dụng công cụ Android Studio*

### 1.5.3. Postman

Kiểm tra, chạy thử API của chương trình thông qua giao diện của Postman.

Postman là một trong những công cụ phổ biến nhất được sử dụng trong thử nghiệm các API. Với Postman, ta có thể gọi Rest API mà không cần viết dòng code nào và còn cho phép lưu lại lịch sử các lần request.

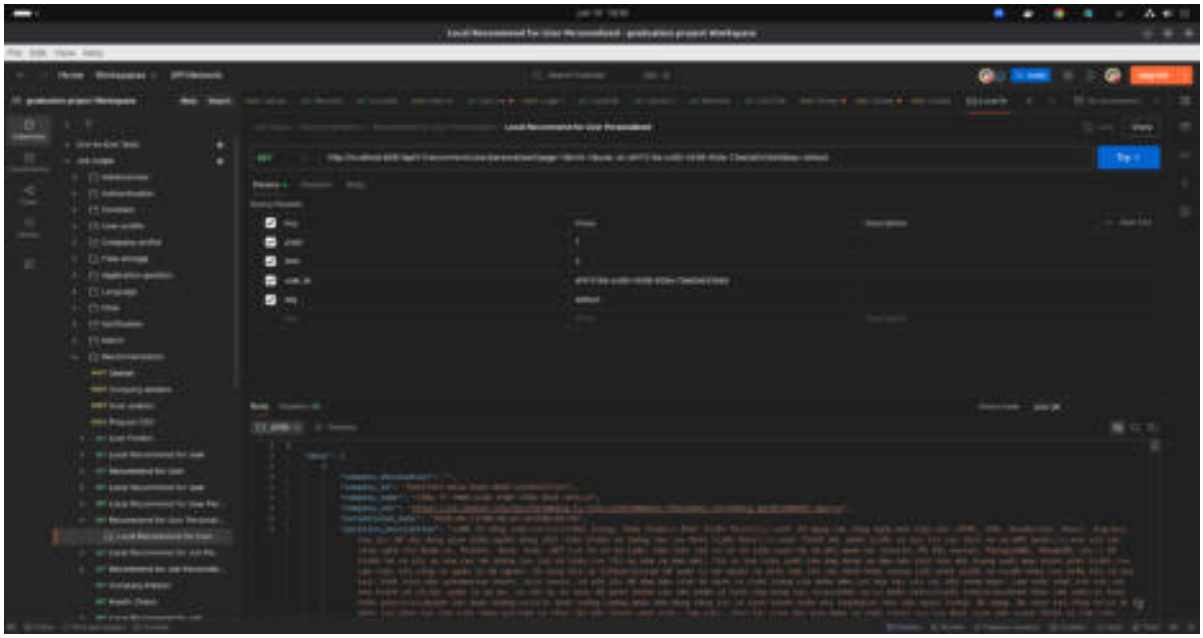
Postman hỗ trợ tất cả các phương thức HTTP (GET, POST, PUT, PATCH, DELETE).

Những lợi ích khi sử dụng Postman:

- Sử dụng Collections (Bộ sưu tập) – Postman cho phép người dùng tạo bộ sưu tập cho các lệnh gọi API của họ. Mỗi bộ sưu tập có thể tạo các thư mục con và nhiều yêu cầu (request). Điều này giúp việc tổ chức các bộ thử nghiệm.
- Collaboration – Collections và environment có thể được import hoặc export giúp chia sẻ tệp dễ dàng.
- API Testing – Test trạng thái phản hồi HTTP.
- Gỡ lỗi – Bảng điều khiển Postman giúp kiểm tra dữ liệu nào đã được truy xuất giúp dễ dàng gỡ lỗi kiểm tra.

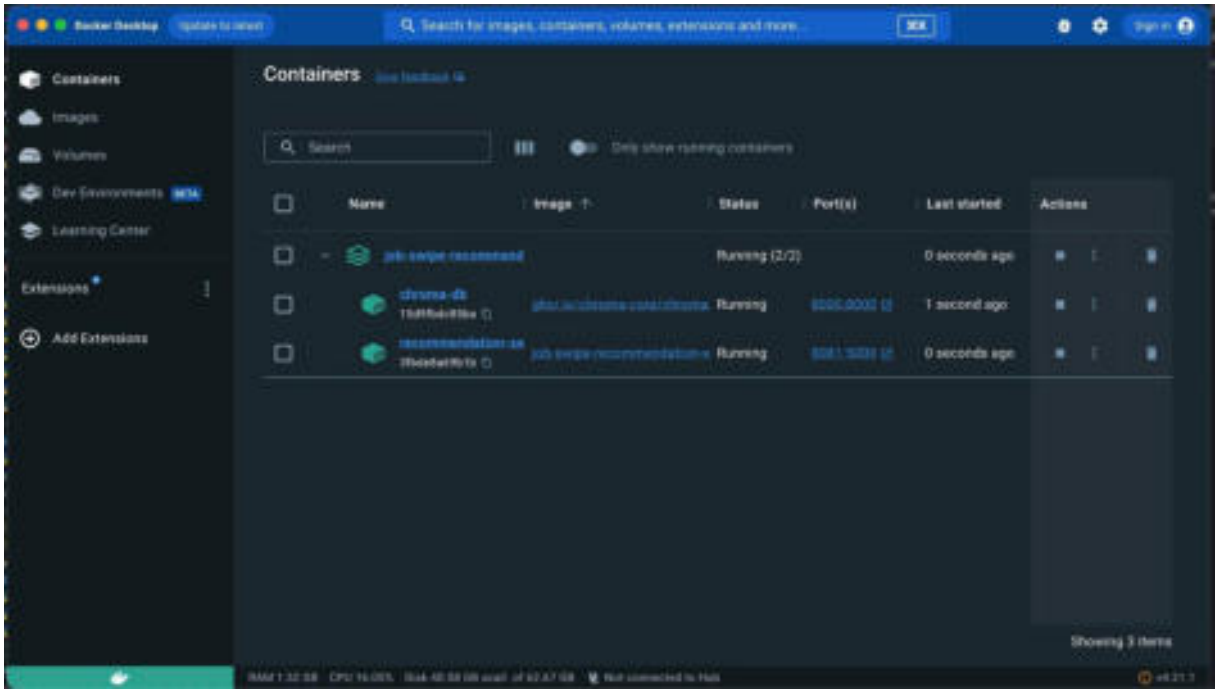
Các chức năng cơ bản

- Cho phép gửi HTTP Request với các method GET, POST, PUT, DELETE.
- Cho phép post dữ liệu dưới dạng form (key-value), text, json.
- Hiện kết quả trả về dạng text, hình ảnh, XML, JSON.
- Hỗ trợ authorization (Oauth1, 2).
- Cho phép thay đổi header của các request.



*Hình 1.21: Giao diện Postman*

### 1.5.4. Docker




*Hình 1.22: Giao diện Docker*

Docker là nền tảng container mã nguồn mở, tự động hóa triển khai và quản lý ứng dụng bằng cách đóng gói mã nguồn cùng toàn bộ phụ thuộc vào các container nhẹ, đảm bảo chạy nhất quán trên mọi môi trường.

Thành phần chính

- Docker Engine: Tạo và quản lý container.
- Images: Mẫu chỉ đọc chứa ứng dụng và thư viện.
- Containers: Phiên bản chạy được của image, cách ly nhau.
- Docker Hub: Kho lưu trữ và chia sẻ images.

Lợi ích

- Di động: Chạy trên bất kỳ hệ thống có Docker.
- Cách ly: Tránh xung đột giữa ứng dụng và môi trường.
- Hiệu quả: Nhẹ, khởi động nhanh hơn VM.
- Mở rộng: Kết hợp với công cụ điều phối để scale dễ dàng.
- Quản lý phiên bản: Gắn tag image, dễ rollback và đồng bộ triển khai.

Docker giúp đơn giản hóa CI/CD, tối ưu tài nguyên và tăng tính ổn định cho phát triển microservices và ứng dụng phân tán.

## 1.6. Kết luận chương 1

Thông qua việc học tập và nghiên cứu các kỹ thuật vector embedding, em đã có thể áp dụng kiến thức và kỹ năng lập trình của mình vào dự án xây dựng một hệ thống hỗ trợ kết nối giữa nhà tuyển dụng và người tìm việc. Bằng cách nhúng các vector đại diện cho công việc và hồ sơ ứng viên, hệ thống tính toán độ tương đồng giữa chúng để đưa ra các gợi ý công việc phù hợp nhất.

Bằng cách sử dụng các công nghệ hiện đại và nổi tiếng, mỗi công nghệ được chọn vì những ưu điểm cụ thể trong việc xây dựng các ứng dụng web và di động mạnh mẽ và có khả năng mở rộng, việc hiểu rõ các khái niệm và cách thức áp dụng chúng vào dự án là yếu tố then chốt. Điều này đã giúp quá trình phát triển hệ thống diễn ra nhanh chóng và hiệu quả hơn, mang lại một giải pháp toàn diện và thân thiện với người dùng.

# CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

Chương này sẽ đi vào chi tiết đặc tả các yêu cầu, các yêu cầu phi chức năng, các ràng buộc thiết kế cần thiết khác để đưa ra một mô tả đầy đủ và toàn diện về các yêu cầu của hệ thống. Cái này bao gồm đặc tả yêu cầu (Requirements specification), sơ đồ use case của từng đối tượng, đặc tả use case và sơ đồ hoạt động của hệ thống. Thể hiện một cách tổng quan những chức năng mà hệ thống có thể đáp ứng. Ngoài ra, còn định nghĩa kiến trúc, modules và dữ liệu cho một hệ thống để đáp ứng các yêu cầu cụ thể

## 2.1. Phân tích yêu cầu

### *2.1.1. Quan điểm về dự án*

Việc xây dựng vừa phát triển hệ thống hỗ trợ kết nối giữa người tuyển dụng của công ty và người tìm việc nhằm tạo ra một ứng dụng có thể đơn giản hóa quá trình tuyển dụng và có thể giúp người tìm việc liên lạc nhanh chóng được với nhà tuyển dụng.

### *2.1.2. Yêu cầu chức năng*

- **Đăng nhập, đăng kí và xác thực:** Chức năng này được sử dụng để nhà tuyển dụng và ứng viên có thể đăng kí tài khoản, đăng nhập vào tài khoản của họ và xác thực tính hợp lệ của họ.
- **Quản lý tin tuyển dụng:** Chức năng này cho phép nhà tuyển dụng quản lý các tin tuyển dụng của họ, bao gồm việc đăng tin mới, chỉnh sửa hoặc xóa tin, và theo dõi phản hồi từ ứng viên.
- **Lọc và hiển thị hồ sơ:** Bao gồm chức năng lọc các ứng viên trong khu vực, xem hồ sơ cá nhân, xem thông tin về kinh nghiệm làm việc và kỹ năng. Nhà tuyển dụng có thể đặt các tiêu chí tìm kiếm như ngành nghề, vị trí công việc, và trình độ học vấn để tìm kiếm ứng viên phù hợp.
- **Lọc và hiển thị mô tả công việc:** Bao gồm chức năng lọc các công việc trong khu vực, xem mô tả công việc, xem thông tin về công ty. Ứng viên có thể đặt các tiêu chí tìm kiếm như ngành nghề, vị trí công việc, và mức lương để tìm kiếm công việc phù hợp.
- **Quản lý kết nối:** Chức năng này cho phép nhà tuyển dụng quản lý danh sách ứng viên mà họ đã kết nối, bao gồm việc tương tác với họ qua tin nhắn và theo dõi tiến trình tuyển dụng.
- **Thông báo và cài đặt:** Chức năng này cho phép nhà tuyển dụng và ứng viên quản lý cài đặt thông báo và tuỳ chỉnh thông tin nhận được từ ứng dụng. Họ cũng có thể kiểm tra thông báo mới về các ứng viên quan tâm và cập nhật ứng dụng từ cửa hàng ứng dụng.

- **Tính năng quẹt tìm ứng viên và tìm công việc:** Tính năng này cho phép nhà tuyển dụng duyệt qua các hồ sơ ứng viên một cách nhanh chóng bằng cách quẹt qua trái hoặc phải. Khi nhà tuyển dụng quẹt sang phải, họ thể hiện sự quan tâm đến ứng viên và hồ sơ của họ được lưu lại cho việc tiếp tục xem xét. Tương tự với ứng viên khi tìm công việc phù hợp.
- **Tính năng gợi ý:** Tính năng này sử dụng mô hình, thuật toán để đề xuất các lựa chọn phù hợp với Người tìm việc và Người tuyển dụng của công ty, phụ thuộc vào những thông tin mà các bên đã cung cấp với hệ thống.
- **Tính năng phản hồi và cải thiện:** Tính năng này cho phép nhà tuyển dụng hoặc ứng viên cung cấp phản hồi về các ứng viên được đề xuất để cải thiện mô hình, thuật toán. Nhà tuyển dụng hoặc ứng viên có thể đánh giá và cung cấp thông tin về sự phù hợp của các đề xuất để mô hình có thể điều chỉnh và cải thiện việc đề xuất ứng viên trong tương lai.
- **Tính năng trò chuyện trực tuyến:** Tính năng này cho phép ứng viên và nhà tuyển dụng giao tiếp trực tiếp thông qua hệ thống trò chuyện tích hợp trong ứng dụng. Người tìm việc có thể gửi tin nhắn để hỏi về thông tin chi tiết về công việc hoặc lịch phỏng vấn, còn nhà tuyển dụng có thể trả lời các câu hỏi và tạo một môi trường giao tiếp thuận tiện.
- **Tính năng đặt lịch:** Tính năng này cho phép nhà tuyển dụng và ứng viên đặt lịch hẹn phỏng vấn hoặc cuộc họp thông qua ứng dụng. Người tìm việc có thể xem lịch trống của nhà tuyển dụng và đặt lịch phù hợp, còn nhà tuyển dụng có thể đề xuất các thời gian phỏng vấn và gửi mời lịch cho ứng viên.
- **Tính năng tải lên hồ sơ (resume):** Tính năng này cho phép người dùng tải lên hồ sơ cá nhân (CV) dưới dạng hình ảnh hoặc tệp PDF như .jpeg, .jpg, .png, hoặc .pdf. Sau khi được tải lên, hệ thống sẽ sử dụng API Llama Cloud để tự động trích xuất thông tin từ CV như họ tên, kinh nghiệm làm việc, kỹ năng, học vấn, v.v. Nhờ đó, người dùng không cần nhập lại thông tin thủ công, giúp tiết kiệm thời gian và đảm bảo độ chính xác của dữ liệu được sử dụng trong hệ thống.
- **Tính năng gửi mail xác nhận:** Tính năng này tự động gửi email xác nhận đến cả nhà tuyển dụng và ứng viên sau khi họ đã đặt lịch hẹn thành công. Email xác nhận sẽ chứa thông tin chi tiết về thời gian, địa điểm và nội dung cuộc họp hoặc phỏng vấn để đảm bảo mọi người đều biết và chuẩn bị sẵn sàng.
- **Quản lý Người dùng:** Chức năng này cho phép quản trị viên xem, chỉnh sửa và quản lý thông tin của tất cả người dùng trong hệ thống, bao gồm cả nhà tuyển dụng và ứng viên. Người quản trị có thể thực hiện các hành động như tạo, xóa và

khóa tài khoản, cũng như xem lịch sử hoạt động của người dùng.

- **Quản lý hệ thống và cài đặt:** Chức năng này cho phép Người quản trị quản lý các cài đặt và tùy chọn hệ thống, bao gồm cài đặt bảo mật, cập nhật thông tin về ứng dụng và quản lý các loại dữ liệu khác nhau như danh mục ngành nghề, kỹ năng, và trình độ học vấn.

### *2.1.3. Yêu cầu phi chức năng*

#### *2.1.3.1 Yêu cầu về tốc độ xử lý*

Hệ thống cần được tối ưu hóa toàn diện để đảm bảo tốc độ phản hồi nhanh chóng và trải nghiệm người dùng mượt mà trong mọi tình huống. Điều này bao gồm việc:

- Tối ưu hóa tốc độ truy vấn cơ sở dữ liệu: Sử dụng các kỹ thuật như lập chỉ mục (indexing), tối ưu hóa câu truy vấn (query optimization) và phân vùng dữ liệu (data partitioning) để giảm thiểu thời gian phản hồi của cơ sở dữ liệu;
- Sử dụng bộ nhớ đệm (caching): Lưu trữ tạm thời các kết quả tính toán, dữ liệu thường xuyên truy cập hoặc các trang web tĩnh trong bộ nhớ đệm để giảm tải cho cơ sở dữ liệu và tăng tốc độ truy xuất thông tin;
- Tối ưu hóa mã nguồn: Đảm bảo mã nguồn được viết hiệu quả, không chứa các đoạn mã dư thừa hoặc các vòng lặp không cần thiết. Sử dụng các thư viện và framework tối ưu hóa hiệu năng để xử lý các tác vụ phổ biến;
- Giảm thiểu thời gian tải trang: Tối ưu hóa kích thước các tệp tin (hình ảnh, CSS, JavaScript), sử dụng kỹ thuật nén (compression) và giảm thiểu số lượng yêu cầu HTTP để giảm thời gian tải trang;
- Theo dõi và phân tích hiệu năng: Thường xuyên theo dõi và phân tích hiệu năng của hệ thống để phát hiện các điểm nghẽn (bottlenecks) và các vấn đề về hiệu năng khác. Sử dụng các công cụ giám sát hiệu năng để thu thập dữ liệu và phân tích hiệu năng hệ thống.

#### *2.1.3.2 Yêu cầu về đường truyền*

Hệ thống được thiết kế để hoạt động ổn định và đáng tin cậy trên nhiều loại kết nối mạng, bao gồm:

- Mạng Internet cố định: Đảm bảo hệ thống hoạt động tốt trên các kết nối internet cáp quang, ADSL và các loại kết nối cố định khác với tốc độ và độ ổn định cao;
- Mạng di động (4G/5G): Đảm bảo hệ thống hoạt động tốt trên các kết nối mạng di động 4G và 5G, cung cấp trải nghiệm người dùng mượt mà ngay cả khi di chuyển;
- Kết nối kém ổn định: Tối ưu hóa hệ thống để hoạt động được trong điều kiện

kết nối mạng kém ổn định, chẳng hạn như khi sử dụng mạng di động ở vùng sâu vùng xa hoặc khi có nhiều người dùng truy cập đồng thời. c. Băng thông tối thiểu

- Yêu cầu băng thông tối thiểu 30Kbps là mức tối thiểu để đảm bảo các chức năng cơ bản của hệ thống hoạt động được.

- Tuy nhiên, để có trải nghiệm tốt nhất, đặc biệt là khi tải các tệp tin lớn hoặc sử dụng các tính năng yêu cầu băng thông cao như xem video hoặc thực hiện cuộc gọi video, người dùng nên sử dụng kết nối mạng có băng thông cao hơn.

#### *2.1.3.3 Yêu cầu tối thiểu về thiết bị và phần mềm*

- Hệ điều hành: Windows 10;
- RAM: 8GB;
- Vi xử lý: AMD Ryzen 5 hoặc Intel Core i5 thế hệ thứ 9 trở lên.

#### *2.1.3.4 Yêu cầu khuyên dùng về thiết bị và phần mềm*

- Hệ điều hành: Windows 11;
- RAM: 16GB;
- Vi xử lý: AMD Ryzen 7 hoặc Intel Core i7 thế hệ thứ 11 trở lên.
- Môi trường phát triển và ngôn ngữ lập trình: Lựa chọn môi trường và ngôn ngữ lập trình phổ biến, có cộng đồng hỗ trợ lớn, tài liệu hướng dẫn đầy đủ và các công cụ hỗ trợ phát triển ứng dụng mạnh mẽ.

Triển khai các công cụ và tính năng quản trị hệ thống toàn diện, bao gồm quản lý người dùng, phân quyền, sao lưu và khôi phục dữ liệu. Phân quyền chi tiết đến từng module và chức năng, đảm bảo người dùng chỉ có quyền truy cập và thực hiện các thao tác phù hợp với vai trò của mình.

Thiết kế hệ thống theo hướng mở rộng, cho phép dễ dàng thêm mới tính năng, mở rộng cơ sở dữ liệu và đáp ứng nhu cầu phát triển trong tương lai. Áp dụng các kỹ thuật phân cụm và cân bằng tải để đảm bảo hệ thống hoạt động ổn định khi lượng người dùng và dữ liệu tăng cao.

#### *2.1.3.5 Yêu cầu về bảo mật người dùng*

- Kiểm soát chặt chẽ việc truy cập
- Chỉ những người dùng được ủy quyền hợp lệ mới có thể truy cập vào ứng dụng và dữ liệu của nó. Hệ thống cần triển khai các biện pháp xác thực mạnh mẽ như mật khẩu hai yếu tố, sinh trắc học hoặc khóa bảo mật phần cứng để đảm bảo tính an toàn cao nhất.

- Cần phân chia quyền truy cập một cách rõ ràng cho từng người dùng, đảm bảo họ chỉ có thể thực hiện các thao tác phù hợp với vai trò và trách nhiệm của mình. Việc phân cấp này giúp hạn chế tối đa nguy cơ lạm dụng quyền hạn và rò rỉ dữ liệu.

- Mã hóa dữ liệu nhạy cảm: Tất cả các dữ liệu nhạy cảm của người dùng, bao gồm thông tin cá nhân, thông tin liên hệ, mật khẩu và các dữ liệu liên quan đến tài chính, sẽ được mã hóa bằng các thuật toán mã hóa mạnh mẽ như AES (Advanced Encryption Standard). Việc mã hóa này đảm bảo rằng ngay cả khi dữ liệu bị đánh cắp, kẻ tấn công cũng không thể đọc hoặc hiểu được nội dung của dữ liệu;
- Mã hóa dữ liệu truyền tải: Dữ liệu được truyền tải giữa máy chủ và máy khách cũng được mã hóa bằng giao thức HTTPS (Hypertext Transfer Protocol Secure). Giao thức này sử dụng các chứng chỉ SSL/ TLS để xác thực danh tính của máy chủ và mã hóa dữ liệu truyền đi, đảm bảo tính bảo mật và toàn vẹn của dữ liệu trong quá trình truyền tải;
- Quản lý khóa mã hóa: Hệ thống sử dụng cơ chế quản lý khóa mã hóa an toàn để bảo vệ các khóa mã hóa. Các khóa mã hóa được lưu trữ ở một vị trí an toàn và chỉ những người dùng được ủy quyền mới có thể truy cập và sử dụng chúng. Kiểm soát truy cập dữ liệu:
- Xác thực và ủy quyền: Hệ thống thực hiện xác thực người dùng thông qua tên người dùng và mật khẩu hoặc các phương thức xác thực mạnh hơn như xác thực hai yếu tố (2FA). Sau khi xác thực, hệ thống sẽ kiểm tra quyền hạn của người dùng để xác định xem họ có được phép truy cập vào dữ liệu cụ thể hay không;
- Phân quyền chi tiết: Hệ thống phân quyền chi tiết đến từng người dùng và từng loại dữ liệu. Ví dụ, một nhân viên tuyển dụng chỉ có thể truy cập vào hồ sơ của các ứng viên mà họ phụ trách, trong khi một quản trị viên hệ thống có thể truy cập vào tất cả dữ liệu;
- Ghi nhật ký truy cập: Hệ thống ghi lại chi tiết tất cả các hoạt động truy cập dữ liệu, bao gồm thông tin về người dùng, thời gian truy cập, loại dữ liệu được truy cập và các thao tác được thực hiện. Nhật ký truy cập này giúp quản trị viên giám sát hoạt động của hệ thống, phát hiện các truy cập trái phép và điều tra các sự cố bảo mật.
- Cập nhật hệ thống thường xuyên: Hệ thống được cập nhật thường xuyên để vá các lỗ hổng bảo mật mới nhất và đảm bảo hệ thống luôn ở trạng thái bảo mật tốt nhất;
- Sử dụng tường lửa: Hệ thống sử dụng tường lửa để kiểm soát lưu lượng mạng và ngăn chặn các truy cập trái phép từ bên ngoài.

- Phát hiện và ngăn chặn tấn công: Hệ thống sử dụng các phần mềm và công cụ bảo mật để phát hiện và ngăn chặn các cuộc tấn công mạng như tấn công từ chối dịch vụ (DoS), tấn công SQL injection và các loại tấn công khác;
- Quét và loại bỏ mã độc: Hệ thống thường xuyên quét và loại bỏ các phần mềm độc hại như virus, worm và trojan để bảo vệ hệ thống khỏi các mối đe dọa từ mã độc.

#### *2.1.3.6 Yêu cầu về giao diện và bộ chữ*

Giao diện người dùng trực quan, dễ sử dụng và thân thiện: Giao diện được thiết kế với bố cục rõ ràng, hợp lý, các thành phần được sắp xếp khoa học và dễ dàng tiếp cận. Màu sắc, hình ảnh và các yếu tố đồ họa được sử dụng hài hòa, tạo cảm giác thoải mái và dễ chịu cho người dùng. Các nút bấm, biểu tượng và menu được thiết kế dễ nhận biết và sử dụng;

- Hỗ trợ đa ngôn ngữ: Hệ thống hỗ trợ nhiều ngôn ngữ khác nhau, cho phép người dùng lựa chọn ngôn ngữ hiển thị phù hợp với nhu cầu và sở thích của họ. Ngôn ngữ mặc định của hệ thống là tiếng Anh, và người dùng có thể dễ dàng chuyển đổi sang các ngôn ngữ khác được hỗ trợ;
- Tối ưu hóa trải nghiệm người dùng: Giao diện được thiết kế đáp ứng các nguyên tắc thiết kế UI/UX (User Interface/ User Experience), đảm bảo tính nhất quán, dễ sử dụng và hiệu quả trên cả máy tính để bàn và thiết bị di động. Giao diện cần thích ứng với các kích thước màn hình và các thiết bị khác nhau, đảm bảo trải nghiệm người dùng luôn mượt mà và tối ưu.
- Bộ chữ Unicode UTF-8: Hệ thống sử dụng bộ chữ Unicode UTF-8 để đảm bảo khả năng hiển thị đầy đủ và chính xác các ký tự đặc biệt, bao gồm các ký tự trong các ngôn ngữ khác nhau trên thế giới. Việc sử dụng Unicode UTF-8 giúp tránh các vấn đề về lỗi font chữ, hiển thị sai ký tự và đảm bảo tính toàn vẹn của dữ liệu văn bản;
- Bộ chữ hỗ trợ đa dạng font chữ: Hệ thống hỗ trợ nhiều loại font chữ khác nhau, cho phép người dùng tùy chỉnh giao diện theo sở thích cá nhân. Tuy nhiên, cần đảm bảo rằng các font chữ được lựa chọn phải hỗ trợ đầy đủ bộ ký tự Unicode UTF-8 để tránh các vấn đề về hiển thị.

#### *2.1.3.7 Yêu cầu về tính toàn vẹn dữ liệu*

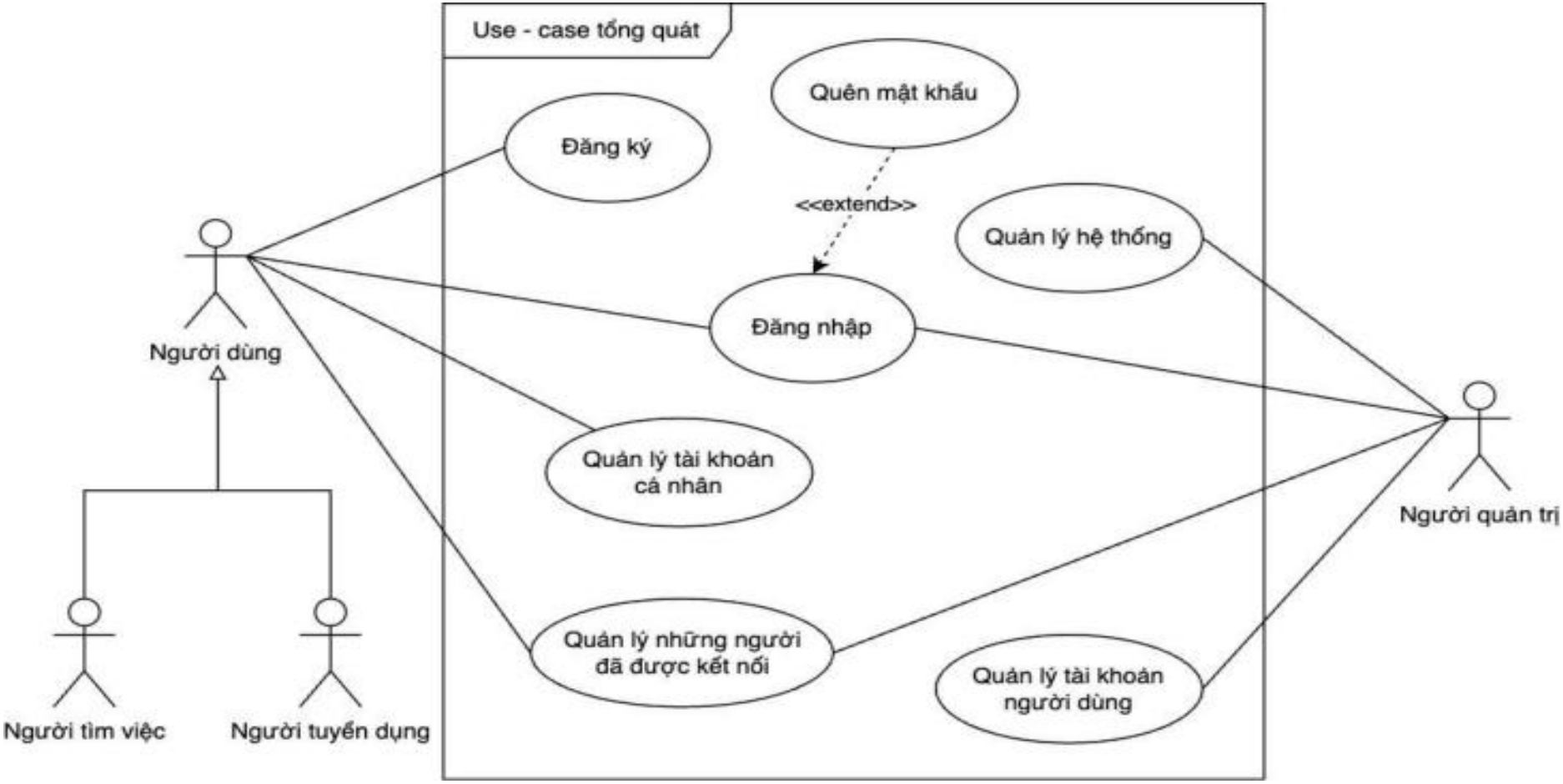
Dữ liệu được thu thập từ những nguồn đáng tin cậy, chẳng hạn như thông tin do chính người dùng cung cấp và xác minh qua email hoặc số điện thoại. Ngoài ra, hệ thống có thể tích hợp với các nguồn dữ liệu uy tín khác để đảm bảo tính chính xác của thông tin.

- Trước khi lưu trữ, dữ liệu sẽ trải qua quá trình xác minh tự động và thủ công. Các quy trình kiểm tra tự động sẽ kiểm tra định dạng, kiểu dữ liệu và các ràng buộc cơ bản. Đối với những thông tin quan trọng, hệ thống sẽ có các bước xác minh bổ sung như yêu cầu người dùng cung cấp bằng chứng hoặc xác nhận qua email/ tin nhắn.
- Hệ thống sẽ thực hiện kiểm tra định kỳ và đối chiếu dữ liệu giữa các nguồn khác nhau để đảm bảo tính nhất quán và phát hiện sớm các sai lệch hoặc lỗi dữ liệu. Các quy trình đối chiếu này có thể bao gồm việc so sánh thông tin người dùng với các cơ sở dữ liệu bên ngoài hoặc đối chiếu dữ liệu giữa các phiên bản khác nhau của hệ thống.
- Khi phát hiện ra lỗi hoặc sai lệch trong dữ liệu, hệ thống sẽ có cơ chế tự động hoặc thủ công để sửa lỗi. Người dùng cũng có thể báo cáo các lỗi dữ liệu mà họ phát hiện, và hệ thống sẽ có quy trình xử lý các báo cáo này một cách nhanh chóng và hiệu quả.
- Hệ thống sử dụng cơ sở dữ liệu quan hệ (RDBMS) để đảm bảo tính nhất quán và toàn vẹn dữ liệu. RDBMS cung cấp các tính năng như khóa (locking), giao dịch (transaction) và ràng buộc toàn vẹn (constraints) để đảm bảo rằng dữ liệu luôn ở trạng thái hợp lệ và nhất quán.
- Các ràng buộc toàn vẹn được thiết lập trong cơ sở dữ liệu để kiểm soát tính hợp lệ của dữ liệu. Ví dụ, ràng buộc khóa ngoại (foreign key constraint) đảm bảo rằng các tham chiếu đến các bảng khác là hợp lệ, trong khi ràng buộc duy nhất (unique constraint) đảm bảo rằng các giá trị trong một cột là duy nhất.
- Hệ thống thực hiện sao lưu dữ liệu định kỳ để phòng ngừa mất mát dữ liệu do lỗi phần cứng, lỗi phần mềm hoặc các sự cố khác. Các bản sao lưu này được lưu trữ ở một vị trí an toàn và có thể được sử dụng để khôi phục dữ liệu trong trường hợp cần thiết.
- Hệ thống thu thập đầy đủ các thông tin cần thiết cho hoạt động của ứng dụng, bao gồm thông tin cá nhân của người dùng, thông tin về công việc, kỹ năng, kinh nghiệm làm việc, …
- Các trường dữ liệu được phân loại rõ ràng thành các trường bắt buộc và không bắt buộc. Người dùng sẽ được yêu cầu nhập đầy đủ các trường bắt buộc trước khi có thể hoàn thành một tác vụ cụ thể.
- Hệ thống cung cấp các tùy chọn bổ sung cho người dùng để nhập thêm thông tin nếu họ muốn. Ví dụ, người dùng có thể thêm các kỹ năng bổ sung, chứng chỉ hoặc kinh nghiệm làm việc không bắt buộc.

- Hệ thống xây dựng các quy tắc kiểm tra dữ liệu đầu vào để đảm bảo rằng dữ liệu nhập vào hệ thống tuân thủ đúng định dạng và yêu cầu nghiệp vụ. Ví dụ, hệ thống có thể kiểm tra xem địa chỉ email có đúng định dạng hay không, số điện thoại có đúng số lượng chữ số hay không, …
- Khi phát hiện dữ liệu không hợp lệ, hệ thống sẽ hiển thị các thông báo lỗi rõ ràng và dễ hiểu để hướng dẫn người dùng nhập lại dữ liệu. Các thông báo lỗi này cần chỉ rõ lỗi cụ thể và cách khắc phục.
- Trước khi xử lý và lưu trữ dữ liệu, hệ thống sẽ thực hiện kiểm tra tính hợp lệ của dữ liệu. Nếu dữ liệu không hợp lệ, hệ thống sẽ không xử lý hoặc lưu trữ dữ liệu đó và sẽ yêu cầu người dùng sửa lỗi.

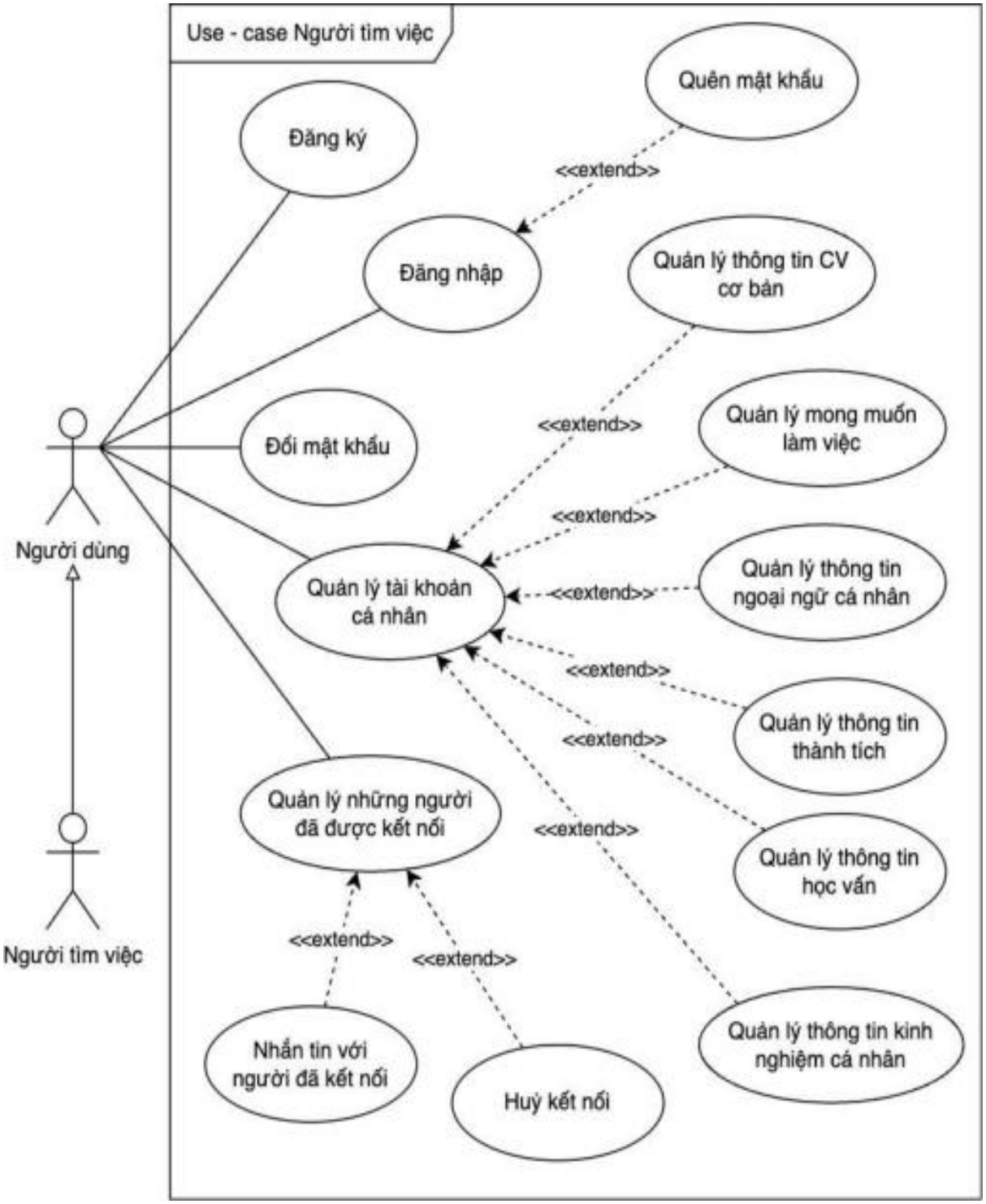
## 2.2. Use-case diagrams

### *2.2.1 Sơ đồ Use Case tổng quan của hệ thống*

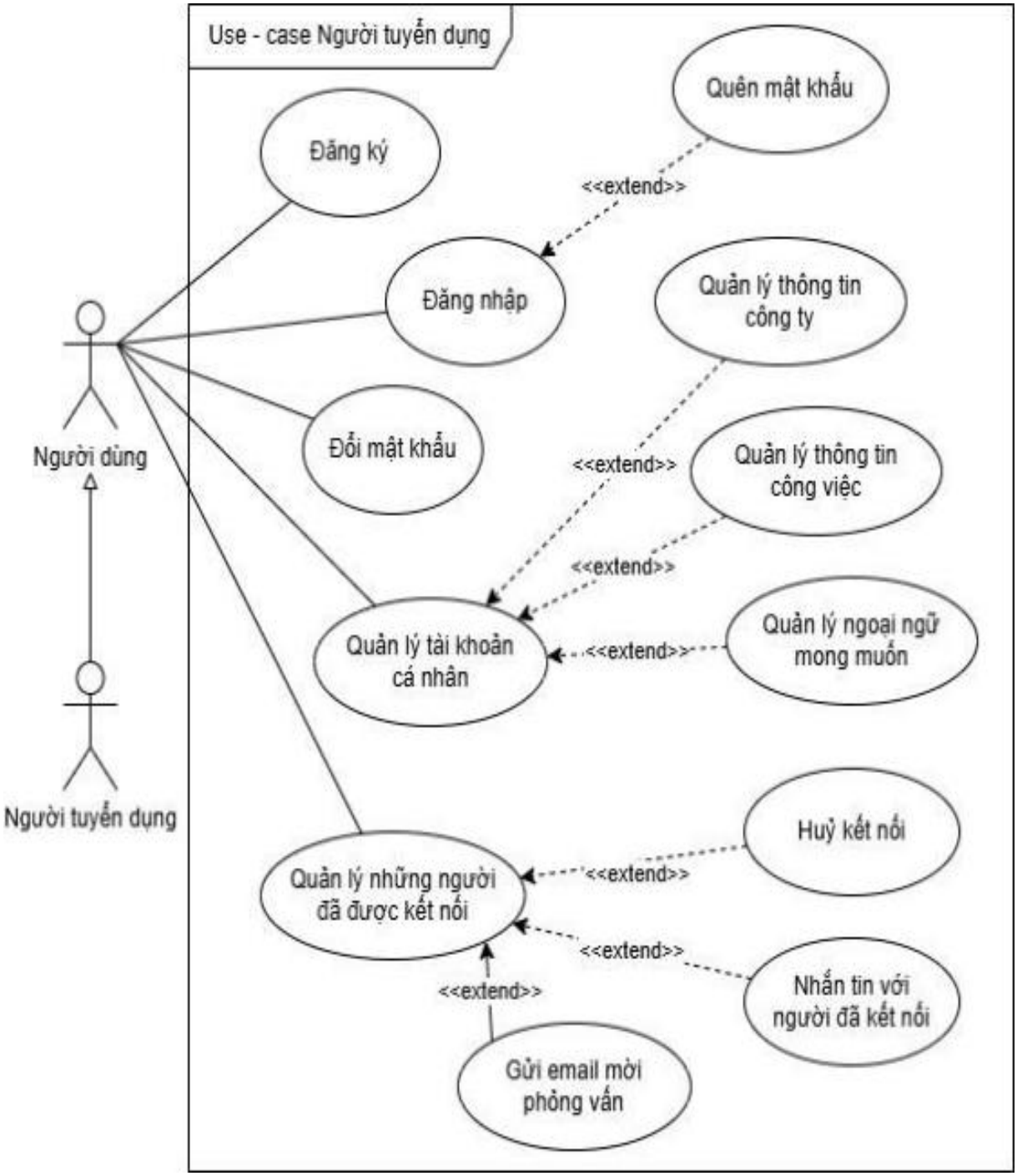



*Hình 2.1: Sơ đồ Use Case tổng quan của hệ thống*

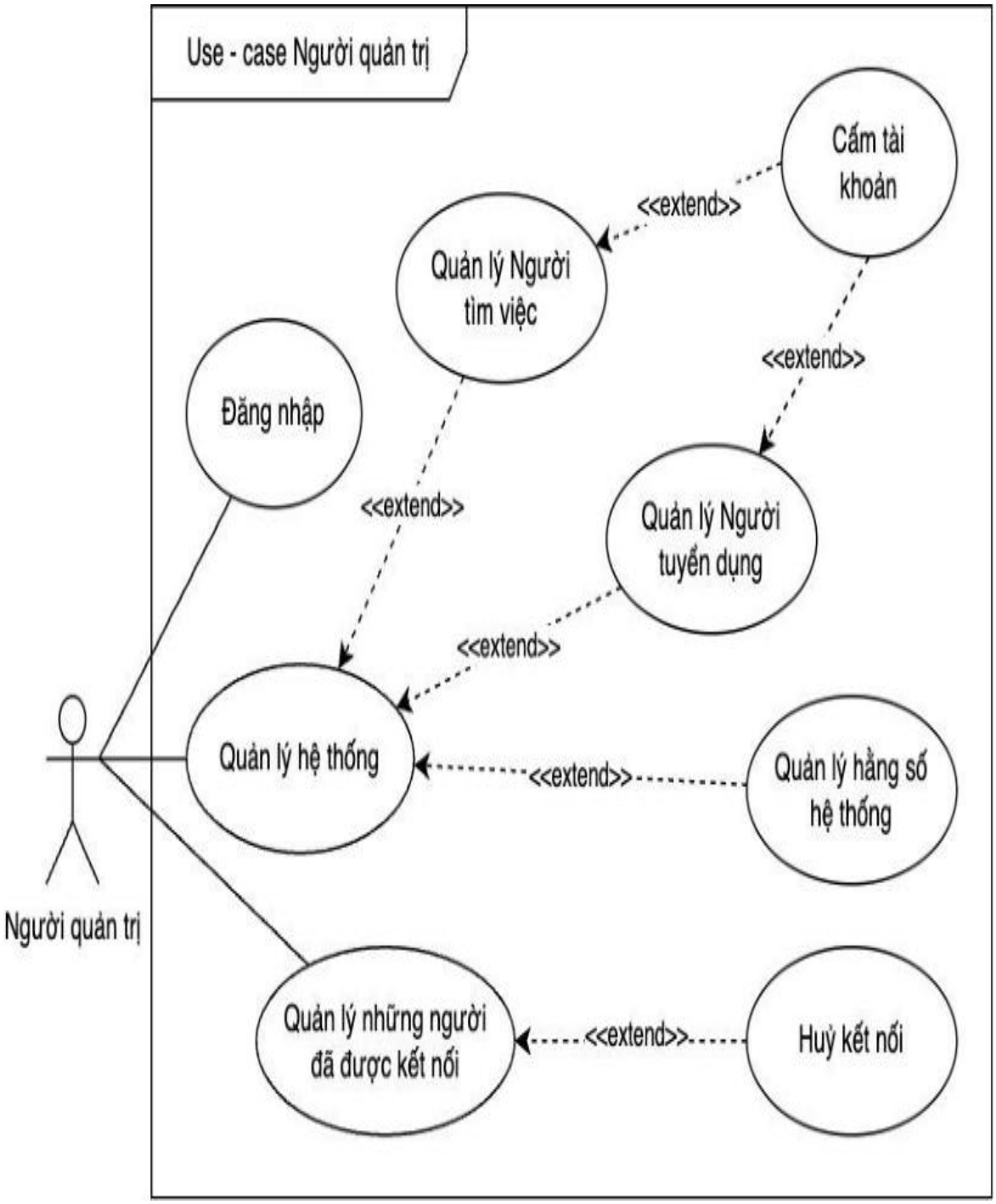
### 2.2.2 Sơ đồ usecase theo tác nhân




*Hình 2.2: Sơ đồ Use Case của người tìm việc*




*Hình 2.3: Sơ đồ Use Case của nhà tuyển dụng*




*Hình 2.4: Sơ đồ Use case của người quản trị*

### *2.2.3 Đặc tả Use Case*

**Đặc tả các Use Cases chính:**

*Bảng 2.1: Bảng đặc tả Use Case đăng nhập*

| **Mã Use Case** | **UC-02** | | |
|---|---|---|---|
| **Tên Use Case** | Đăng nhập | | |
| **Tác nhân** | Người quản trị;<br>Người tuyển dụng;<br>Người tìm việc | | |
| **Mô tả** | Là một Người quản trị, tôi muốn đăng nhập để sử dụng hệ thống;<br>Là một Người tuyển dụng, tôi muốn đăng nhập để sử dụng hệ thống;<br>Là một Người tìm việc, tôi muốn đăng nhập để sử dụng hệ thống. | | |
| **Điều kiện trước** | Tài khoản người dùng đã được tạo sẵn;<br>Tài khoản người dùng đã được phân quyền;<br>Thiết bị của tác nhân đã được kết nối Internet khi thực hiện đăng nhập. | | |
| **Điều kiện sau** | Tác nhân đăng nhập vào ứng dụng thành công. | | |
| | | Hành động của tác nhân | Hành động của hệ thống |
| | 1 | Tác nhân (Người quản trị, Người tuyển dụng, hoặc Người tìm việc) truy cập vào trang đăng nhập của hệ thống. | Hệ thống hiển thị giao diện trang đăng nhập, bao gồm các trường để nhập thông tin tài khoản (tên đăng nhập/email và mật khẩu). |

| | | | |
|---|---|---|---|
| **Kịch bản** | 2 | Tác nhân nhập tên đăng nhập (hoặc email) và mật khẩu vào các trường tương ứng, sau đó nhấn nút "Đăng nhập". | Hệ thống nhận thông tin đăng nhập và kiểm tra tính hợp lệ của tài khoản (so sánh với cơ sở dữ liệu). |
| | 3 | (Trường hợp lỗi) Nếu thông tin đăng nhập không chính xác, tác nhân sẽ nhận được thông báo lỗi và yêu cầu nhập lại. | Hệ thống hiển thị thông báo lỗi. |
| | 4 | (Trường hợp thành công) Nếu thông tin đăng nhập chính xác, tác nhân được chuyển hướng đến trang chính của vai trò tương ứng (ví dụ: bảng điều khiển cho Người quản trị, trang quản lý tuyển dụng cho Người tuyển dụng, hoặc trang tìm việc cho Người tìm việc) | Hệ thống xác thực thông tin đăng nhập, cấp quyền truy cập dựa trên vai trò của tài khoản, và chuyển hướng tác nhân đến giao diện phù hợp với vai trò của họ. Đồng thời, hệ thống lưu phiên đăng nhập (session) vào cơ sở dữ liệu. |

*Bảng 2.2: Bảng đặc tả Use Case đăng ký*

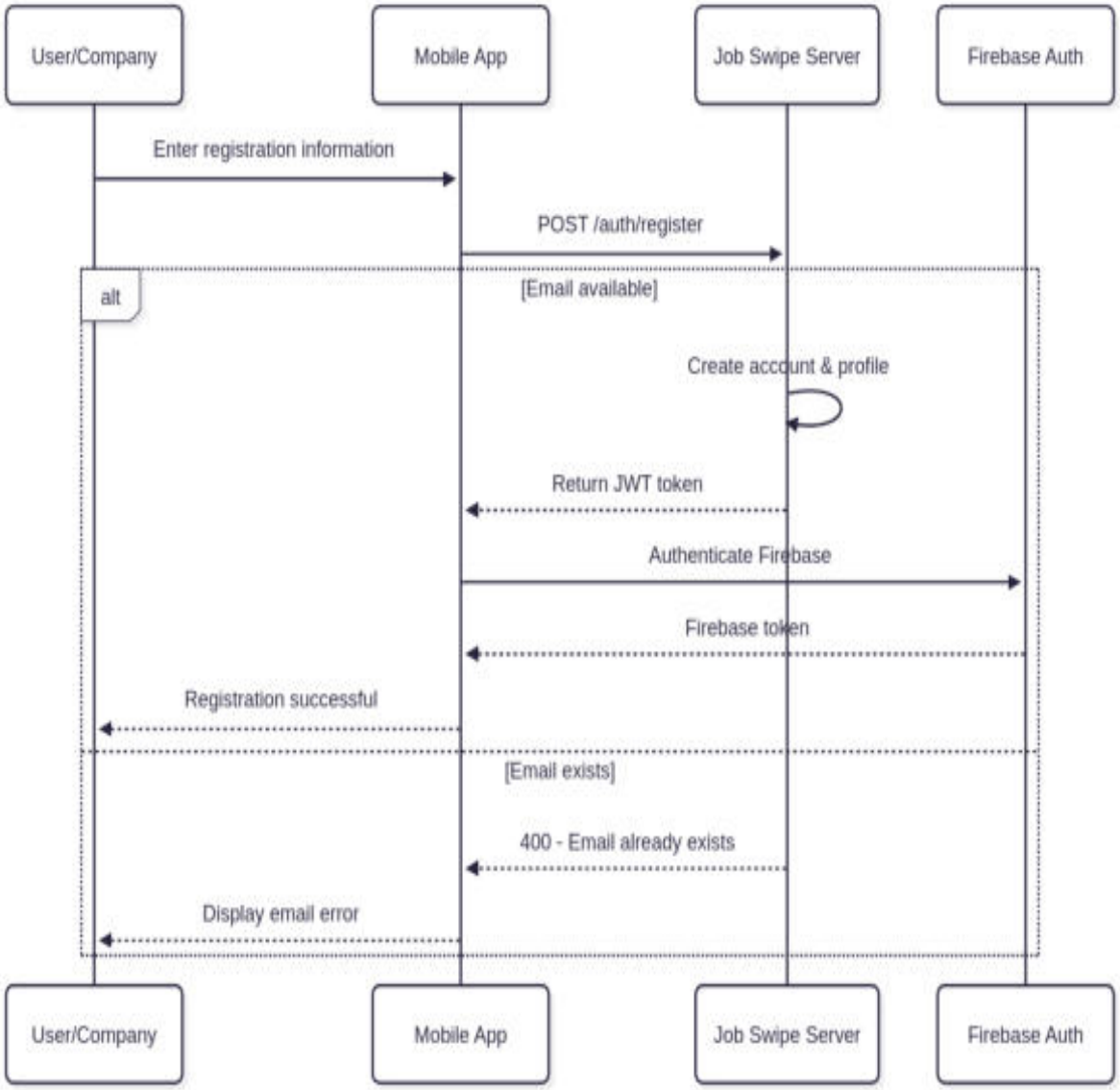
| **Mã Use Case** | **UC-03** | | |
|---|---|---|---|
| **Tên Use Case** | Đăng ký | | |
| **Tác nhân** | Người tuyển dụng;<br>Người tìm việc | | |
| **Mô tả** | Là một Người tuyển dụng, tôi muốn đăng ký để tạo tài khoản với vai trò mong muốn;<br>Là một Người tìm việc, tôi muốn đăng ký để tạo tài khoản với vai trò mong muốn; | | |
| **Điều kiện trước** | Thiết bị của tác nhân đã được kết nối Internet khi thực hiện đăng ký. | | |
| **Điều kiện sau** | Tác nhân có tài khoản hợp lệ để đăng nhập vào ứng dụng. | | |
| **Kịch bản** | | **Hành động của tác nhân** | **Hành động của hệ thống** |
| | 1 | Truy cập trang đăng ký. | Hiển thị form đăng ký với các trường: vai trò, tên, email, mật khẩu. |
| | 2 | Người dùng chọn thời gian bắt đầu và thời gian kết thúc cho lịch hẹn, sau đó bấm nút "Register" | Hệ thống chuyển sang trang yêu cầu người dùng nhập ghi chú |
| | 3 | Người dùng có thể thêm ghi chú hoặc không, sau đó bấm nút "Book" | Người dùng có thể thêm ghi chú hoặc không, sau đó bấm nút "Book" |

*Bảng 2.3: Bảng đặc tả Use Case đổi mật khẩu*

| **Mã Use Case** | **UC-04** | | |
|---|---|---|---|
| **Tên Use Case** | Đổi mật khẩu | | |
| **Tác nhân** | Người tuyển dụng;<br>Người tìm việc | | |
| **Mô tả** | Là một Người tuyển dụng, tôi muốn đổi mật khẩu mới để thay thế mật khẩu hiện tại;<br>Là một Người tìm việc, tôi muốn đổi mật khẩu mới để thay thế mật khẩu hiện tại. | | |
| **Điều kiện trước** | Tác nhân phải biết được mật khẩu hiện tại của tài khoản;<br>Thiết bị của tác nhân đã được kết nối Internet khi thực hiện đổi mật khẩu. | | |
| **Điều kiện sau** | Tác nhân có thể đăng nhập vào ứng dụng bằng mật khẩu mới. | | |
| **Kịch bản**<br><br>**Kịch bản** | | Hành động của tác nhân | Hành động của hệ thống |
| | 1 | Nhấn vào nút "add new" ở bảng sản phẩm | Hệ thống chuyển người dùng đến trang tạo sản phẩm mới |
| | 2 | Người quản lý nhập thông tin sản phẩm mới kèm hình ảnh của sản phẩm. Sau đó nhấn nút "Save" để hoàn thành tạo sản phẩm mới. | Hệ thống nhận thông tin trả về và lưu vào cơ sở dữ liệu. Chuyển hướng người dùng quay lại trang danh sách sản phẩm. |

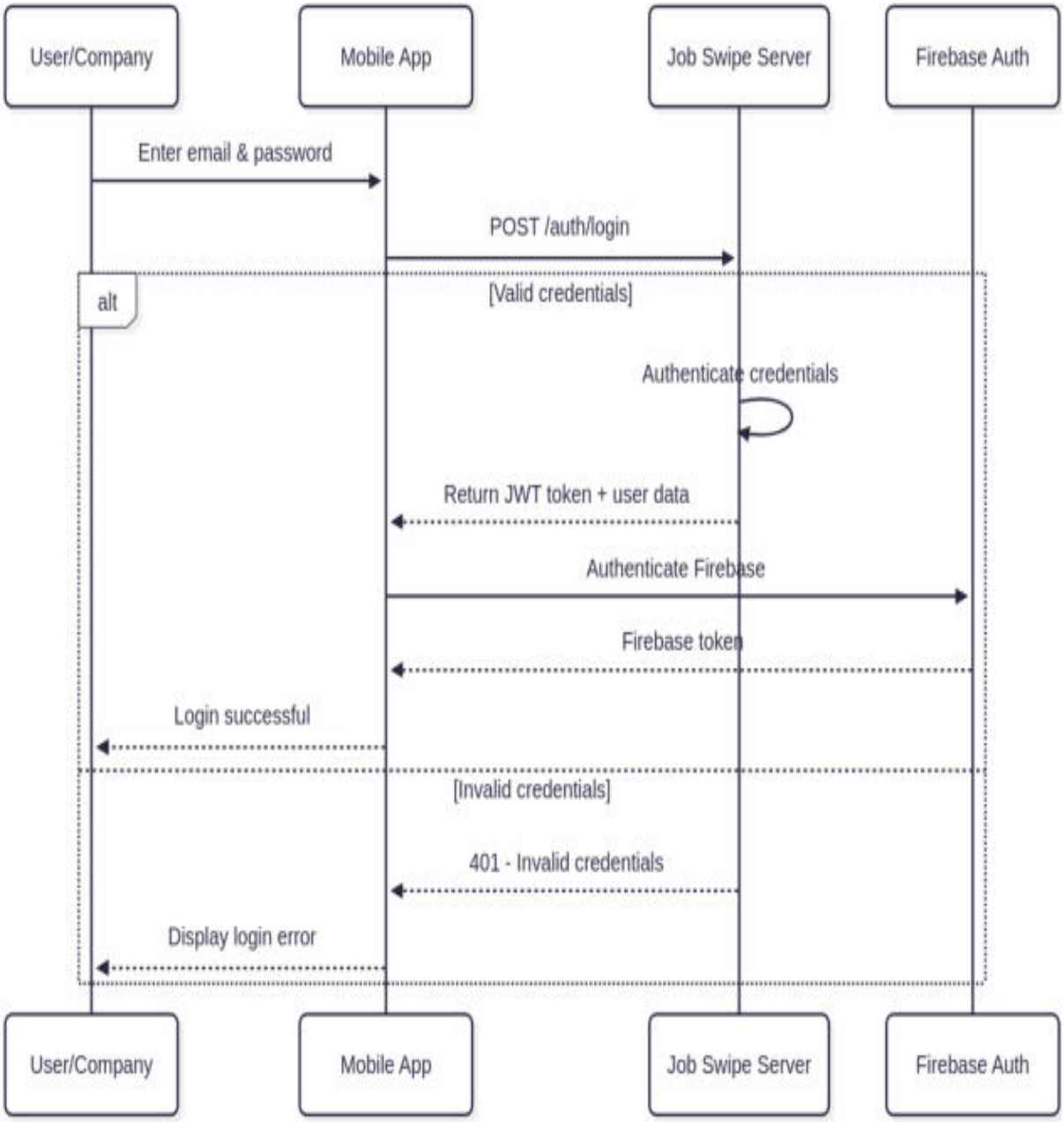
### 2.2.4 Sơ đồ tuần tự của hệ thống

#### 2.2.4.1. Sơ đồ tuần tự đăng ký



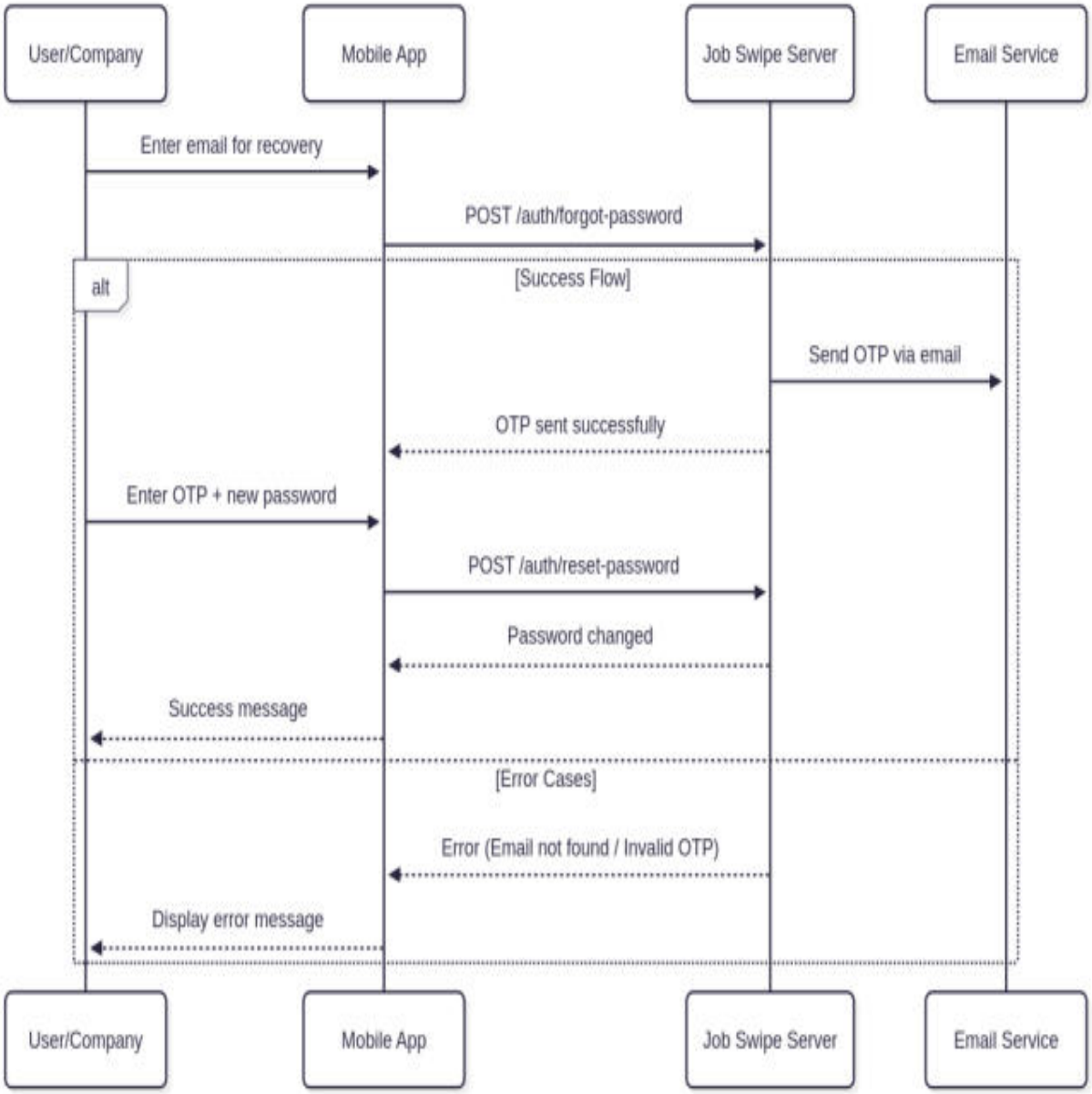

Hình 2.5: Sơ đồ tuần tự đăng ký

### 2.2.4.2. Sơ đồ tuần tự đăng nhập



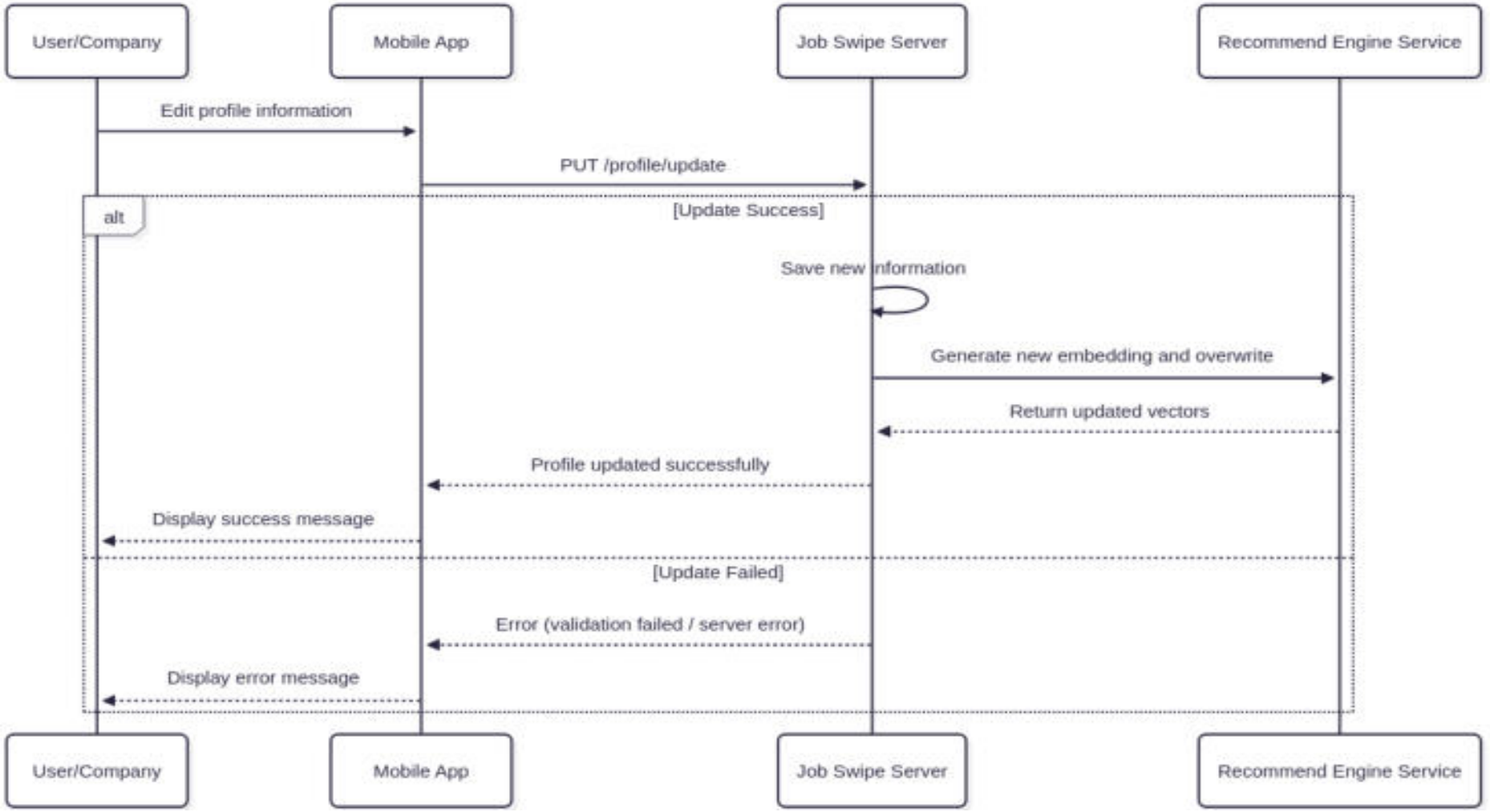

*Hình 2.6: Sơ đồ tuần tự đăng nhập*

### 2.2.4.3. Sơ đồ tuần tự khôi phục mật khẩu



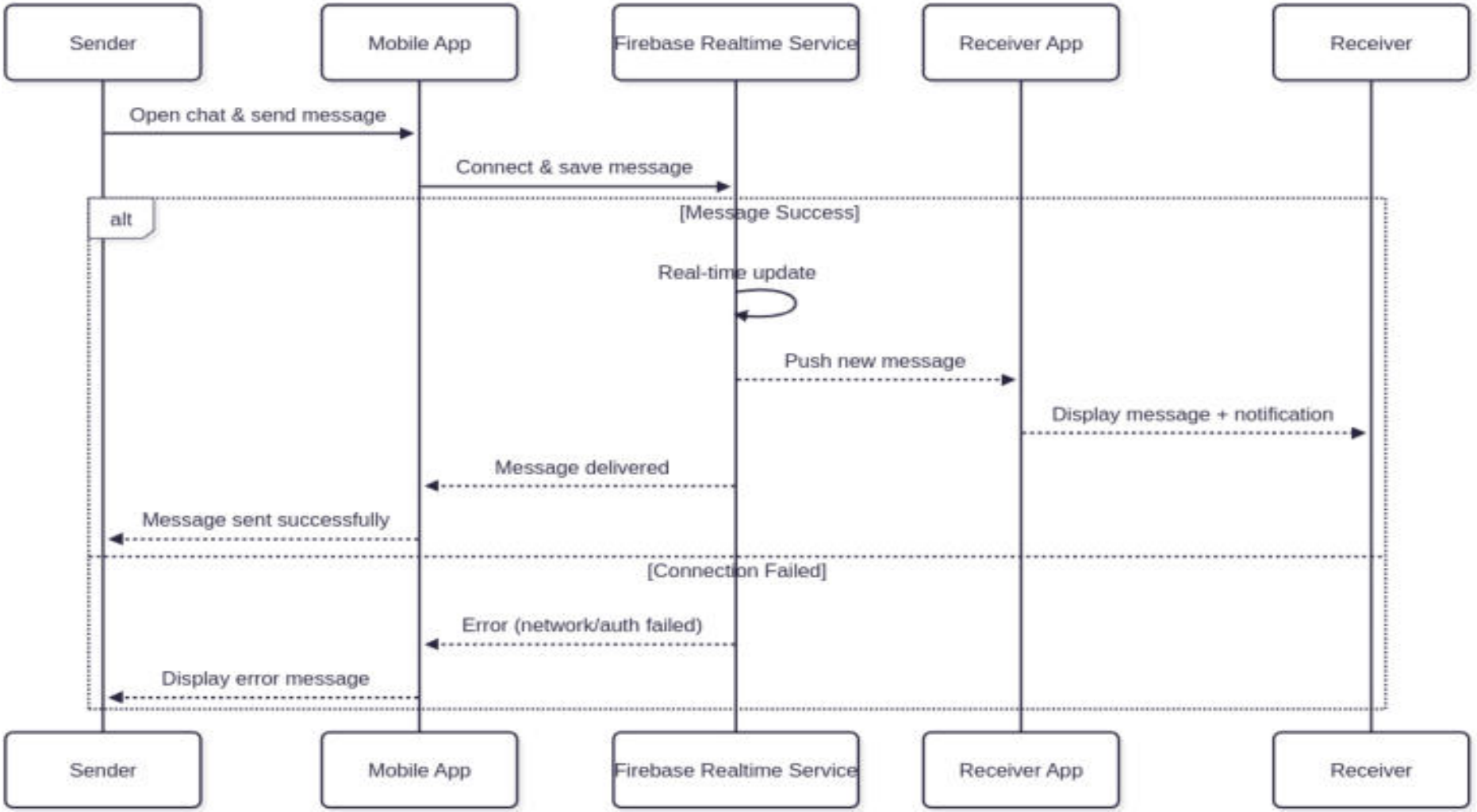

*Hình 2.7: Sơ đồ tuần tự khôi phục mật khẩu*

### 2.2.4.4. Sơ đồ tuần tự cập nhật dữ liệu



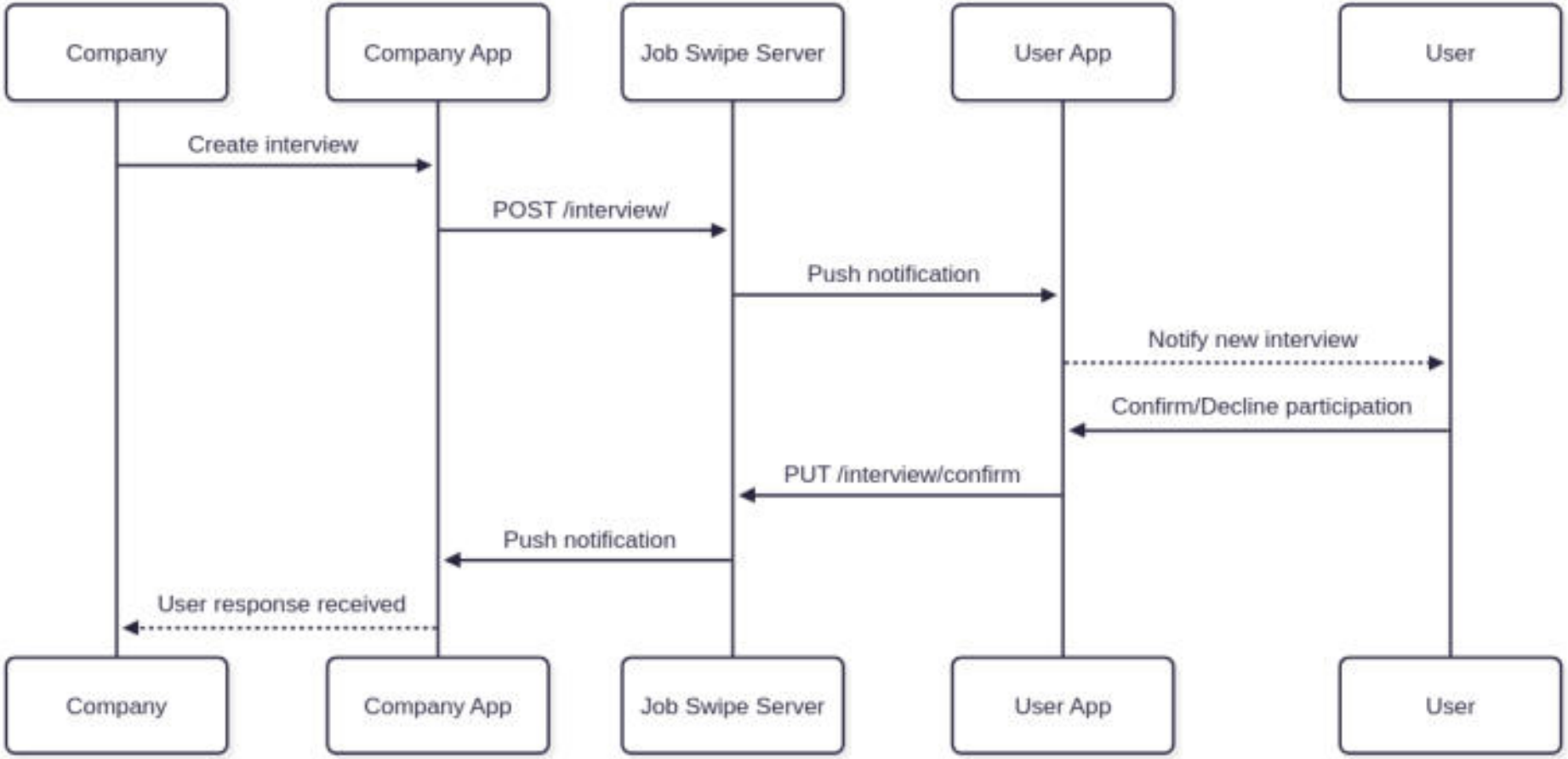

*Hình 2.8: Sơ đồ tuần tự cập nhật dữ liệu*

### 2.2.4.5. Sơ đồ tuần tự realtime – chat



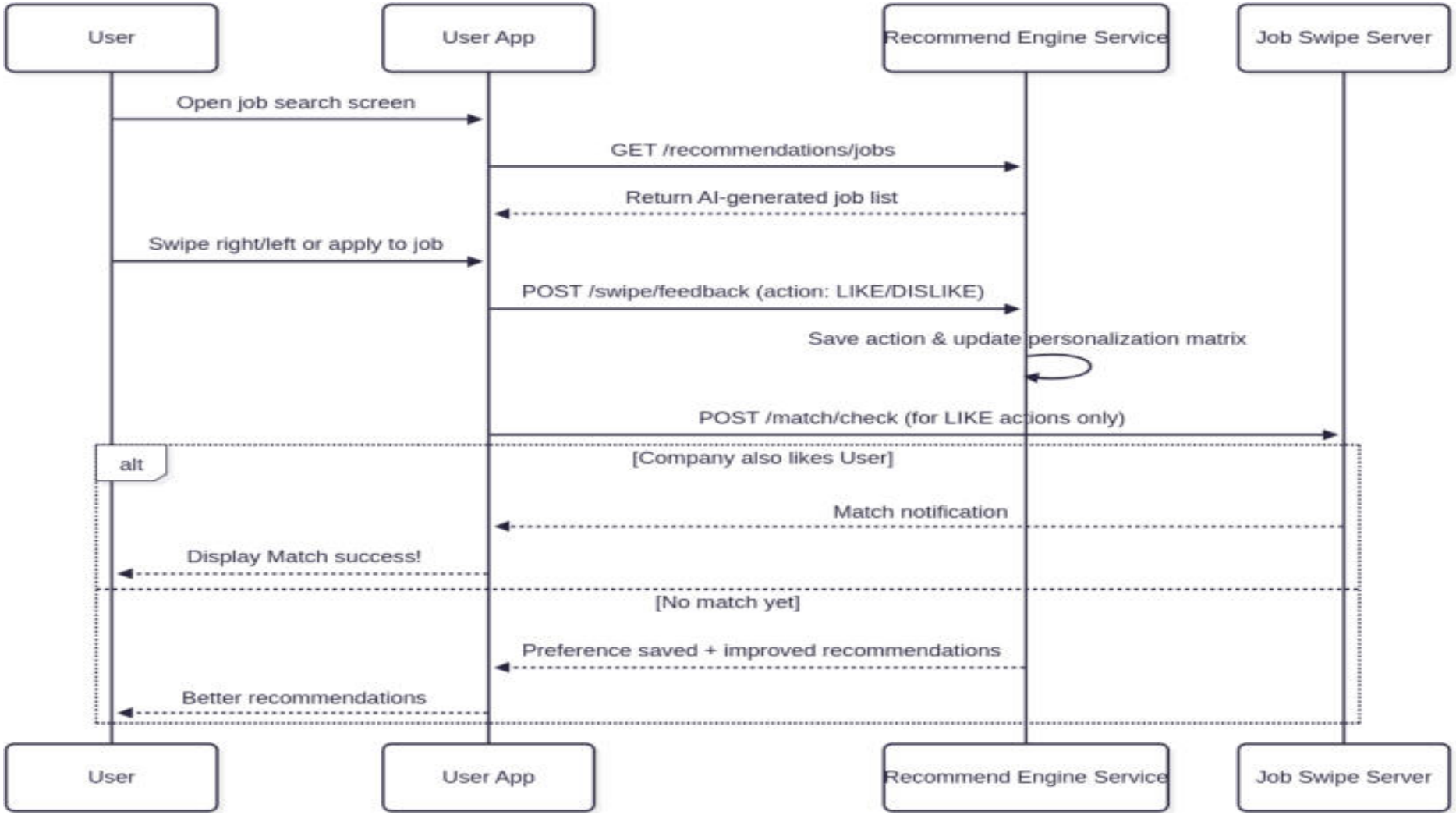

*Hình 2.9: Sơ đồ tuần tự realtime - chat*

### 2.2.4.6. Sơ đồ tuần tự gửi lời mời phỏng vấn



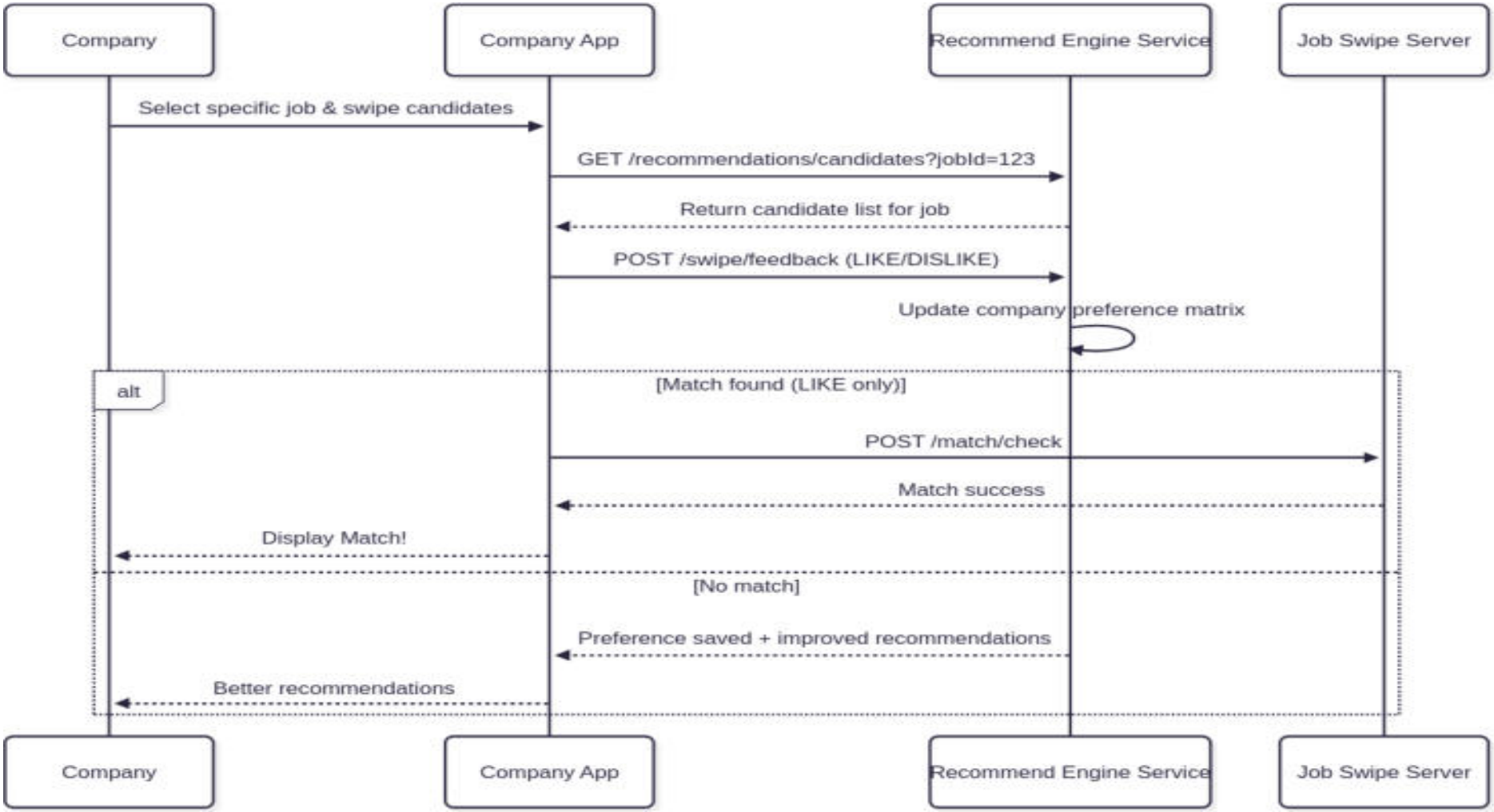

*Hình 2.10: Sơ đồ tuần tự gửi lời mời phỏng vấn*

### 2.2.4.7. Sơ đồ tuần tự quá trình User vuốt chọn job và thực hiện matching




*Hình 2.11: Sơ đồ tuần tự quá trình user vuốt chọn job và thực hiện matching*

### 2.2.4.8. Sơ đồ tuần tự quá trình nhà tuyển dụng vuốt chọn ứng viên và matching




*Hình 2.12: Sơ đồ quá trình nhà tuyển dụng vuốt chọn ứng viên và matching*

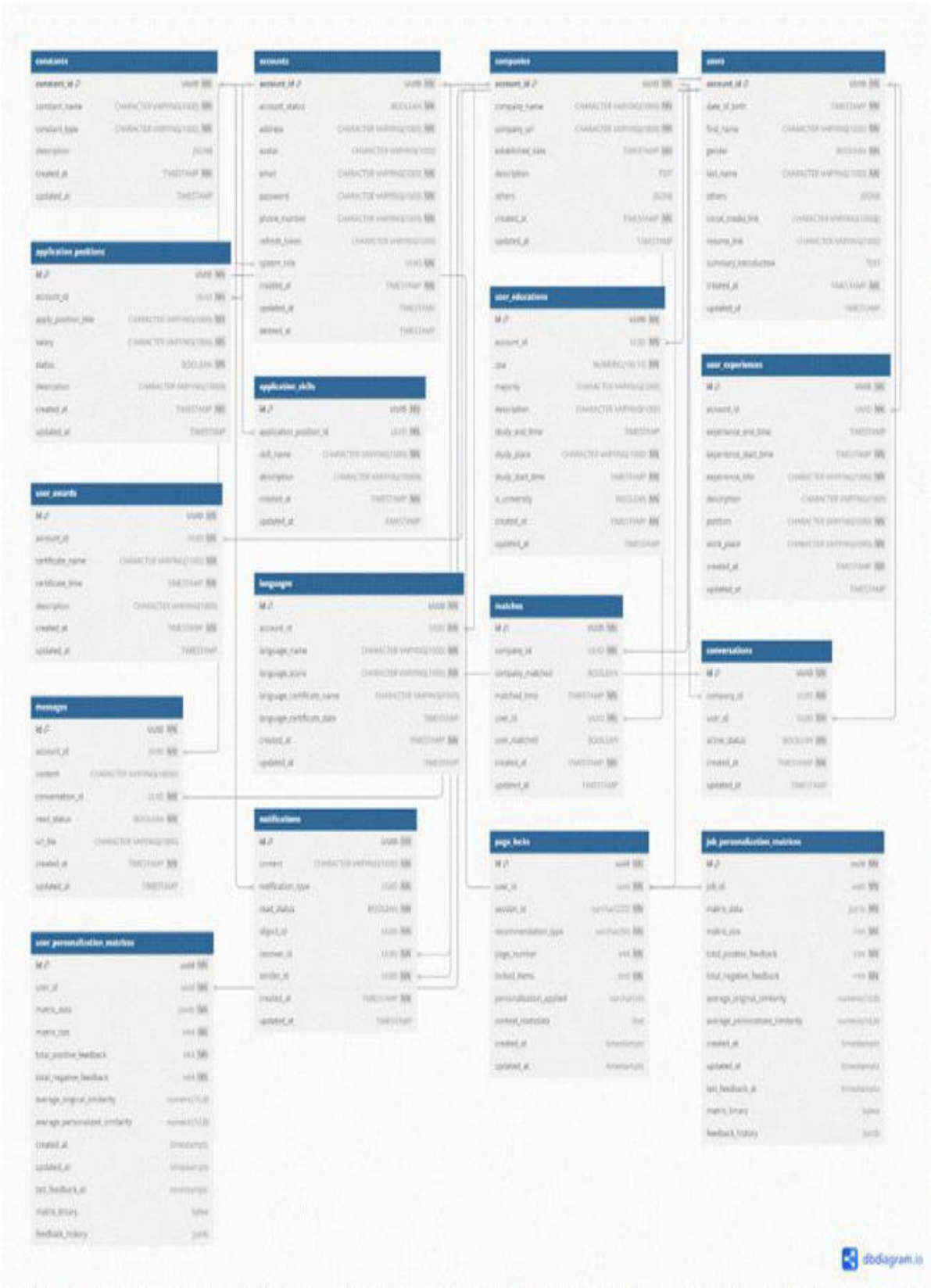
### 2.2.5 Thiết kế cơ sở dữ liệu

#### 2.2.5.1. Mô hình thiết kế cơ sở dữ liệu

Hệ thống Cơ sở dữ liệu bao gồm những bảng sau:

- Bảng constants: lưu thông tin các hằng số hệ thống như vai trò hoặc loại thông báo.
- Bảng accounts: lưu thông tin tài khoản gốc dùng chung cho người dùng và nhà tuyển dụng.
- Bảng companies: lưu thông tin chi tiết về công ty (dành cho nhà tuyển dụng).
- Bảng users: lưu thông tin chi tiết cá nhân của người tìm việc.
- Bảng application_positions: lưu thông tin vị trí công việc mà người dùng ứng tuyển.
- Bảng application_skills: lưu thông tin kỹ năng ứng với từng vị trí ứng tuyển.
- Bảng user_educations: lưu thông tin quá trình học tập của người dùng.
- Bảng user_experiences: lưu thông tin kinh nghiệm làm việc của người dùng.
- Bảng user_awards: lưu thông tin về các chứng chỉ/thành tích của người dùng.
- Bảng languages: lưu thông tin năng lực ngoại ngữ của người dùng.
- Bảng matches: lưu thông tin các kết nối giữa công ty và người tìm việc.
- Bảng notifications: lưu thông báo được gửi giữa các người dùng trong hệ thống.
- Bảng user_personalization_matrices: lưu ma trận cá nhân hoá riêng cho người dùng dựa theo tương tác.
- Bảng job_personalization_matrices: lưu ma trận cá nhân hoá riêng cho công việc dựa theo tương tác.
- Bảng page_locks: lưu trữ thông tin khoá dữ liệu theo từng trang khuyến nghị cho mỗi phiên người dùng, đảm bảo tính nhất quán nội dung khuyến nghị trong quá trình phân trang và tái xếp hạng kết quả.

### 2.2.5.2. Sơ đồ quan hệ giữa các bảng trong cơ sở dữ liệu




*Hình 2.13: Sơ đồ quan hệ giữa các bảng trong cơ sở dữ liệu*

## Mô tả các bảng trong Cơ sở dữ liệu

- Bảng constants

*Bảng 2.4: Bảng cơ sở dữ liệu constants*

| Tên trường | Kiểu dữ liệu | Miêu tả | Chú thích |
|---|---|---|---|
| constant_id | UUID | Mã hằng số | Khoá chính |
| constant_name | CHARACTER VARYING(1000) | Tên hằng số | |
| constant_type | CHARACTER VARYING(1000) | Loại hằng số (SYSTEM_ROLE,...) | |
| description | JSONB | Mô tả chi tiết | |
| created_at | TIMESTAMP WITH TIME ZONE | Thời gian tạo | |
| updated_at | TIMESTAMP WITH TIME ZONE | Thời gian cập nhật | |

- Bảng accounts

*Bảng 2.5: Bảng cơ sở dữ liệu accounts*

| Tên trường | Kiểu dữ liệu | Miêu tả | Chú thích |
|---|---|---|---|
| account_id | UUID | Mã tài khoản | Khoá chính |
| account_status | BOOLEAN | Trạng thái hoạt động | Mặc định TRUE |
| address | CHARACTER VARYING(1000) | Địa chỉ người dùng | |
| avatar | CHARACTER VARYING(1000) | Ảnh đại diện | |
| email | CHARACTER VARYING(1000) | Email người dùng | |
| password | CHARACTER VARYING(1000) | Mật khẩu người dùng | |
| phone_number | CHARACTER VARYING(1000) | Số điện thoại | |
| refresh_token | CHARACTER VARYING(1000) | Token xác thực lại | |
| system_role | UUID | Mã vai trò | Khoá ngoại (constants.constant_id) |
| created_at | TIMESTAMP WITH TIME ZONE | Thời gian tạo tài khoản | |
| updated_at | TIMESTAMP WITH TIME ZONE | Thời gian cập nhật tài khoản | |

– Bảng companies

*Bảng 2.6: Bảng cơ sở dữ liệu nhà tuyển dụng Company*

| Tên trường | Kiểu dữ liệu | Miêu tả | Chú thích |
|---|---|---|---|
| account_id | UUID | Mã tài khoản công ty | Khoá chính, khoá ngoại (accounts. account id) |
| company_name | CHARACTER VARYING(1000) | Tên công ty | |
| company_url | CHARACTER VARYING(1000) | Trang web công ty | |
| established_date | TIMESTAMP WITH TIME ZONE | Ngày thành lập công ty | |
| description | TEXT | Mô tả công ty | |
| others | JSONB | Thông tin khác | |
| created_at | TIMESTAMP WITH TIME ZONE | Thời gian tạo | |
| updated_at | TIMESTAMP WITH TIME ZONE | Thời gian cập nhật | |

– Bảng users

*Bảng 2.7: Bảng cơ sở dữ liệu người dùng Users*

| Tên trường | Kiểu dữ liệu | Miêu tả | Chú thích |
|---|---|---|---|
| account_id | UUID | Mã tài khoản người dùng | Khoá chính, khoá ngoại (accounts.account_id) |
| date_of_birth | TIMESTAMP WITH TIME ZONE | Ngày sinh | |
| first_name | CHARACTER VARYING(1000) | Họ người dùng | |
| last_name | CHARACTER VARYING(1000) | Tên người dùng | |
| gender | BOOLEAN | Giới tính | |
| others | JSONB | Thông tin khác | |
| social_media_link | CHARACTER VARYING(1000)[] | Liên kết mạng xã hội | |
| resume_link | CHARACTER VARYING(1000) | Link CV | |
| summary_introduction | TEXT | Giới thiệu bản thân | |
| created_at | TIMESTAMP WITH TIME ZONE | Thời gian tạo | |
| updated_at | TIMESTAMP WITH TIME ZONE | Thời gian cập nhật | |

– Bảng application_positions

*Bảng 2.8: Bảng cơ sở dữ liệu application_positions*

| Tên trường | Kiểu dữ liệu | Miêu tả | Chú thích |
|---|---|---|---|
| id | UUID | Mã vị trí ứng tuyển | Khoá chính |
| account_id | UUID | Mã tài khoản ứng tuyển | Khoá ngoại |
| apply_position_title | CHARACTER VARYING(1000) | Tên vị trí ứng tuyển | |
| salary | CHARACTER VARYING(1000) | Mức lương | |
| status | BOOLEAN | Trạng thái | |
| description | CHARACTER VARYING(10000) | Mô tả chi tiết | |
| created_at | TIMESTAMP WITH TIME ZONE | Thời gian tạo | |
| updated_at | TIMESTAMP WITH TIME ZONE | Thời gian cập nhật | |

– Bảng user_educations

*Bảng 2.9: Bảng cơ sở dữ liệu user_educations*

| Tên trường | Kiểu dữ liệu | Miêu tả | Chú thích |
|---|---|---|---|
| id | UUID | Mã học vấn | Khoá chính |
| account_id | UUID | Mã người dùng | Khoá ngoại |
| cpa | NUMERIC(100,10) | Điểm trung bình | |
| majority | CHARACTER VARYING(1000) | Chuyên ngành | |
| description | CHARACTER VARYING(1000) | Mô tả chi tiết | |
| study_end_time | TIMESTAMP WITH TIME ZONE | Thời gian kết thúc | |
| study_place | CHARACTER VARYING(1000) | Nơi học | |
| study_start_time | TIMESTAMP WITH TIME ZONE | Thời gian bắt đầu | |
| is_university | BOOLEAN | Có phải đại học không | Mặc định TRUE |
| created_at | TIMESTAMP WITH TIME ZONE | Thời gian tạo | |
| updated_at | TIMESTAMP WITH TIME ZONE | Thời gian cập nhật | |

– Bảng user_experiences

*Bảng 2.10: Bảng cơ sở dữ liệu user_experiences*

| Tên trường | Kiểu dữ liệu | Miêu tả | Chú thích |
|---|---|---|---|
| id | UUID | Mã kinh nghiệm | Khoá chính |
| account_id | UUID | Mã người dùng | Khoá ngoại |
| experience_end_time | TIMESTAMP WITH TIME ZONE | Thời gian kết thúc | |
| experience_start_time | TIMESTAMP WITH TIME ZONE | Thời gian bắt đầu | |
| experience_title | CHARACTER VARYING(1000) | Tên kinh nghiệm | |
| description | CHARACTER VARYING(1000) | Mô tả chi tiết | |
| position | CHARACTER VARYING(1000) | Chức danh | |
| work_place | CHARACTER VARYING(1000) | Nơi làm việc | |
| created_at | TIMESTAMP WITH TIME ZONE | Thời gian tạo | |
| updated_at | TIMESTAMP WITH TIME ZONE | Thời gian cập nhật | |

– Bảng user_awards

*Bảng 2.11: Bảng cơ sở dữ liệu user_awards*

| Tên trường | Kiểu dữ liệu | Miêu tả | Chú thích |
|---|---|---|---|
| id | UUID | Mã chứng nhận | Khoá chính |
| account_id | UUID | Mã người dùng | Khoá ngoại |
| certificate_name | CHARACTER VARYING(1000) | Tên chứng nhận | |
| certificate_time | TIMESTAMP WITH TIME ZONE | Thời gian nhận chứng nhận | |
| description | CHARACTER VARYING(1000) | Mô tả | |
| created_at | TIMESTAMP WITH TIME ZONE | Thời gian tạo | |
| updated_at | TIMESTAMP WITH TIME ZONE | Thời gian cập nhật | |

*Bảng 2.12: Bảng cơ sở dữ liệu languages*

| Tên trường | Kiểu dữ liệu | Miêu tả | Chú thích |
|---|---|---|---|
| id | UUID | Mã ngôn ngữ | Khoá chính |
| account_id | UUID | Mã người dùng | Khoá ngoại |
| language_name | CHARACTER VARYING(1000) | Tên ngôn ngữ | |
| language_score | CHARACTER VARYING(1000) | Trình độ/ngưỡng điểm | |
| language_certificate_name | CHARACTER VARYING(1000) | Tên chứng chỉ ngoại ngữ | |
| language_certificate_date | TIMESTAMP WITH TIME ZONE | Ngày cấp chứng chỉ | |
| created_at | TIMESTAMP WITH TIME ZONE | Thời gian tạo | |
| updated_at | TIMESTAMP WITH TIME ZONE | Thời gian cập nhật | |

– Bảng matches

*Bảng 2.13: Bảng cơ sở dữ liệu matches*

| Tên trường | Kiểu dữ liệu | Miêu tả | Chú thích |
|---|---|---|---|
| id | UUID | Mã kết nối | Khoá chính |
| company_id | UUID | Mã công ty | Khoá ngoại (companies.account_id) |
| company_matched | BOOLEAN | Công ty đã chấp nhận kết nối | |
| matched_time | TIMESTAMP WITH TIME ZONE | Thời gian kết nối thành công | |
| user_id | UUID | Mã người dùng | Khoá ngoại (users.account_id) |
| user_matched | BOOLEAN | Người dùng đã chấp nhận kết nối | |
| created_at | TIMESTAMP WITH TIME ZONE | Thời gian tạo | |
| updated_at | TIMESTAMP WITH TIME ZONE | Thời gian cập nhật | |

– Bảng notifications

*Bảng 2.14: Bảng cơ sở dữ liệu notifications*

| Tên trường | Kiểu dữ liệu | Miêu tả | Chú thích |
|---|---|---|---|
| id | UUID | Mã thông báo | Khoá chính |
| content | CHARACTER VARYING(1000) | Nội dung thông báo | |
| notification_type | UUID | Loại thông báo | Khoá ngoại (constants.constant_id) |
| read_status | BOOLEAN | Trạng thái đọc | Mặc định FALSE |
| object_id | UUID | Đối tượng liên quan | |
| receiver_id | UUID | Người nhận | Khoá ngoại (accounts.account_id) |
| sender_id | UUID | Người gửi | Khoá ngoại (accounts.account_id) |
| created_at | TIMESTAMP WITH TIME ZONE | Thời gian tạo | |
| updated_at | TIMESTAMP WITH TIME ZONE | Thời gian cập nhật | |

– Bảng user_personalization_matrices

*Bảng 2.15: Bảng cơ sở dữ liệu user_personalization_matrices*

| Tên trường | Kiểu dữ liệu | Miêu tả | Chú thích |
|---|---|---|---|
| id | UUID | Mã ma trận cá nhân hoá của người dùng | Khoá chính |
| user_id | UUID | Mã người dùng | Khoá ngoại |
| matrix_data | JSONB | Dữ liệu ma trận ở dạng JSON | Bắt buộc |
| matrix_size | INT4 | Kích thước ma trận (số chiều) | |
| total_positive_feedback | INT4 | Số lượt phản hồi tích cực | Mặc định 0 |
| total_negative_feedback | INT4 | Số lượt phản hồi tiêu cực | Mặc định 0 |
| average_original_similarity | NUMERIC(10,8) | Độ tương đồng gốc trung bình | |
| average_personalized_similarity | NUMERIC(10,8) | Độ tương đồng cá nhân hoá trung bình | |
| created_at | TIMESTAMP WITH TIME ZONE | Thời gian tạo dữ liệu | Mặc định now() |
| updated_at | TIMESTAMP WITH TIME ZONE | Thời gian cập nhật | Mặc định now() |
| last_feedback_at | TIMESTAMP WITH TIME ZONE | Lần cuối phản hồi người dùng | |
| matrix_binary | BYTEA | Ma trận dạng nhị phân | |
| feedback_history | JSONB | Lịch sử phản hồi (liked/disliked jobs/users) | Mặc định cấu trúc JSON rỗng |

– Bảng job_personalization_matrices

*Bảng 2.16: Bảng cơ sở dữ liệu job_personalization_matrices*

| Tên trường | Kiểu dữ liệu | Miêu tả | Chú thích |
|---|---|---|---|
| id | UUID | Mã ma trận cá nhân hoá của công việc | Khoá chính |
| job_id | UUID | Mã công việc | Khoá ngoại |
| matrix_data | JSONB | Dữ liệu ma trận ở dạng JSON | Bắt buộc |
| matrix_size | INT4 | Kích thước ma trận (số chiều) | |
| total_positive_feedback | INT4 | Số lượt phản hồi tích cực | Mặc định 0 |
| total_negative_feedback | INT4 | Số lượt phản hồi tiêu cực | Mặc định 0 |
| average_original_similarity | NUMERIC(10,8) | Độ tương đồng gốc trung bình | |
| average_personalized_similarity | NUMERIC(10,8) | Độ tương đồng cá nhân hoá trung bình | |
| created_at | TIMESTAMP WITH TIME ZONE | Thời gian tạo dữ liệu | Mặc định now() |
| updated_at | TIMESTAMP WITH TIME ZONE | Thời gian cập nhật | Mặc định now() |
| last_feedback_at | TIMESTAMP WITH TIME ZONE | Lần cuối phản hồi người dùng | |
| matrix_binary | BYTEA | Ma trận dạng nhị phân | |
| feedback_history | JSONB | Lịch sử phản hồi (liked/disliked jobs/users) | Mặc định cấu trúc JSON rỗng |

– Bảng page_locks

*Bảng 2.17: Bảng cơ sở dữ liệu page_locks*

| Tên trường | Kiểu dữ liệu | Miêu tả | Chú thích |
|---|---|---|---|
| id | UUID | Mã định danh khoá trang | Khoá chính |
| user_id | UUID | Mã người dùng | Khoá ngoại tới accounts.account_id |
| session_id | VARCHAR(255) | Mã phiên làm việc của người dùng | Bắt buộc |
| recommendation_type | VARCHAR(50) | Loại khuyến nghị (ví dụ: 'job', 'user') | Bắt buộc |
| page_number | INT4 | Số thứ tự của trang | Bắt buộc |
| locked_items | TEXT[] | Danh sách các mục (item) được hiển thị và khóa trong trang đó | Mặc định là mảng rỗng |
| personalization_applied | VARCHAR(50) | Trạng thái cá nhân hoá áp dụng trên trang | Tuỳ chọn |
| context_metadata | TEXT | Dữ liệu ngữ cảnh (dạng chuỗi JSON) | Tuỳ chọn |
| created_at | TIMESTAMP WITH TIME ZONE | Thời gian tạo bản ghi | Mặc định là CURRENT_TIMESTAMP |
| updated_at | TIMESTAMP WITH TIME ZONE | Thời gian cập nhật bản ghi | Mặc định là CURRENT_TIMESTAMP, tự động cập nhật |

## 2.3. Kết luận chương 2

Chương này đã trình bày rõ về hướng phân tích và thiết kế hệ thống cũng như thể hiện rõ các yêu cầu chức năng và phi chức năng của hệ thống. Ngoài ra, em cũng đã sử dụng các công cụ để vẽ các sơ đồ như Whimsical, Diagram.io, … để có thể vẽ các Use Case, sơ đồ quan trọng để mô tả hệ thống.

Bên cạnh đó, việc mô tả cấu trúc hệ thống thông qua sơ đồ cũng như trình tự hành động cho từng chức năng sẽ giúp người dùng có một cái nhìn tổng quan về cách hoạt động và cách sử dụng hệ thống.

# CHƯƠNG 3: PHÁT TRIỂN HỆ THỐNG

## 3.1. Thiết lập môi trường để phát triển hệ thống

### *3.1.1. Quản lý mã nguồn*


*Hình 3.1: Logo github*

Trong quy trình phát triển phần mềm hiện đại, việc quản lý mã nguồn và kiểm soát phiên bản đóng vai trò rất quan trọng để đảm bảo tính nhất quán, dễ theo dõi và hỗ trợ làm việc nhóm hiệu quả. Đối với đồ án JobSwipe, nhóm phát triển đã sử dụng Git kết hợp với nền tảng lưu trữ GitHub để thực hiện các tác vụ này.

Git là hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS) phổ biến nhất hiện nay. Git cho phép theo dõi mọi thay đổi của mã nguồn theo thời gian, giúp lập trình viên có thể quay lại các phiên bản trước hoặc tạo nhiều nhánh (branch) để phát triển song song.

GitHub là dịch vụ lưu trữ mã nguồn dựa trên Git, hỗ trợ quản lý kho mã nguồn từ xa, đồng thời cung cấp nhiều công cụ hỗ trợ cộng tác như: pull request, issue tracker, actions CI/CD,... Việc sử dụng GitHub giúp nhóm dễ dàng chia sẻ mã nguồn, phân công công việc và kiểm soát tiến độ dự án.

Một số thao tác chính đã được sử dụng trong dự án:

- git clone: Tải mã nguồn từ GitHub về máy cục bộ.
- git checkout -b <branch>: Tạo và chuyển sang nhánh mới để phát triển tính năng riêng biệt.
- git add, git commit: Ghi lại các thay đổi vào lịch sử phiên bản.
- git push, git pull: Đồng bộ mã nguồn giữa máy cục bộ và GitHub.
- Pull Request (PR): Được sử dụng để kiểm tra, xem xét và hợp nhất mã từ các nhánh feature vào nhánh chính.

Việc áp dụng Git + GitHub đã giúp đảm bảo quy trình làm việc mạch lạc, có thể kiểm soát xung đột mã nguồn, theo dõi lịch sử thay đổi rõ ràng và hỗ trợ làm việc nhóm hiệu quả.

Git là hệ thống quản lý phiên bản phân tán (VCS) cho phép theo dõi và kiểm soát thay đổi trong mã nguồn hiệu quả.

### *3.1.2. Môi trường phát triển*

Hệ thống JobSwipe được phát triển trên nhiều thành phần, phân chia theo chức năng:

- Frontend: Ứng dụng di động phát triển bằng Flutter, hỗ trợ đa nền tảng (Android/iOS), cung cấp trải nghiệm người dùng hiện đại.
- Backend chính: Xây dựng bằng Java Spring Boot, đóng vai trò là Web API xử lý logic nghiệp vụ và giao tiếp với cơ sở dữ liệu.
- Recommendation API: Một server riêng sử dụng Python Flask, phục vụ các tính năng gợi ý công việc/ứng viên.
- Cơ sở dữ liệu: Hệ quản trị cơ sở dữ liệu sử dụng PostgreSQL.
- Bộ nhớ tạm: Hệ thống sử dụng Redis để cache dữ liệu hiệu năng cao.
- ChromaDB: Được sử dụng làm vector store cho hệ thống gợi ý, giúp lưu trữ và truy vấn embedding hiệu quả.

**Flutter**


*Hình 3.2: Logo của flutter*

**Java**


*Hình 3.3: Logo của Java*

**Spring boot**


*Hình 3.4: Log của spring boot framework*

**PostgreSQL**


*Hình 3.5: Hệ quản trị cơ sở dữ liệu PostgreSQL logo*

**Redis**


*Hình 3.6: Redis logo*

**ChromaDB**


*Hình 3.7: ChromaDB logo*

### 3.1.3. Thiết lập môi trường để phát triển Frontend

1. Khởi tạo Flutter project:
   - *flutter create jobswipe_app*
   - *cd jobswipe_app*
2. Cài đặt các dependency cần thiết: Sử dụng pubspec.yaml để thêm các thư viện hỗ trợ UI, HTTP, Firebase,...
3. Chạy ứng dụng
   - *flutter run*
4. Tự động hot reload: Flutter hỗ trợ hot reload giúp kiểm tra nhanh giao diện khi thay đổi mã nguồn.

### 3.1.4. Thiết lập môi trường để phát triển Backend

1. Tạo project Spring Boot: Sử dụng Spring Initializr (https://start.spring.io) để tạo project với các dependency như: Spring Web, Spring Data JPA, PostgreSQL Driver,…
2. Cấu hình kết nối PostgreSQL
   - *spring.datasource.url=jdbc:postgresql://localhost:5432/jobswipe_db*
   - *spring.datasource.username=postgres*
   - *spring.datasource.password=*******
   - *spring.jpa.hibernate.ddl-auto=update*
3. Khởi chạy backend
   - *> ./mvnw spring-boot:run*

### 3.1.5. Thiết lập môi trường Flask API (Recommendation System)

1. Tạo môi trường Python
   - *> python -m venv venv*
   - *> source venv/bin/activate*
   - *> pip install flask redis chromadb*
2. Khởi tạo Flask app
   - *from flask import Flask*
   - *app = Flask(__name__)*
   - 
   - *@app.route("/api/recommend")*
   - *def recommend():*
   - *    return {"result": "recommendations"}*
   - *if __name__ == "__main__":*

- *app.run(debug=True)*

3. Tích hợp Redis và ChromaDB cho việc lưu trữ trạng thái và embedding vector.

### *3.1.6. Thiết lập cơ sở dữ liệu PostgreSQL*

1. Cài đặt PostgreSQL: Có thể cài từ https://www.postgresql.org/ hoặc sử dụng Docker.
2. Tạo cơ sở dữ liệu:
   - *CREATE DATABASE jobswipe_db;*
3. Cấu hình trong Spring Boot
4. Quản lý schema

- Sử dụng JPA + Hibernate để tự động tạo bảng từ các entity Java.
- Có thể kết hợp thêm Flyway hoặc Liquibase nếu cần quản lý migration.

## 3.2. Ứng dụng học máy ở trong chương trình

### *3.2.1. Phát biểu bài toán*

Mô tả đầu vào (Input) và đầu ra (Output):

**Input:**

- Thông tin người dùng: Dữ liệu hồ sơ cá nhân bao gồm kinh nghiệm làm việc, học vấn, kỹ năng, ngôn ngữ, giải thưởng và mô tả bản thân
- Thông tin công việc: Mô tả công việc, yêu cầu kỹ năng, vị trí ứng tuyển, mức lương và thông tin công ty
- Feedback tương tác: Dữ liệu swipe (like/dislike) của người dùng đối với các công việc được đề xuất
- Truy vấn tìm kiếm: Câu hỏi hoặc yêu cầu tìm kiếm công việc từ người dùng

**Output:**

- Danh sách công việc được đề xuất: Được sắp xếp theo độ phù hợp và cá nhân hóa cho từng người dùng
- Điểm số tương tự (Similarity Score): Thể hiện mức độ phù hợp giữa người dùng và công việc (0-1)
- Ma trận cá nhân hóa: Ma trận 256x256 được cập nhật theo thời gian để phản ánh sở thích cá nhân

**Trường hợp xử lý:**

- Người dùng mới: Sử dụng vector embeddings cơ bản từ thông tin hồ sơ để tìm kiếm tương tự
- Người dùng có feedback: Sử dụng ma trận cá nhân hóa để biến đổi vector embeddings và cải thiện độ chính xác
- Không tìm thấy kết quả phù hợp: Trả về thông báo đề xuất tiêu chí tìm kiếm

### *3.2.2. Dataset*

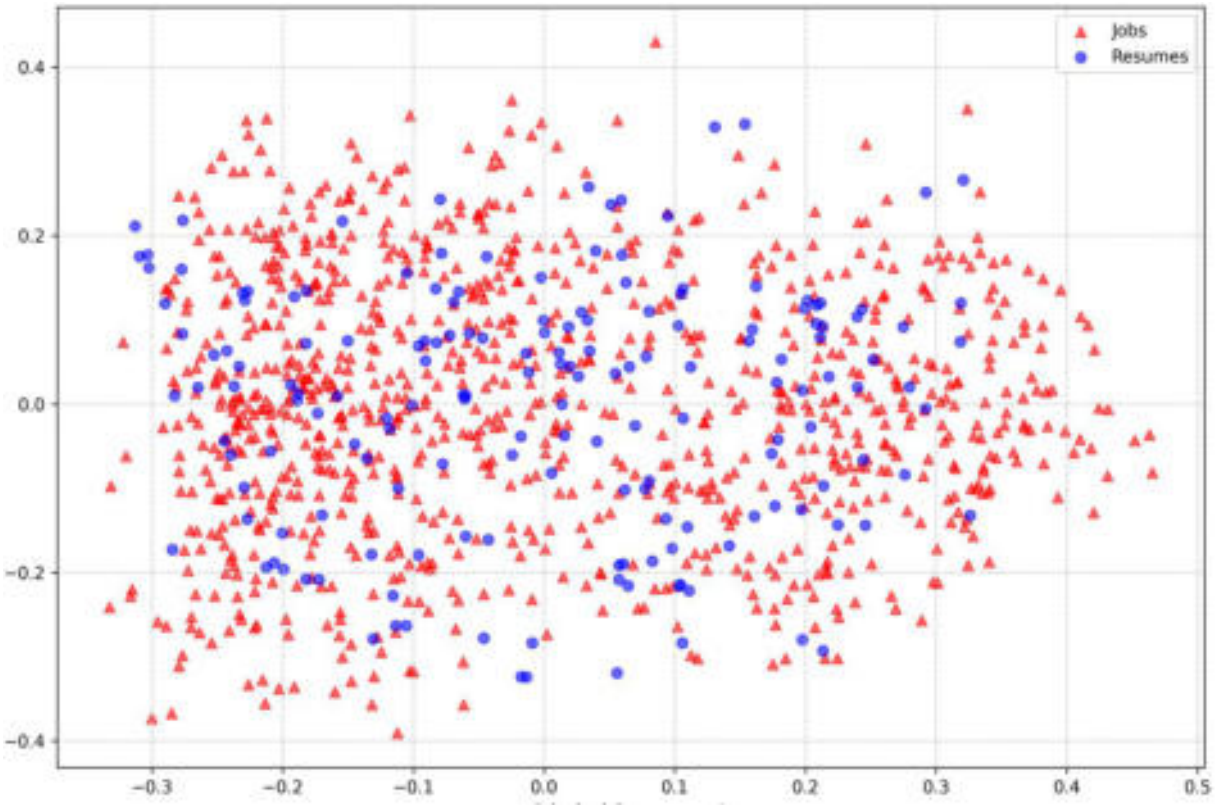
Data ở đây là các vector embedding được tạo từ profile, resume của user (User Embedding Vector), các vector embedding được tạo từ thông tin mô tả công việc (Job Embedding Vector) và các feedback được gửi trực tiếp từ người dùng ứng dụng.

**User Embedding Vector:**

- Dimensionality: 256 chiều
- Nguồn dữ liệu: Thông tin hồ sơ, kinh nghiệm, kỹ năng
- Số lượng: ~200 user embeddings
- Format: Float32 arrays, normalized vectors

**Job Embedding Vector:**

- Dimensionality: 256 chiều
- Nguồn dữ liệu: Mô tả công việc, yêu cầu, thông tin công ty
- Số lượng: ~1,000 job embeddings
- Format: Float32 arrays, normalized vectors




*Hình 3.8: Visualization vector jov và vector user ở dạng 2D*

**Feedback:**

Hệ thống thu thập feedback từ người dùng thông qua các thao tác tương tác trực tiếp trên giao diện ứng dụng di động. Cơ chế feedback được thiết kế dựa trên mô hình swipe quen thuộc, cho phép người dùng thể hiện sở thích một cách tự nhiên và nhanh chóng. Feedback được thu thập thông qua hai loại thao tác chính:

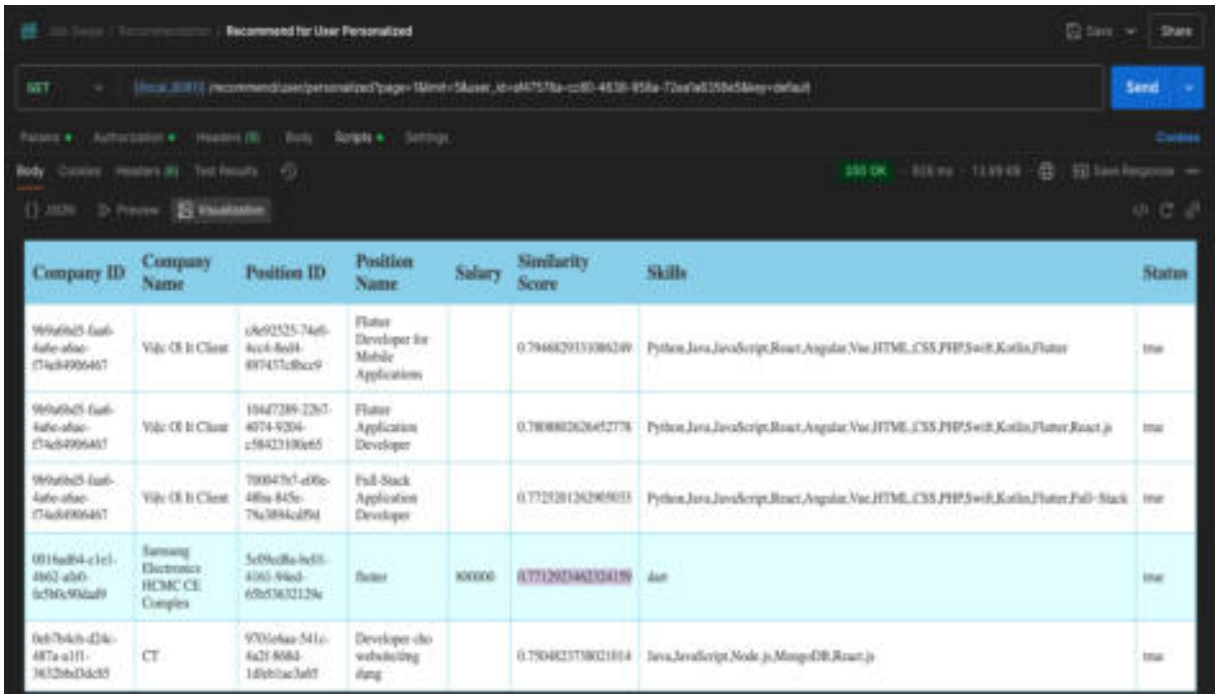
- **Positive Feedback (Feedback = +1)**: Khi người dùng thực hiện thao tác swipe phải hoặc nhấn nút "Ứng tuyển" đối với một job posting, hệ thống sẽ ghi nhận đây là tín hiệu tích cực, cho thấy người dùng quan tâm và có xu hướng ứng tuyển vào vị trí công việc này.
- **Negative Feedback (Feedback = -1)**: Khi người dùng thực hiện thao tác swipe trái hoặc nhấn nút "Từ chối/Bỏ qua" đối với một lời đề nghị match, hệ thống sẽ ghi nhận đây là tín hiệu tiêu cực, cho thấy người dùng không quan tâm hoặc không phù hợp với vị trí công việc này.

**Personalization Matrix:**

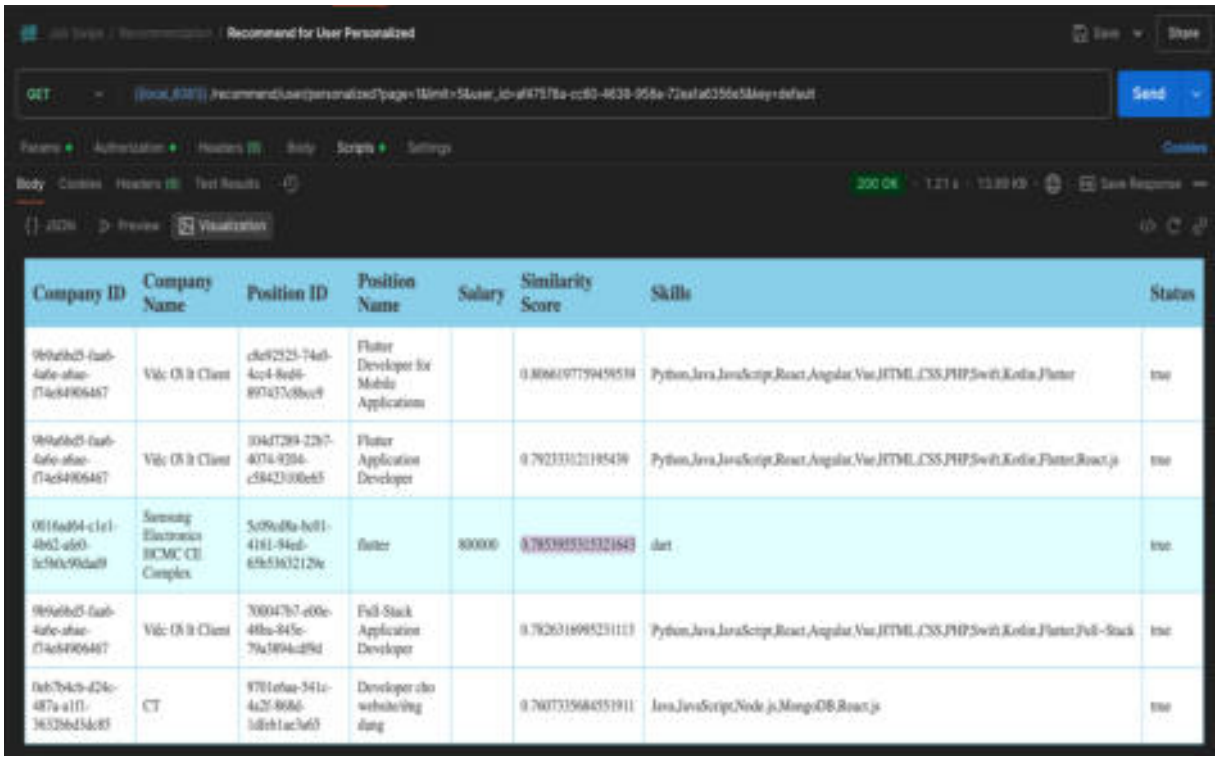
- Kích thước: 256x256 cho mỗi người dùng
- Khởi tạo: Identity matrix (I) cho người dùng mới
- Lưu trữ: Binary compressed format trong PostgreSQL bằng kiểu ByteA

| PERSONALIZATION_MATRICES | | |
|---|---|---|
| uuid | id | PK |
| uuid | user_id | |
| jsonb | matrix_data | |
| integer | matrix_size | |
| timestamp | created_at | |
| timestamp | updated_at | |
| bytea | matrix_binary | |
| jsonb | feedback_history | |

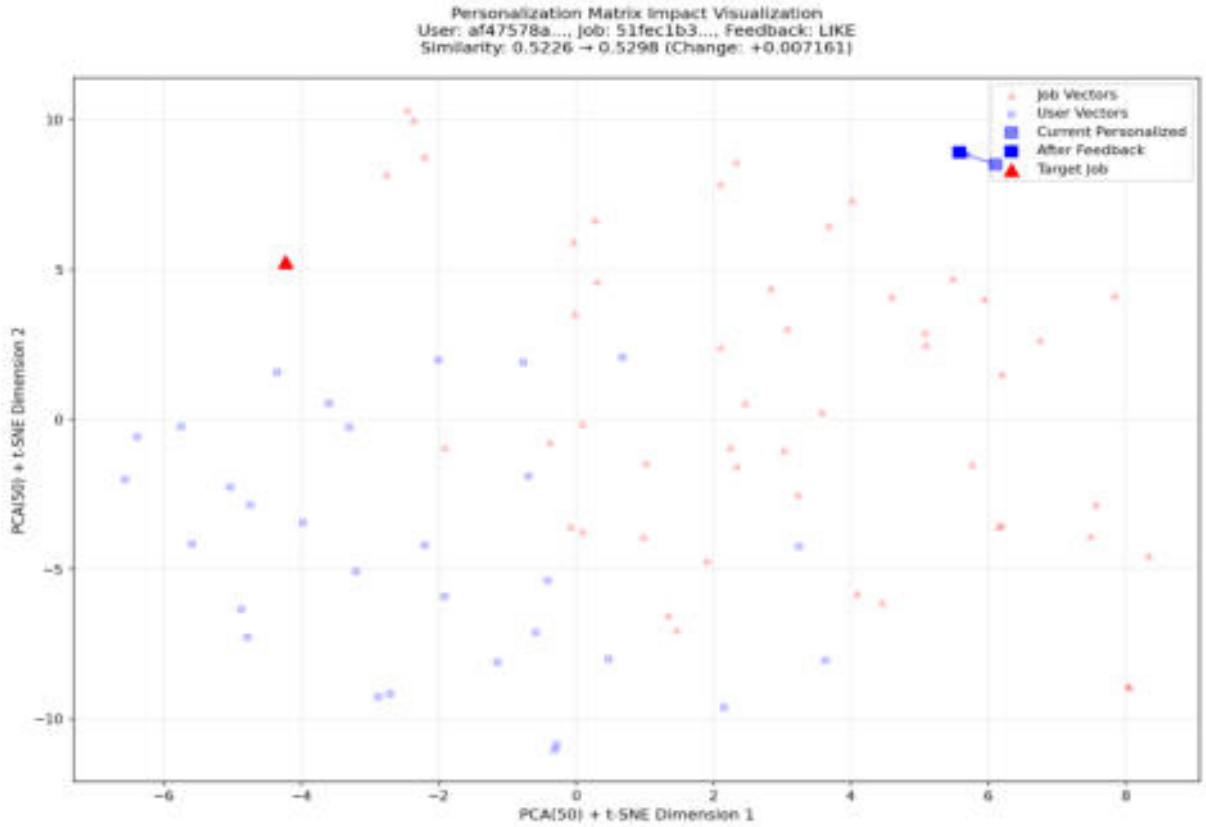
*Hình 3.9: Cấu trúc bảng để lưu ma trận cá nhân hóa*



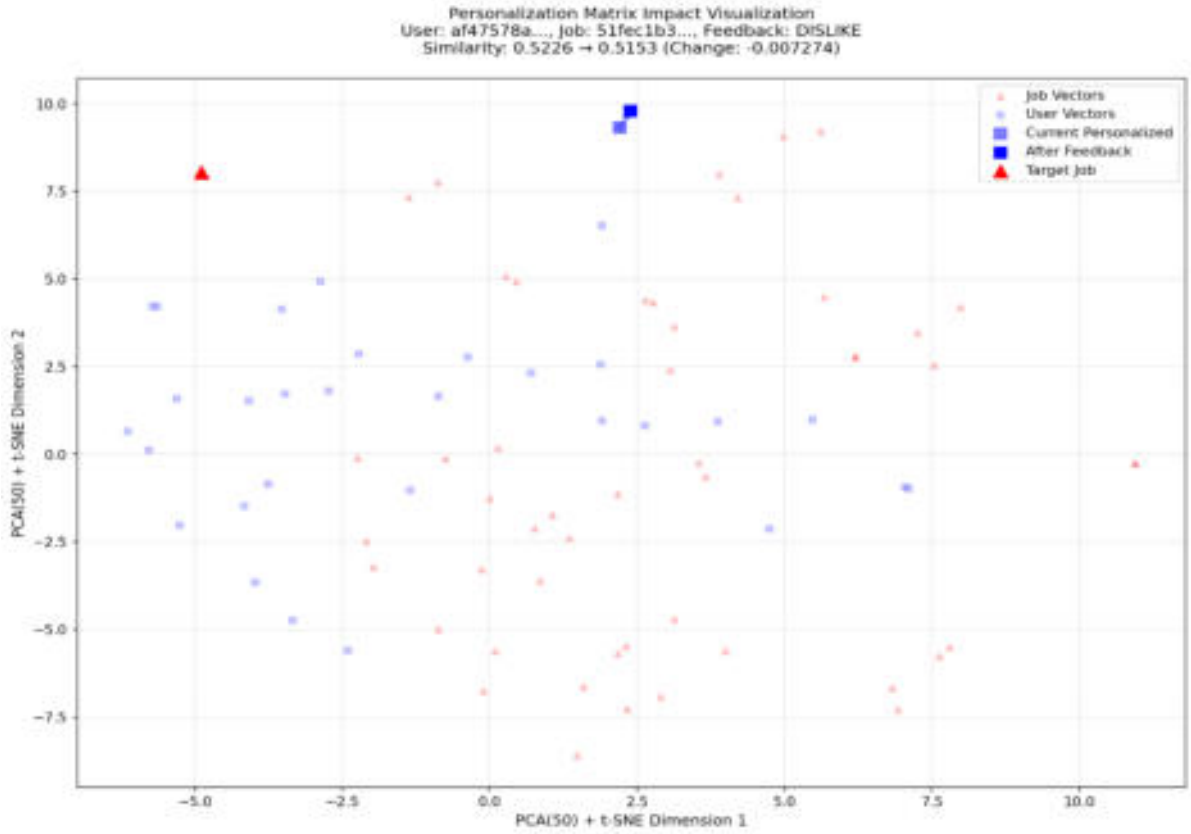

*Hình 3.10: Các vector được gợi ý trước khi feedback*




*Hình 3.11: Các vector được gợi ý sau khi feedback*




*Hình 3.12: Visualize vector di chuyển sau khi feedback like dạng 2D*




*Hình 3.13: Visualize vector di chuyển sau khi feedback dislike dạng 2D*

### 3.2.3. Quy trình tạo và cập nhật vector embeddings

Hệ thống gợi ý việc làm sử dụng một kiến trúc học máy kết hợp vector embedding và ma trận cá nhân hóa để cung cấp các gợi ý chính xác và được cá nhân hóa.

1. Chuẩn bị dữ liệu đầu vào từ thông tin của user, thông tin của công việc và công ty tương ứng.
2. Sử dụng text-embedding-ada-002 để tạo vector embedding với kích thước 256 chiều để có thể cân bằng giữa độ chính xác và tốc độ xử lý.
3. Chuẩn hóa và lưu trữ vector embedding vào trong ChromaDB.
4. Nếu người dùng cuối cập nhật thông tin thì hệ thống sẽ chạy lại quy trình tạo vector embedding và ghi đè lên vector cũ trong ChromaDB.

### 3.2.4. Quy trình tạo ma trận cá nhân hóa dựa trên feedback

1. **Khởi tạo ma trận cá nhân hóa**: Khi một người dùng hoặc công việc mới được tạo embedding, hệ thống tự động khởi tạo một ma trận cá nhân hóa dưới dạng ma trận đơn vị để có thể đảm bảo được hệ thống hoạt động bình thường ngay cả khi chưa có feedback vì $I \cdot \boldsymbol{v} = \boldsymbol{v}$ sẽ không làm thay đổi vector embedding gốc. Khởi tạo ma trận cá nhân hóa ban đầu sẽ có dạng:

   $$P_0 = I_{d \times d}$$

   $P_0$: Ma trận cá nhân hóa ban đầu

   $I$: Ma trận đơn vị

   d = 1536$: Kích thước embedding
2. **Lưu trữ ma trận cá nhân hóa:** Lưu trữ ma trận cá nhân hóa vào trong database PostgresSQL bằng cách nén dữ liệu từ numpy array thành dạng BYTEA giúp giảm đáng kể dung lượng lưu trữ
3. **Sử dụng ma trận cá nhân hóa:**

   Thay vì sử dụng cosine similarity truyền thống:

   $$\text{sim}(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{|\boldsymbol{u}||\boldsymbol{v}|}$$

   Hệ thống sử dụng Personalized Quadratic Form:

   $$\text{sim}_P(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u}^T P \boldsymbol{v}}{|\boldsymbol{u}||\boldsymbol{v}|}$$

   $\boldsymbol{u}$ : User embedding

   $\boldsymbol{v}$: Job embedding

   $P$: Ma trận cá nhân hóa của user

   $\text{sim}_P$: Similarity score được cá nhân hóa

4. **Cập nhật ma trận cá nhân hóa:**
   Hệ thống sử dụng gradient descent với regularization để cập nhật ma trận dựa trên feedback với công thức cập nhật:

$$P_{t+1} = P_t + \alpha \cdot f \cdot (\boldsymbol{v}\boldsymbol{v}^{\boldsymbol{T}}) + \beta(I - P_t)$$

   $P_t$: Ma trận cá nhân hóa tại thời điểm $t$
   $\alpha$: Learning rate (0.02)
   $f$: Feedback score (+1 cho like, -1 cho dislike)
   $\boldsymbol{v}$: Item embedding được chuẩn hóa
   $\boldsymbol{v}\boldsymbol{v}^{\boldsymbol{T}}$: Outer product tạo ma trận cập nhật
   $\beta$: Shrinkage coefficient (0.05)
   $I$: Ma trận đơn vị
5. **Ứng dụng trong recommendation system:**
   - ***Kiến trúc hệ thống gợi ý hai chiều với personalization:***
     + Hệ thống hỗ trợ hai loại recommendation chính: gợi ý công việc cho người tìm việc và gợi ý ứng viên cho nhà tuyển dụng.
     + Thay vì sử dụng cosine similarity truyền thống, hệ thống áp dụng Personalized Quadratic Form kết hợp user/job embedding với ma trận cá nhân hóa để tính toán điểm số similarity đã được cá nhân hóa.
     + Hệ thống cũng tích hợp cơ chế Page Locking để đảm bảo tính nhất quán trong phân trang khi ma trận được cập nhật real-time.
   - ***Thu thập feedback và cập nhật ma trận cá nhân hóa:***
     + Hệ thống thu thập feedback thông qua hai cơ chế: feedback trực tiếp (like/dislike ngay lập tức) và swipe feedback (xử lý theo batch).
     + Khi nhận feedback, ma trận cá nhân hóa được cập nhật theo công thức gradient descent với regularization, trong đó learning rate α=0.02 và shrinkage coefficient β=0.05 để cân bằng giữa việc học từ feedback mới và duy trì tính ổn định.
     + Hệ thống lưu trữ lịch sử feedback và cập nhật ma trận real-time để cải thiện chất lượng recommendation theo thời gian.
   - **Tối ưu hiệu suất và kết quả recommendation:**
     + Để xử lý khối lượng lớn người dùng, hệ thống áp dụng nhiều kỹ thuật tối ưu: caching ma trận và embedding, lưu trữ ma trận dưới dạng binary compressed trong PostgreSQL, xử lý feedback theo batch, và tối ưu truy vấn ChromaDB.

+ Kết quả cuối cùng là danh sách recommendations được sắp xếp theo điểm số đã cá nhân hóa, bao gồm thông tin chi tiết về công việc/ứng viên, điểm số similarity gốc và đã personalized, cùng metadata về pagination và session management.

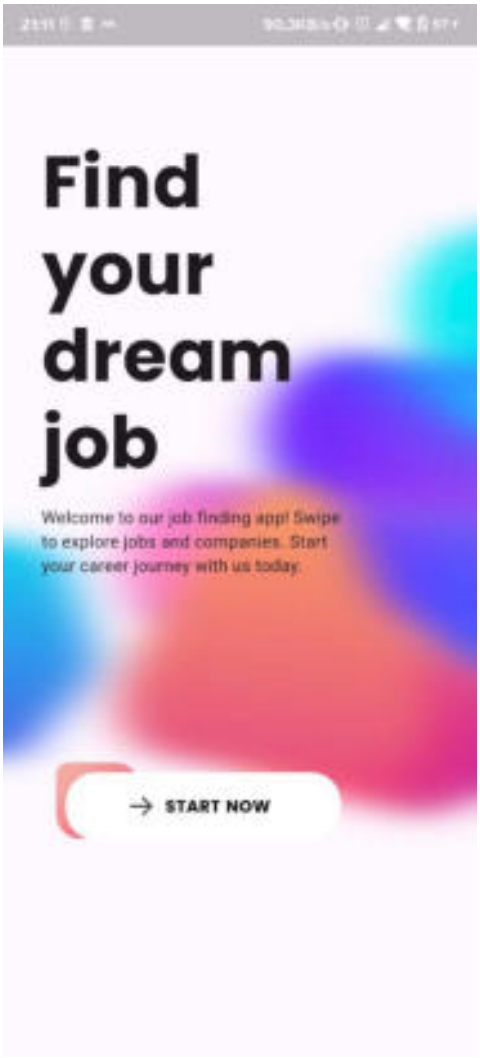
## *3.3.* Kết luận chương 3

Chương 3 em đã trình bày chi tiết quá trình triển khai và phát triển hệ thống JobSwipe, biến các thiết kế và phân tích thành một sản phẩm phần mềm hoàn chỉnh. Quá trình này bắt đầu bằng việc thiết lập một môi trường phát triển toàn diện, bao gồm việc quản lý mã nguồn bằng Git/GitHub, xây dựng giao diện người dung với Flutter, phát triển logic xử lý phía máy chủ bằng Spring Boot, và triển khai dịch vụ đề xuất thông minh với Python và Flask. Bên cạnh đó, các hệ cơ sở dữ liệu như PostgreSQL, Redis và ChromaDB cũng được cấu hình để đảm bảo lưu trữ và truy xuất dữ liệu hiệu quả. Trọng tâm của chương là phần ứng dụng học máy, nơi mô tả cách hệ thống biểu diễn người dùng và công việc dưới dạng các vector embedding. Điểm nổi bật là việc xây dựng và cập nhật ma trận cá nhân hóa dựa trên phản hồi (feedback) trực tiếp từ người dùng. Cơ chế này cho phép hệ thống học từ sở thích của người dùng, từ đó tinh chỉnh các gợi ý để ngày càng chính xác và phù hợp hơn.

# CHƯƠNG 4: TRIỂN KHAI HỆ THỐNG VÀ KẾT QUẢ

## 4.1. Kết quả hình ảnh

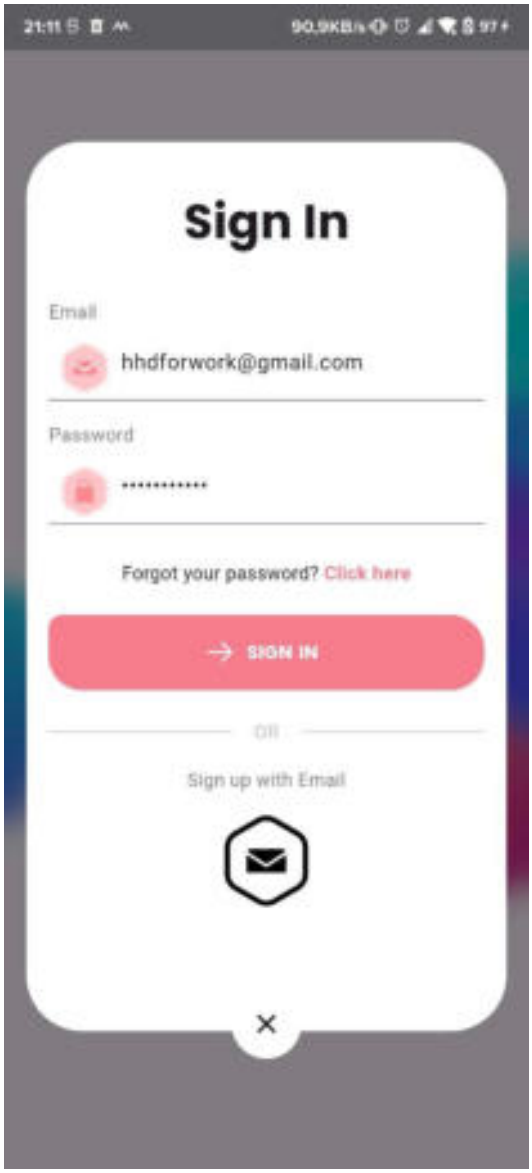
### *4.1.1. Màn hình truy cập của hệ thống*




*Hình 4.1: Màn hình splash truy cập hệ thống*

- Tiêu đề nổi bật: “Find your dream job”.
- Đoạn giới thiệu ngắn: Hướng dẫn người dùng vuốt để khám phá công việc và công ty.
- Nút Start Now giúp người dùng bắt đầu hành trình tìm việc.

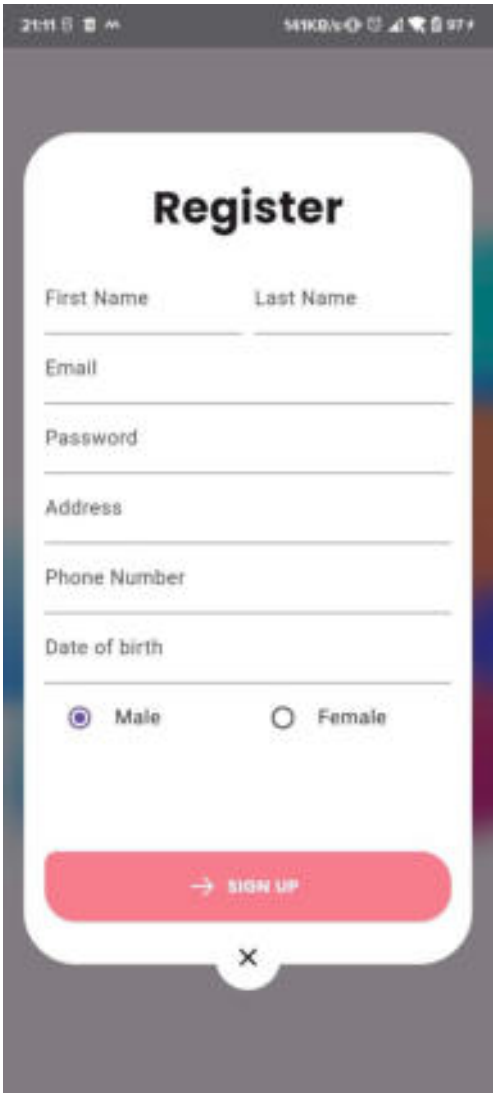
### 4.1.2. Màn hình đăng nhập tài khoản




*Hình 4.2: Màn Sign in đăng nhập tài khoản*

- Trường Email: Nhập địa chỉ email của người dùng.
- Trường Mật khẩu: Nhập mật khẩu tương ứng.
- Liên kết "Forgot your password?": Cho phép người dùng khôi phục mật khẩu nếu quên.
- Nút "SIGN IN": Đăng nhập vào hệ thống.
- Tuỳ chọn "Sign up with Email": Dành cho người dùng chưa có tài khoản.

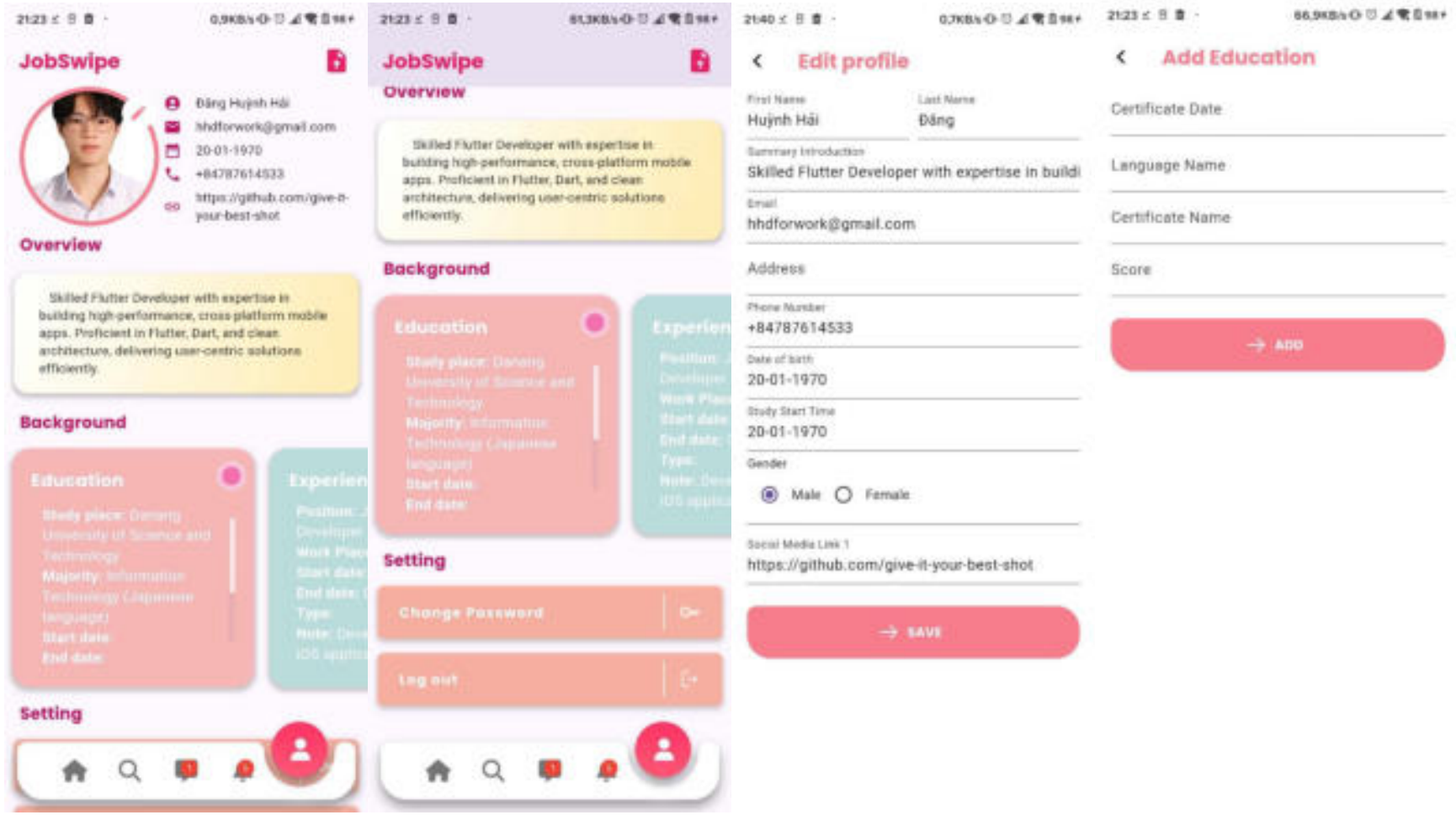
### 4.1.3. Màn hình đăng ký tài khoản

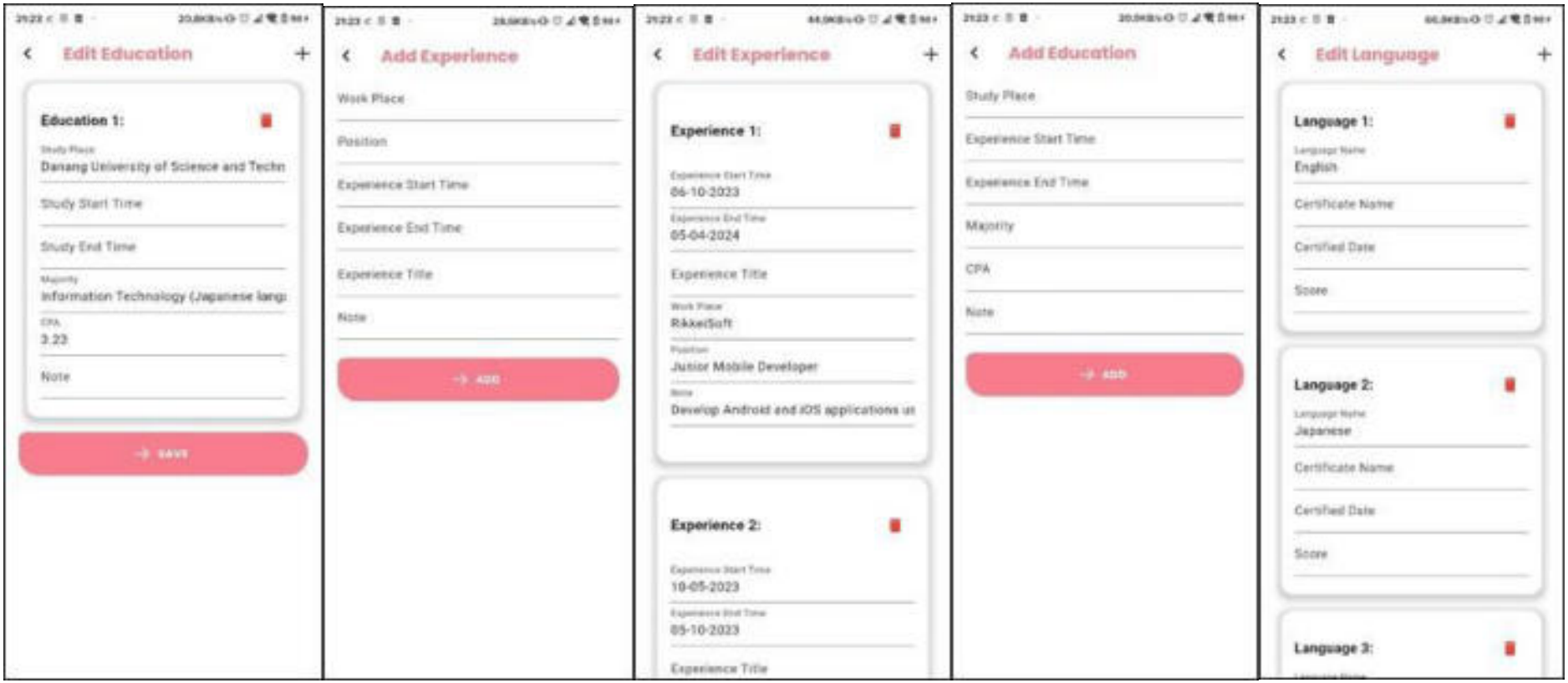



*Hình 4.3: Màn Resigter đăng ký tài khoản*

- Màn hình "Register" (Đăng ký)
  - + Thông tin cá nhân:
  - + First Name, Last Name
  - + Email, Password
  - + Address, Phone Number
  - + Date of birth
  - + Giới tính (Male / Female)
- Nút "SIGN UP": Hoàn tất đăng ký tài khoản.

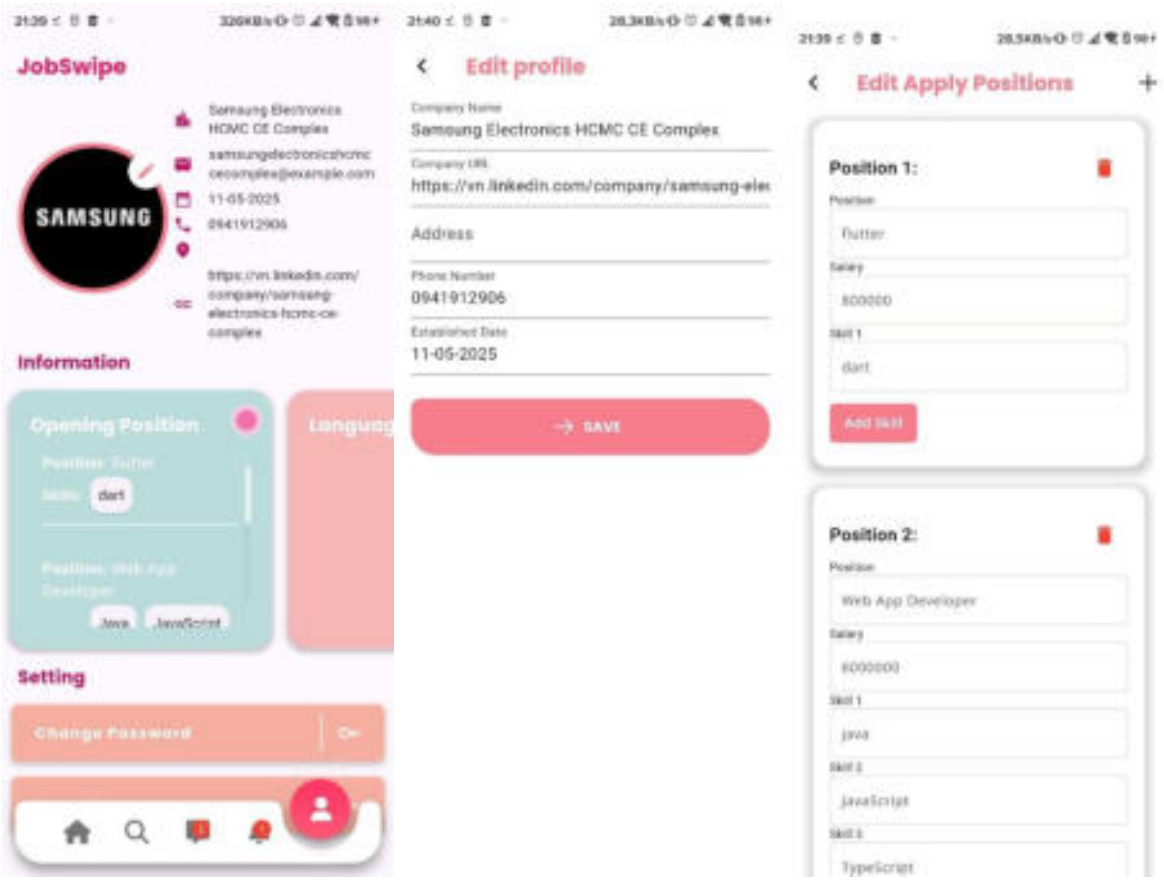
### 4.1.4. Màn hình quản lý thông tin






*Hình 4.4: Màn hình quản lý thông tin users*

- Hiển thị thông tin công ty: ảnh đại diện, tên công ty, email, ngày sinh, số điện thoại, giới tính, liên kết mạng xã hội.
- Phần giới thiệu năng lực (“Summary Introduction”) được làm nổi bật.
- Các mục như Education và Experience được phân chia rõ ràng theo dạng thẻ màu. Các tùy chọn Change Password và Log out, màn hình thể hiện chức năng xem/thêm/xóa/sửa các thông tin về học vấn, kinh nghiệm, ngôn ngữ, giải thưởng,…




*Hình 4.5: Màn hình quản lý thông tin công ty*

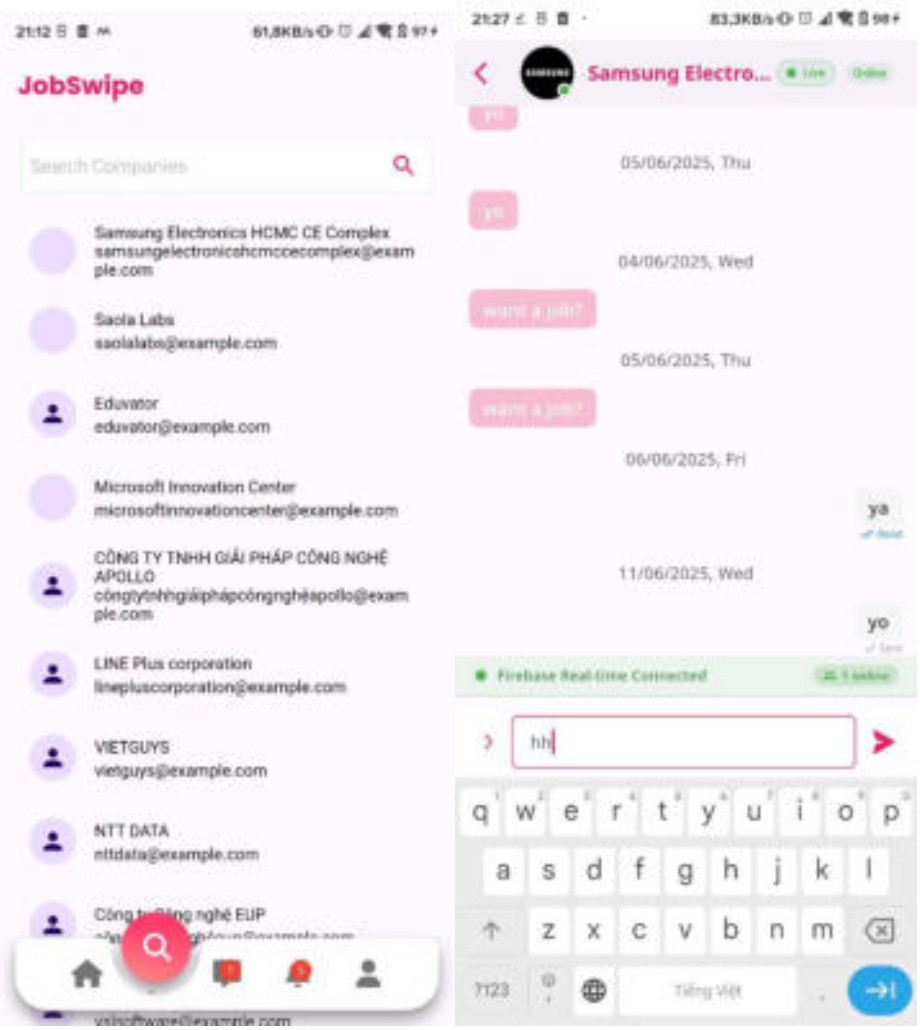
- Hiển thị thông tin công ty: ảnh đại diện, tên công ty, email, ngày thành lập, số điện thoại, liên kết mạng xã hội.
- Các mục như công việc đang tuyển và ngôn ngữ công ty đang sử dụng được phân chia rõ ràng theo dạng thẻ màu. Các tùy chọn Change Password và Log out, màn hình thể hiện chức năng xem/thêm/xóa/sửa các thông tin về công việc đang tuyển, thông tin công ty, ngôn ngữ,…

### 4.1.5. Màn hình thông báo


*Hình 4.6: Màn hình thông báo*

- Danh sách thông báo:
  - + Các trạng thái matching với ứng viên/công việc
  - + Trạng thái: Đã matching, yêu cầu mới, hoặc từ chối.
  - + Thời gian: 6 giờ, 7 ngày, 12 ngày trước.
  - + Nút “Request matching” (Yêu cầu matching): Tạo yêu cầu mới.
  - + Nút “Reject” (Từ chối): Từ chối yêu cầu.

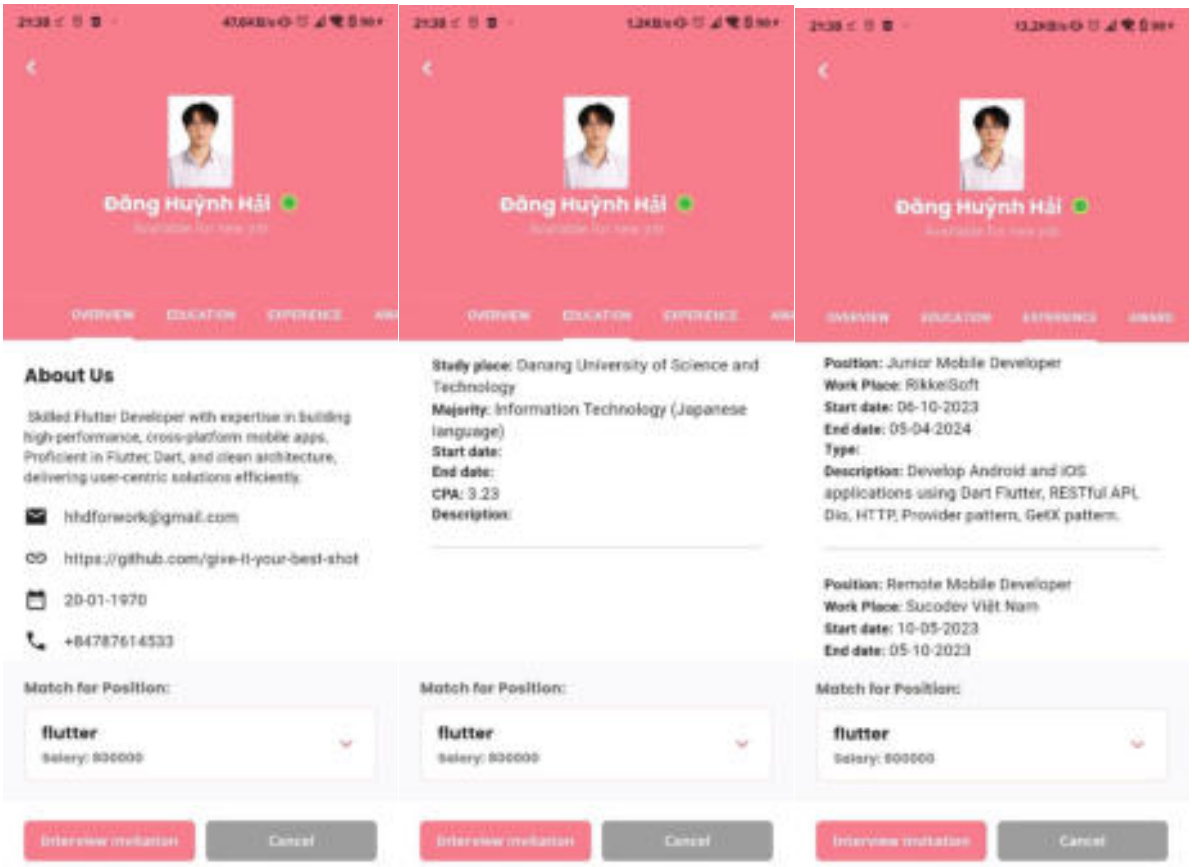
### 4.1.6. Màn hình nhắn tin




*Hình 4.7: Màn hình danh sách nhắn tin và nhắn tin riêng*

- Màn hình Danh sách nhắn tin
  - Hiển thị danh sách người dùng có thể nhắn tin, kèm theo ảnh đại diện, tên và email.
  - Có thanh tìm kiếm để lọc theo tên công ty, tên công việc / tên ứng viên.
- Màn hình Nhắn tin
  - Giao diện trò chuyện 1-1 với người dùng khác.
  - Hiển thị trạng thái "Online" và lịch sử hội thoại theo thời gian.
  - Tin nhắn của người dùng và đối phương được phân biệt trái/phải.
  - Có ô nhập tin nhắn, biểu tượng gửi ảnh và tệp đính kèm.

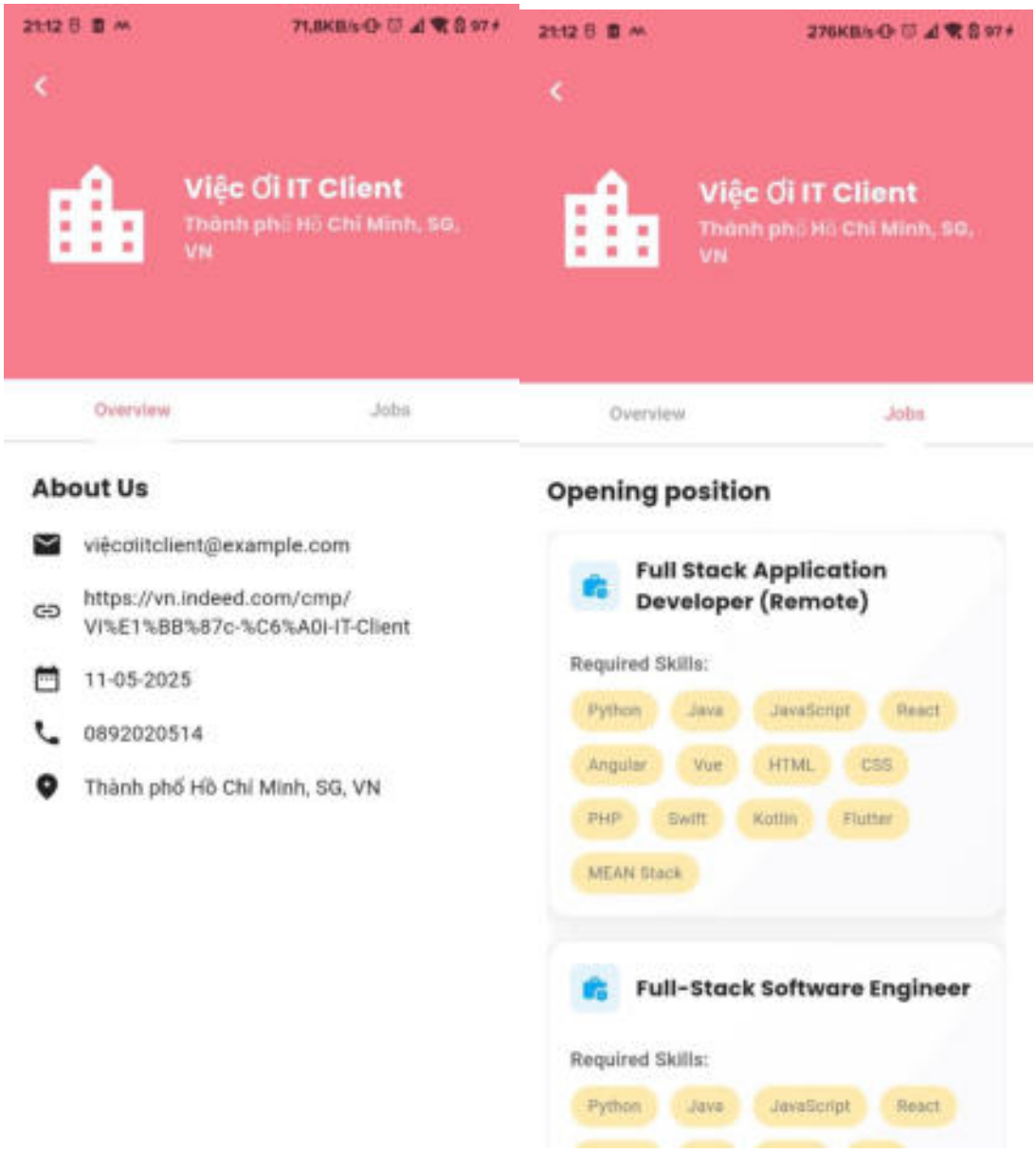
### 4.1.7. Màn hình xem thông tin chi tiết người dùng




*Hình 4.8: Màn hình xem thông tin chi tiết người dung*

- Người dùng (nhà tuyển dụng, người dùng tương tự) sẽ xem được thông tin công khai của người dùng khác gồm tổng quan, học vấn, kinh nghiệm, giải thưởng,…
- Ngoài ra, nhà tuyển dụng có thể mời tham gia phỏng vấn ở các vị trí công việc phù hợp với kinh nghiệm của người dùng.

### 4.1.8. Màn hình xem thông tin chi tiết công ty




*Hình 4.9: Màn hình xem thông tin công ty*

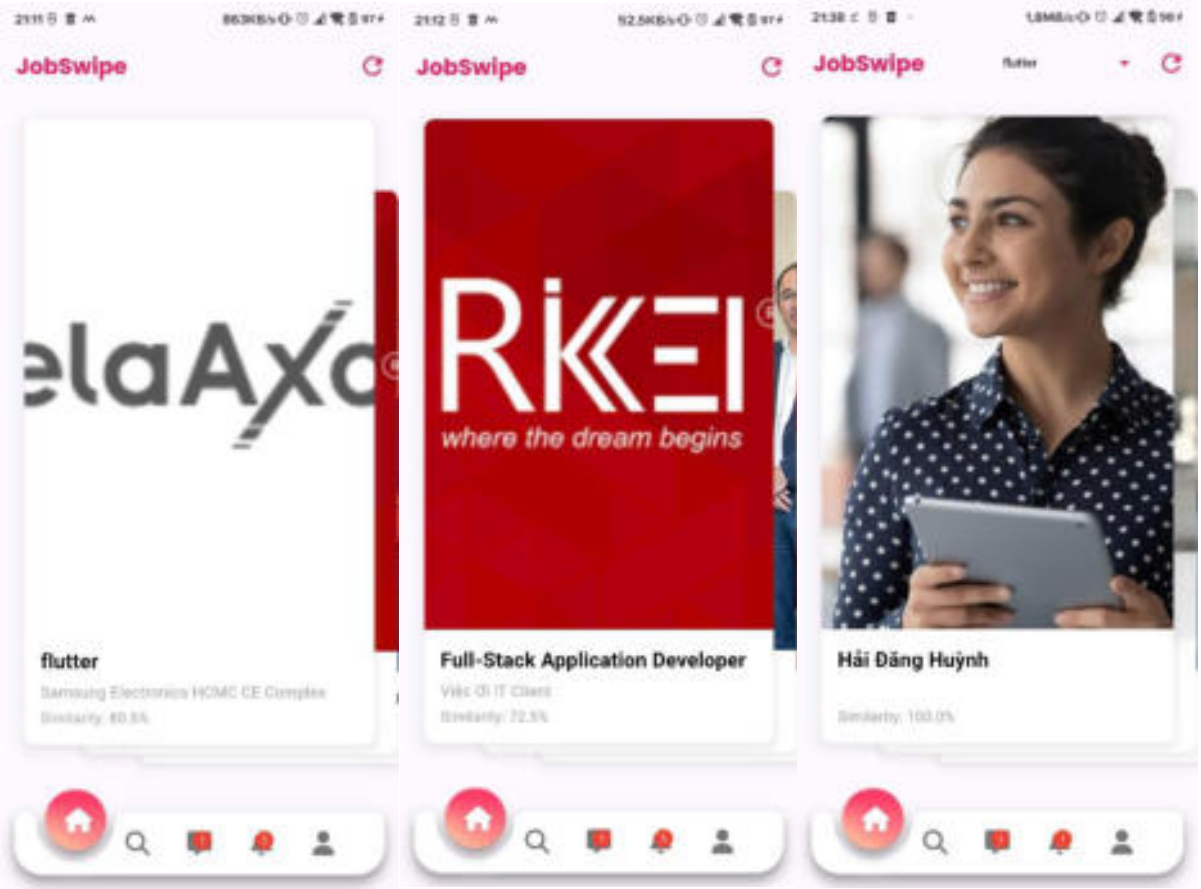
- Người dùng có thể xem thông tin giới thiệu của công ty gồm địa chỉ email, website, ngày thành lập, số điện thoại, địa chỉ,…
- Ngoài ra, người dùng có thể xem danh sách các công việc hiện tại đang có nhu cầu tuyển dụng ở công ty.

### 4.1.9. Màn hình chi tiết việc làm


*Hình 4.10: Màn hình ứng viên xem chi tiết thông tin tuyển dụng*

- Từ danh sách hiển thị các công việc, người dùng có thể chọn một công việc và xem thông tin chi tiết của nó.
- Thông tin chi tiết bao gồm thông tin công ty, mô tả, mức lương,…
- Ngoài ra, còn có nút "Apply this job" để ứng tuyển nhanh vị trí này.

### 4.1.10. Màn hình lướt và "quẹt" để matching




*Hình 4.11: Màn hình danh sách công việc / ứng viên phù hợp*

- Giao diện này sẽ tự động hiển thị danh sách các công việc cho người dùng ứng viên xem, ngoài thông tin về tên công việc, tên đơn vị tuyển dụng thì nó còn thể hiện mức độ tương thích của ứng viên và công việc.
- Đối với nhà tuyển dụng, họ cũng có thể xem được các ứng viên và mức độ tương thích của họ với từng công việc đăng có nhu cầu tuyển dụng.

## 4.2. Kết luận chương 4

Trong chương này, em đã trình bày quá trình triển khai thành công hệ thống JobSwipe – một nền tảng hỗ trợ tìm kiếm việc làm hiệu quả, được xây dựng bằng Flutter cho giao diện người dùng, Spring Boot (Java) cho phần backend và PostgreSQL làm hệ quản trị cơ sở dữ liệu.

Giao diện người dùng của ứng dụng được thiết kế hiện đại, trực quan, hướng đến trải nghiệm mượt mà trên thiết bị di động. Các tính năng chính như: đăng ký, đăng nhập, quản lý hồ sơ cá nhân, tìm kiếm công việc, nhắn tin, kết nối nhà tuyển dụng – đều được tích hợp và vận hành hiệu quả.

Việc kết hợp Flutter và Spring Boot giúp tối ưu hiệu suất giữa client và server, trong khi PostgreSQL hỗ trợ lưu trữ dữ liệu ổn định và linh hoạt. Hệ thống đã chứng minh được khả năng hoạt động ổn định, dễ sử dụng, đồng thời tạo ra trải nghiệm cá nhân hóa và chuyên nghiệp cho người tìm việc cũng như nhà tuyển dụng.

# KẾT LUẬN

## 1. Kết quả đạt được

Trong quá trình tìm hiểu, nghiên cứu cơ sở lý thuyết, phát triển và triển khai hệ thống, đồ án đã đạt được những kết quả sau:

**Về mặt lý thuyết:**

- Nắm được quy trình phát triển phần mềm hoàn chỉnh, từ lên kế hoạch đến triển khai và hoàn thiện.
- Áp dụng kiến thức để phân tích và thiết kế hệ thống, bao gồm việc sử dụng Dart và Flutter cho lập trình frontend, Java và Spring Boot cho backend, và Python với Flask cho phát triển AI.
- Đặc biệt, bạn đã làm quen với các kỹ thuật machine learning như vector embedding, tính toán độ tương đồng (cosine similarity, dot product,…), và ma trận cá nhân hóa cho hệ thống gợi ý.
- Có kinh nghiệm thiết kế cơ sở dữ liệu với PostgreSQL và thiết kế giao diện di động với Flutter.
- Có kinh nghiệm triển khai hệ thống trên AWS, sử dụng các dịch vụ như EC2 và S3, đảm bảo tính ổn định và khả năng mở rộng.

**Về mặt ứng dụng:**

- Giao diện swipe kiểu Tinder, cho phép người dùng nhanh chóng thể hiện sự quan tâm đến công việc hoặc ứng viên.
- Tính năng trích xuất thông tin CV tự động từ file PDF, giúp chuẩn hóa dữ liệu và giảm công sức nhập liệu.
- Hệ thống gợi ý thông minh dựa trên vector embedding và ma trận cá nhân hóa, tối ưu hóa việc đề xuất công việc và ứng viên.
- Tính năng chat trực tiếp, hỗ trợ giao tiếp tức thời giữa hai bên, cùng với quản lý hồ sơ, công việc, và lịch phỏng vấn hiệu quả.
- Ứng dụng AI tiên tiến giúp cải thiện trải nghiệm người dùng, tiết kiệm thời gian và chi phí cho cả người tìm việc và nhà tuyển dụng.

## 2. Những vấn đề chưa giải quyết

Mặc dù đã đạt được những kết quả nhất định của một hệ thống tìm kiếm việc làm, hệ thống vẫn còn tồn tại một số hạn chế như sau:

- Chưa hỗ trợ phiên bản web, khiến người dùng chỉ có thể truy cập qua ứng dụng di động, hạn chế khả năng sử dụng trên máy tính.
- Chưa cho phép upload video, chẳng hạn như video giới thiệu bản thân của ứng viên hoặc video giới thiệu công ty của nhà tuyển dụng, có thể làm giảm tính hấp dẫn và cá nhân hóa.
- Chưa tích hợp với các nền tảng việc làm khác, dẫn đến việc đồng bộ dữ liệu và mở rộng nguồn ứng viên/công việc bị hạn chế.

## 3. Hướng phát triển

Bên cạnh những kết quả đạt được đã nêu trên, đồ án này còn có thể phát triển và hoàn thiện thêm nhiều tính năng:

- Phát triển phiên bản web để mở rộng khả năng truy cập, giúp người dùng có thể sử dụng hệ thống trên máy tính, tăng tính tiện lợi.
- Thêm chức năng upload video, cho phép ứng viên thêm video giới thiệu bản thân và nhà tuyển dụng thêm video giới thiệu công ty, tăng tính cá nhân hóa và hấp dẫn.
- Tích hợp với các nền tảng việc làm khác, chẳng hạn như LinkedIn hoặc các trang web tuyển dụng phổ biến, để đồng bộ dữ liệu, mở rộng nguồn ứng viên và công việc, và cải thiện hiệu quả kết nối.
- Tiến hành khảo sát mức độ quan tâm và chuẩn hóa các hành vi tương tác của người dùng thành thang điểm từ -1 đến 1, nhằm phục vụ cho việc tinh chỉnh và nâng cao độ chính xác của hệ thống gợi ý.

# DANH MỤC TÀI LIỆU THAM KHẢO

[1] Flutter, “Official Documentation.” [Online]. Available: https://docs.flutter.dev/. [Accessed: Jun. 15, 2025].

[2] Dart, “Official Documentation.” [Online]. Available: https://dart.dev/docs. [Accessed: Jun. 15, 2025].

[3] Spring Boot, “Official Documentation.” [Online]. Available: https://docs.spring.io/spring-boot/. [Accessed: Jun. 15, 2025].

[4] Flask, “Official Documentation.” [Online]. Available: https://flask.palletsprojects.com/. [Accessed: Jun. 15, 2025].

[5] PostgreSQL, “Official Documentation.” [Online]. Available: https://www.postgresql.org/docs/. [Accessed: Jun. 15, 2025].

[6] Redis, “Official Documentation.” [Online]. Available: https://redis.io/docs/. [Accessed: Jun. 15, 2025].

[7] ChromaDB, “Official Documentation.” [Online]. Available: https://docs.trychroma.com/. [Accessed: Jun. 15, 2025].

[8] Amazon Web Services, “AWS Documentation.” [Online]. Available: https://docs.aws.amazon.com/. [Accessed: Jun. 15, 2025].

[9] Git, “Official Documentation.” [Online]. Available: https://git-scm.com/doc. [Accessed: Jun. 15, 2025].

[10] GitHub, “Official Documentation.” [Online]. Available: https://docs.github.com/. [Accessed: Jun. 15, 2025].

[11] JWT.io, “JSON Web Tokens (JWT) – Introduction.” [Online]. Available: https://jwt.io/. [Accessed: Jun. 15, 2025].

[12] OpenAI, "OpenAI Embeddings Guide." [Online]. Available: https://platform.openai.com/docs/guides/embeddings. [Accessed: Jun. 15, 2025].

[13] scikit-learn, "sklearn.metrics.pairwise.cosine_similarity." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html. [Accessed: Jun. 15, 2025].

[14] X. Zhao et al., "Embedding in Recommender Systems: A Survey," *arXiv preprint arXiv:2310.18608*, 2023. [Online]. Available: https://arxiv.org/abs/2310.18608. [Accessed: Jun. 15, 2025].

[15] DataCamp, "Gradient Descent in Machine Learning: A Deep Dive." [Online]. Available: https://www.datacamp.com/tutorial/tutorial-gradient-descent. [Accessed: Jun. 15, 2025].