Form Employee

* Trang Information:
  + Hiển thị thông tin cá nhân, Department Name, Position và Email của Employee đó
  + Cho phép người dùng Update: Name, Phone, Gender, HomeTown và Email
* Trang Manager:
  + Nếu Department của nhân viên không có Manager thì sẽ hiện “ This department does not have manager ” và các ô thuộc Manager sẽ để trống
  + Hiển thị thông tin cá nhân, email, Department Name, Department Phone, Department Address
  + Nếu Manager không có account thì Email sẽ hiện “ Does not have an email ”
* Trang Project:
  + Hiển thị Employee\_id, Project\_id, Project\_name, Description, Start Date, End Date và Active của Project
  + Xử lý tìm kiếm trên tất cả các trường dữ liệu đã hiển thị ngoại trừ Employee\_id

Create Account

* Nhập vào: User\_name, Password, Full\_name, Email, Type
* Xử lý kiểm tra không cho trùng Email và User\_name
* Xử lý Hash password
  + Tạo một salt ngẫu nhiên có kích thước 32 byte sử dụng lớp ‘**RNGCryptoServiceProvider**’. Salt được sử dụng để tạo ra một mã hóa khác nhau cho mật khẩu ngay cả khi cùng một mật khẩu được sử dụng bởi nhiều người dùng.
  + Chuyển đổi mật khẩu và salt thành một mảng byte. Để làm điều này, phương thức sử dụng lớp ‘**Encoding.UTF8’** để chuyển đổi mật khẩu thành mảng byte và sau đó sử dụng ‘**Buffer.BlockCopy’**để kết hợp mảng mật khẩu và salt vào một mảng mới.
  + Khởi tạo đối tượng ‘**SHA256’** để thực hiện thuật toán băm trên mật khẩu và salt.
  + Thực hiện thuật toán băm trên mảng byte chứa mật khẩu và salt sử dụng ‘**SHA256.ComputeHash’**. Kết quả là một mảng byte.
  + Chuyển đổi mảng byte thu được từ bước 4 thành chuỗi hexa để lưu trữ. Để làm điều này, phương thức sử dụng lớp ‘**StringBuilder’** và một vòng lặp để chuyển đổi mỗi byte thành hai ký tự hexa.
  + Thêm salt vào đầu chuỗi hexa thu được từ bước 5 để có thể phục hồi salt và kiểm tra mật khẩu sau này.
  + Trả về chuỗi hexa cuối cùng chứa cả salt và mã hóa mật khẩu.

Login

* Xử lý so sánh mật khẩu với mật khẩu đã Hash
  + Lấy salt từ chuỗi hexa đầu tiên trong mật khẩu lưu trữ. Salt là một chuỗi ngẫu nhiên được sử dụng để tăng tính bảo mật của mật khẩu băm, bằng cách trộn thêm một chuỗi ngẫu nhiên vào trước khi thực hiện thuật toán băm.
  + Tạo một mảng byte để chứa salt và mật khẩu đầu vào.
  + Thực hiện thuật toán băm trên mảng byte chứa salt và mật khẩu đầu vào, sử dụng đối tượng **SHA256**.
  + Chuyển đổi kết quả thuật toán băm từ mảng byte thành chuỗi hexa để so sánh với mật khẩu lưu trữ.
  + Thêm salt vào đầu chuỗi hexa kết quả thuật toán băm.
  + So sánh chuỗi hexa kết quả thuật toán băm với chuỗi hexa của mật khẩu lưu trữ. Nếu hai chuỗi giống nhau, mật khẩu đầu vào được xác thực đúng và phương thức trả về giá trị true, ngược lại phương thức trả về giá trị false.
  + Lưu ý rằng trong quá trình so sánh chuỗi, phương thức sử dụng ‘**StringComparison.OrdinalIgnoreCase’** để bỏ qua sự khác biệt về chữ hoa, chữ thường giữa hai chuỗi.

Admin

* Xuất tất cả thông tin trong database lên table ( Project, Department, Salary )
* Chức năng thêm, sửa, xóa, tìm kiếm các trường hiển thị trên bảng
* **Chưa xử lý ràng buộc**