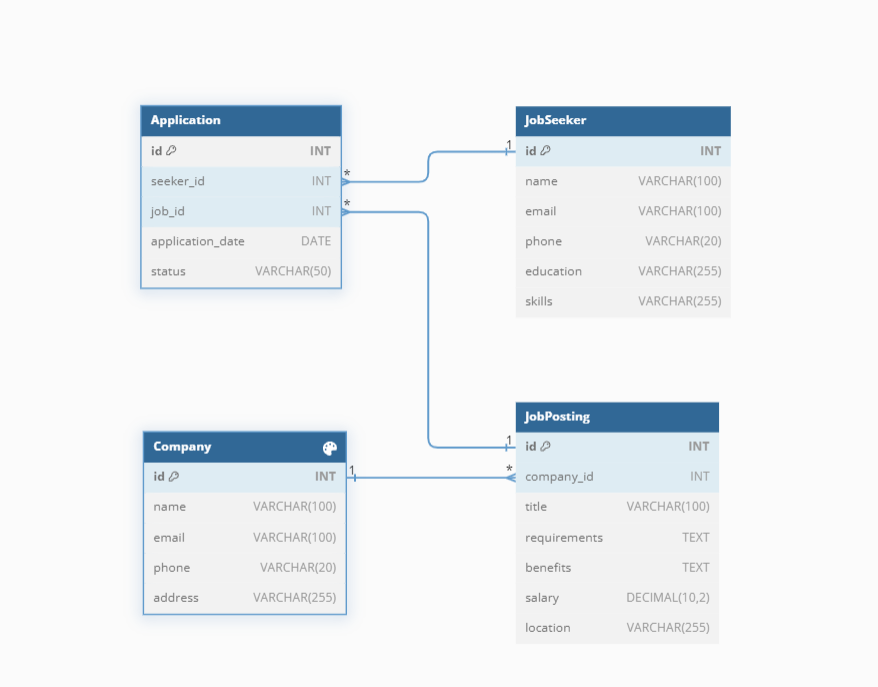
**BÀI KIỂM TRA**

1. **Thiết kết**
2. Dưới dây là sơ đồ mối quan hệ thực thể



1. Mô tả mối quan hệ :
2. Company (Công ty) có một hoặc nhiều Job Posting (Công việc).

Quan hệ: Một-nhiều (One-to-Many)

Khóa ngoại: company\_id trong bảng JobPosting tham chiếu đến company\_id trong bảng Company.

1. Job Seeker (Người tìm việc) có thể ứng tuyển vào nhiều Job Posting (Công việc).

Quan hệ: Nhiều-nhiều (Many-to-Many)

Bảng liên kết: Application với các khóa ngoại seeker\_id và job\_id tham chiếu đến JobSeeker và JobPosting.

1. Job Posting (Công việc) có nhiều Application (Đơn ứng tuyển).

Quan hệ: Một-nhiều (One-to-Many)

Khóa ngoại: job\_id trong bảng Application tham chiếu đến job\_id trong bảng JobPosting.

1. Job Seeker (Người tìm việc) có nhiều Application (Đơn ứng tuyển).

Quan hệ: Một-nhiều (One-to-Many)

Khóa ngoại: seeker\_id trong bảng Application tham chiếu đến seeker\_id trong bảng JobSeeker.

1. **Quy trình dự án**

1. Số nhánh trong GIT và mục đích của từng nhánh

Trong thế giới phát triển phần mềm, Git đóng vai trò quan trọng như công cụ quản lý phiên bản, giúp theo dõi thay đổi code và cộng tác hiệu quả. Hiểu rõ các nhánh Git và mục đích sử dụng sẽ giúp bạn tối ưu hóa quy trình làm việc, nâng cao năng suất và chất lượng dự án. Một số nhánh cơ bản khi làm việc với GIT:

Nhánh chính (Main/Master): Đây là nhánh chính của dự án, chứa mã nguồn ở trạng thái ổn định, đã được kiểm tra và sẵn sàng để phát hành.

* Nhánh Develop: Nhánh phát triển, nơi mà tất cả các tính năng mới được tích hợp trước khi được đưa vào nhánh chính.
* Nhánh tính năng (Feature Branch): Tách ra từ nhánh Develop để triển khai từng tính năng riêng biệt. Mỗi nhánh tính năng chứa các thay đổi liên quan đến tính năng đó và sau khi hoàn thành sẽ được hợp nhất vào nhánh Develop.
* Nhánh Release: Được tạo từ nhánh Develop khi chuẩn bị cho bản phát hành mới, cho phép thực hiện các bước hoàn thiện cuối cùng trước khi ra mắt.
* Nhánh Hotfix: "Cứu hỏa" từ nhánh chính để sửa lỗi khẩn cấp. Sau khi sửa lỗi, nhánh này sẽ được hợp nhất lại vào cả nhánh chính và nhánh Develop.

Trong một dự án phát triển phần mềm, Giám đốc kỹ thuật thường là người to nhất, quản lí các nhân viên , nên tôi nghĩ giám giốc chỉ quan tâm đến hai nhánh chính là nhánh Main (hoặc Master),  Develop, **Hotfix Branches** và **Release Branches**:

1. Quy trình phát triển từ đầu đến phát hành mô-đun

1. Khởi tạo mô-đun

Tạo một feature branch từ develop branch: Bắt đầu bằng cách tạo một nhánh tính năng (feature branch) từ nhánh phát triển (develop branch). Điều này giúp tách biệt các thay đổi mới khỏi mã nguồn chính.

Lên kế hoạch và xác định yêu cầu cho mô-đun mới: Xác định các yêu cầu chi tiết cho mô-đun mới, bao gồm các tính năng cần có và các yêu cầu kỹ thuật. Lên kế hoạch chi tiết bao gồm phân công nhiệm vụ và thời gian hoàn thành.

2. Phát triển mô-đun

Code và thực hiện các thay đổi cần thiết trong feature branch: Tiến hành viết mã cho các tính năng mới theo kế hoạch đã định và thực hiện các thay đổi cần thiết trong feature branch.

Thực hiện unit testing cho các phần mã đã viết: Sau khi hoàn thành việc viết mã, thực hiện kiểm thử đơn vị (unit testing) để đảm bảo rằng mỗi phần mã hoạt động đúng như mong đợi.

3. Kiểm tra và tích hợp

Gửi pull request (PR) từ feature branch sang develop branch: Gửi một yêu cầu kéo (pull request - PR) từ feature branch sang develop branch để các thành viên khác trong nhóm xem xét và thảo luận về các thay đổi.

Code review và thực hiện integration testing: Thực hiện code review để đảm bảo rằng mã viết ra đáp ứng các tiêu chuẩn về chất lượng. Tiếp theo, thực hiện kiểm thử tích hợp (integration testing) để đảm bảo rằng các phần mã hoạt động tốt khi kết hợp với nhau.

Hợp nhất feature branch vào develop branch sau khi các bài kiểm tra đều thành công: Sau khi hoàn thành code review và kiểm thử tích hợp, hợp nhất feature branch vào develop branch.

4. Chuẩn bị phát hành

Tạo release branch từ develop branch: Khi chuẩn bị phát hành một phiên bản mới, tạo một nhánh phát hành (release branch) từ nhánh phát triển.

Thực hiện các bước kiểm tra cuối cùng và kiểm tra chất lượng: Thực hiện các kiểm thử cuối cùng để đảm bảo rằng phiên bản chuẩn bị phát hành không có lỗi và đáp ứng tất cả các yêu cầu.

Cập nhật tài liệu và ghi chú phát hành: Cập nhật tài liệu liên quan đến phiên bản mới, bao gồm hướng dẫn sử dụng, tài liệu kỹ thuật và ghi chú phát hành (release notes).

5. Phát hành mô-đun

Hợp nhất release branch vào main branch: Sau khi hoàn tất các kiểm thử cuối cùng và cập nhật tài liệu, hợp nhất release branch vào main branch. Nhánh chính (main branch) sẽ chứa phiên bản ổn định và đã sẵn sàng để phát hành của phần mềm.

Tạo tag cho phiên bản mới trên main branch: Tạo một thẻ (tag) trên nhánh chính để đánh dấu phiên bản mới.

Deploy mô-đun mới vào môi trường sản xuất: Triển khai phiên bản mới vào môi trường sản xuất. Đảm bảo rằng quá trình triển khai diễn ra suôn sẻ và hệ thống hoạt động ổn định sau khi triển khai.

Hợp nhất release branch vào develop branch để đồng bộ hóa các thay đổi: Cuối cùng, hợp nhất release branch vào develop branch để đảm bảo rằng tất cả các thay đổi và sửa lỗi đã được đồng bộ hóa giữa các nhánh.

1. **Kiểm tra**

1. Các loại chiến lược thử nghiệm

Unit Testing: Kiểm tra từng đơn vị mã nguồn để đảm bảo rằng mỗi phần riêng lẻ hoạt động chính xác.

Integration Testing: Kiểm tra sự kết hợp của các đơn vị mã nguồn để đảm bảo chúng hoạt động tốt với nhau.

System Testing: Kiểm tra hệ thống tổng thể để đảm bảo rằng tất cả các thành phần hoạt động tốt trong môi trường thực tế.

Acceptance Testing: Kiểm tra hệ thống để đảm bảo rằng nó đáp ứng các yêu cầu và mong đợi của người dùng cuối.

Regression Testing: Kiểm tra lại hệ thống sau khi có các thay đổi để đảm bảo rằng các tính năng cũ vẫn hoạt động tốt.

1. Mục đích của kiểm tra đơn vị (Unit Testing)

Unit Testing được sử dụng để kiểm tra từng đơn vị hoặc thành phần của mã nguồn, như các hàm hoặc phương thức, đảm bảo rằng chúng hoạt động đúng như mong đợi. Điều này giúp phát hiện lỗi sớm trong quá trình phát triển và giảm thiểu sự cố trong giai đoạn sau.

Mục đích của kiểm thử tích hợp (Integration Testing)

Integration Testing được sử dụng để kiểm tra sự kết hợp của các đơn vị mã nguồn và đảm bảo rằng chúng hoạt động tốt khi tích hợp với nhau. Điều này giúp phát hiện các vấn đề liên quan đến tương tác giữa các thành phần khác nhau trong hệ thống.

1. Quy trình kiểm thử mô-đun từ đầu đến phát hành

Chuẩn bị kiểm thử:

Đọc và hiểu yêu cầu của mô-đun.

Chuẩn bị môi trường kiểm thử.

Viết và xem xét kế hoạch kiểm thử.

Kiểm thử đơn vị (Unit Testing):

Viết và chạy các unit test cho từng thành phần của mô-đun.

Đảm bảo rằng từng đơn vị hoạt động đúng như mong đợi.

Kiểm thử tích hợp (Integration Testing):

Viết và chạy các integration test để kiểm tra sự kết hợp của các thành phần.

Đảm bảo rằng các thành phần hoạt động tốt khi tích hợp với nhau.

Kiểm thử hệ thống (System Testing):

Thực hiện kiểm thử toàn bộ hệ thống để đảm bảo rằng mô-đun hoạt động tốt trong môi trường thực tế.

Kiểm tra các kịch bản sử dụng và luồng công việc chính.

Kiểm thử chấp nhận (Acceptance Testing):

Thực hiện kiểm thử chấp nhận để đảm bảo rằng mô-đun đáp ứng các yêu cầu và mong đợi của người dùng cuối.

Thu thập phản hồi từ người dùng và thực hiện các điều chỉnh nếu cần.

Kiểm thử hồi quy (Regression Testing):

Thực hiện kiểm thử hồi quy để đảm bảo rằng các tính năng cũ vẫn hoạt động tốt sau khi tích hợp mô-đun mới.

Hoàn thiện và phát hành:

Chuẩn bị tài liệu và ghi chú phát hành.

Thực hiện kiểm tra cuối cùng.

Phát hành mô-đun mới vào môi trường sản xuất.

Theo dõi và hỗ trợ sau phát hành để đảm bảo mọi vấn đề được giải quyết kịp thời.