**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**======\*\*\*======**

****

**BÁO CÁO BTL THUỘC HỌC PHẦN:**

**CÔNG CỤ PHÁT TRIÊN PHẦN MỀM**

**Nghiên cứu xây dựng hệ thống chấm công tự động**

**bằng công cụ Emgu, OpenCV trên Visual Studio**

|  |  |
| --- | --- |
| GVHD: | Ths Nguyễn Thái Cường |
| Nhóm - Lớp: | 3 – 20222IT6003004 |
| Thành viên: | Nguyễn Phúc Hưng |
|  | Nguyễn Khánh Thọ  Nguyễn Quang Hưng  Đặng Việt Anh Đinh Anh Quân |
|  | *…* |
|  |  |
|  |  |

Hà Nội, Năm 2023

# **MỤC LỤC**

Contents

[**MỤC LỤC** 2](#_Toc136623286)

[**LỜI NÓI ĐẦU** 3](#_Toc136623287)

[**CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI** 4](#_Toc136623288)

[**1.1** **Đặt vấn đề** 4](#_Toc136623289)

[**1.2** **Một số nghiên cứu trước đây** 4](#_Toc136623290)

[**1.3 Phạm vi đề tài.** 6](#_Toc136623291)

[**1.4 Phương pháp nghiên cứu** 7](#_Toc136623292)

[**1.5 Bố Cục và nội dung** 7](#_Toc136623293)

[**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT** 8](#_Toc136623294)

[**2.1 Giới thiệu về bài toán dò tìm khuôn mặt** 8](#_Toc136623295)

[**2.2 Đặc trưng Haar-Like Features** 8](#_Toc136623296)

[**2.3** **Trích đặc trưng Haar-Like từ ảnh** 9](#_Toc136623297)

[**2.4 Thuật toán AdaBoost** 11](#_Toc136623298)

[**2.5 Bộ phân tầng Cascade of Boosted Classifier** 12](#_Toc136623299)

[**2.6 Định vị đối tượng** 14](#_Toc136623300)

[**2.7 Tổng quan về thư viện OpenCV để nghiên cứu về nhận dạng mặt người** 15](#_Toc136623301)

[**2.8 Thư viện EmguCV** 16](#_Toc136623302)

[**CHƯƠNG 3: YÊU CẦU CHỨC NĂNG VÀ MÔ HÌNH HÓA USECASE** 17](#_Toc136623303)

[**3.1 Yêu cầu chức năng** 17](#_Toc136623304)

[**3.2 Mô hình hóa CSDL** 17](#_Toc136623305)

[3.2.1 Các yêu cầu về dữ liệu 17](#_Toc136623306)

[3.2.2 Biểu đồ thực thể liên kết 18](#_Toc136623307)

[3.2.3 Mô tả chi tiết các bảng CSDL 19](#_Toc136623308)

[**CHƯƠNG 4: MÔ TẢ USECASE** 21](#_Toc136623309)

[**4.1 Danh sách Usecase** 21](#_Toc136623310)

[**4.2 Sơ đồ UC** 23](#_Toc136623311)

[**4.3 Mô tả chi tiết UC** 24](#_Toc136623312)

[UC01: Đăng nhập 24](#_Toc136623313)

[UC02: Thêm tài khoản 25](#_Toc136623314)

[UC03: Cập nhật tài khoản 26](#_Toc136623315)

[UC04: Xóa tài khoản 27](#_Toc136623316)

[UC05: Tìm kiếm tài khoản 29](#_Toc136623317)

[UC06: Thêm nhân viên 30](#_Toc136623318)

[UC07: Cập nhật thông tin nhân viên 31](#_Toc136623319)

[UC08: Xóa nhân viên 33](#_Toc136623320)

[UC09: Tìm kiếm nhân viên 34](#_Toc136623321)

[UC10: Chấm công nhân viên 35](#_Toc136623322)

[UC11: Xem báo cáo theo ngày 37](#_Toc136623323)

[UC12: Xuất báo cáo theo ngày 39](#_Toc136623324)

[UC13: Xem báo cáo theo tháng 40](#_Toc136623325)

[UC14: Xuất báo cáo theo tháng 42](#_Toc136623326)

[UC15: Lấy lại mật khẩu 43](#_Toc136623327)

[UC16: Xem bảng chấm công 45](#_Toc136623328)

[UC17: Xuất bảng chấm công 47](#_Toc136623329)

[UC18: : Gửi báo cáo theo ngày qua email 48](#_Toc136623330)

[UC19: Gửi báo cáo theo tháng qua email 50](#_Toc136623331)

[UC20: Gửi bảng chấm công qua Email 52](#_Toc136623332)

[UC21: Thêm nhân viên đi muộn 54](#_Toc136623333)

[UC22: Sửa thông tin nhân viên đi muộn 55](#_Toc136623334)

[UC23: Xóa thông tin nhân viên đi muộn 57](#_Toc136623335)

[UC24: Tìm thông tin nhân viên đi muộn 58](#_Toc136623336)

[UC25: Gửi yêu cầu 60](#_Toc136623337)

[UC26: Xử lý yêu cầu 62](#_Toc136623338)

[**CHƯƠNG 5: GIAO DIỆN NGƯỜI DÙNG** 64](#_Toc136623339)

[**5.1 Login** 64](#_Toc136623340)

[**5.2 Recovery** 64](#_Toc136623341)

[**5.3 Account manage** 65](#_Toc136623342)

[**5.4 Add account** 65](#_Toc136623343)

[**5.5 Edit account** 66](#_Toc136623344)

[**5.5 Request(Admin)** 66](#_Toc136623345)

[**5.6 Accountant** 67](#_Toc136623346)

[**5.7 Attendance** 67](#_Toc136623347)

[**5.8 Late manage** 68](#_Toc136623348)

[**5.9 Add late** 68](#_Toc136623349)

[**5.10 Edit late** 69](#_Toc136623350)

[**5.11 Report day** 69](#_Toc136623351)

[**5.12 Report month** 69](#_Toc136623352)

[**5.13 Request(Staff)** 70](#_Toc136623353)

[**5.14 Worker manage** 70](#_Toc136623354)

[**5.15 Add worker** 71](#_Toc136623355)

[**5.16 Edit worker** 71](#_Toc136623356)

[**CHƯƠNG 6: THỰC HIỆN BÀI TOÁN** 73](#_Toc136623357)

[**6.1 Đặng Việt Anh** 73](#_Toc136623358)

[**Màn hình Account manage:** 73](#_Toc136623359)

[**Màn hình Add account:** 74](#_Toc136623360)

[**Màn hình Edit account:** 75](#_Toc136623361)

[**Màn hình Request(Admin):** 76](#_Toc136623362)

[**6.2 Nguyễn Phúc Hưng** 77](#_Toc136623363)

[**Màn hình Login:** 78](#_Toc136623364)

[**Màn hình Recover:** 79](#_Toc136623365)

[**Màn hình Attendance:** 80](#_Toc136623366)

[**Màn hình Request(Staff):** 82](#_Toc136623367)

[**6.3 Nguyễn Quang Hưng** 83](#_Toc136623368)

[**Màn hình Worker manage:** 83](#_Toc136623369)

[**Màn hình Add worker:** 83](#_Toc136623370)

[**Màn hình Edit worker:** 85](#_Toc136623371)

[**6.2 Nguyễn Khánh Thọ** 87](#_Toc136623372)

[**Màn hình Report\_Day:** 87](#_Toc136623373)

[**Màn hình Report\_Month:** 94](#_Toc136623374)

[**Màn hình Send\_Email:** 96](#_Toc136623375)

[**Màn hình Late\_Manage:** 99](#_Toc136623376)

[**Màn hình AddLate:** 101](#_Toc136623377)

[**6.3 Đinh Anh Quân** 102](#_Toc136623378)

[**Màn hình Accountant:** 102](#_Toc136623379)

[**Màn hình Late\_Manage:** 106](#_Toc136623380)

[**Màn hình Edit\_Late:** 108](#_Toc136623381)

# **LỜI NÓI ĐẦU**

Là một sinh viên trường đại học Công nghiệp Hà Nội, báo cáo bài tập lớn là minh chứng cho những kiến thức đã có sau thời gian học tập. Trong quá trình hoàn thành bài tập lớn, ngoài sự cố gắng của nhóm thì nhóm cũng nhận được sự hướng dẫn tận tình của thầy **Ths**.Nguyễn Thái Cường. Qua đây, nhóm xin chân thành cảm ơn khoa Công nghệ thông tin, trường đại học Công nghiệp Hà Nội đã trang bị kiến thức cho em trong suốt quá trình học tập. Đặc biệt, nhóm cảm ơn tới thầy ThS. Nguyễn Thái Cường đã luôn giúp đỡ, hướng dẫn, chỉ bảo tận tình để giúp nhóm hoàn thành tốt bài tập lớn của mình. Nhóm đã cố gắng hoàn thành bài tập lớn nhưng vẫn rất mong nhận được sự đóng góp ý của thầy cô và các bạn để bài tập lớn cuả nhóm được hoàn thiện hơn.

# **CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI**

* 1. **Đặt vấn đề**

Trong thời kỳ phát triển mạnh mẽ của công nghệ thông tin, con người đang dần áp dụng các ứng dụng khoa học máy tính để phục vụ các công việc hằng ngày. Trong đó, những phần mềm bảo mật an ninh được thiết kế dựa trên hệ thống thị giác máy tính để thực hiện việc phát hiện, nhận dạng và sàng lọc xuất hiện ngày càng nhiều với những đối tượng ngày càng phong phú, đa dạng. Hiện nay, các ứng dụng nhận dạng khuôn mặt được xây dựng và phát triển trên rất nhiều thiết bị cũng như hệ thống nhằm giúp con người thuận tiện hơn trong việc quản lý, bảo mật, giám sát hoặc tìm kiếm. Các ứng dụng nhận diện khuôn mặt, nụ cười trong máy ảnh hoặc smartphone dần trở nên quen thuộc với mọi người.

Nhận thấy sự cần thiết trong việc nghiên cứu công nghệ nhận diện giúp con người có thể thuận tiện hơn trong nhiều công việc trong cuộc sống,kèm theo đó là tiền đề để nâng cao và phát triển mạnh về hệ thống thị giác máy tính, các ứng dụng về nhận dạng hình ảnh khuôn mặt nhằm mục đích tăng cáo tính bảo mật của hệ thống.

Đề tài với tên gọi “ nghiên cứu về thư viện OpenCV – nghiên cứu về hệ thống tự động chấm công bằng cách nhận dạng khuôn mặt” được thực hiện với mong muốn khai thác những khía cạnh về nhận dạng tự động và tự động vận hành nhằm giảm gánh nặng cho con người cũng như tiết kiệm thời gian. Có thể nói đây là phương hướng nghiên cứu có thể mang lại nhiều thuận lợi cũng nhợi ích cho xã hội.

* 1. **Một số nghiên cứu trước đây**
* Có rất nhiều nghiên cứu sử dụng các phương pháp khác nhau để giải quyết bài toán nhận dạng khuôn mặt, hai cách phổ biến nhất là thường được sử dụng là haarCascade và nhận dạng dựa trên xét tổng thể của khuôn mặt (Appearance based face recognition). Nhìn chung, các phương pháp đều có ưu nhược điểm nhằm giải quyết bài toán nhận dạng mặt người
  + Những năm 1960, Woody Bledsoe, Helen Chan Wolf, và Charles Bisson đã cho ra đời hệ thống nhận dạng khuôn mặt bán tự động đầu tiên. Hệ thống yêu cầu người quản trị xác định vị trí các đặc trưng như: mắt, tai, mũi và miệng trên các tấm ảnh. Sau đó, hệ thống sẽ tính khoảng cách và tỷ lệ đến một điểm tham chiếu chung. Cuối cùng là so sánh với kho dữ liệu lưu trữ.
  + Năm 1970, Goldstein, Harmon và Lesk sử dụng 21 dấu hiệu chủ quan như: màu tóc và độ dày môi để nhận dạng tự động. Giống như ở giải pháp trước, các phép đo và các vị trí được tính bằng tay nên đòi hỏi nhiều thời gian.
  + Năm 1988, Kirby và Sirovich áp dụng phương pháp phân tích thành phần chính (PCA), phương pháp này sử dụng chuẩn đại số tuyến tính cho vấn đề nhận dạng Nguyễn Hoàng Phúc – 1111540 Nghiên cứu thư viện OpenCv - Ứng dụng nhận dạng khuôn mặt người Trang 2 khuôn mặt. Đây được coi là cột mốc quan trọng vì nó cho thấy ít hơn một trăm giá trị cần thiết để nhận dạng một khuôn mặt được chuẩn hoá.
  + Năm 1991, Turk và Pentland sử dụng thuật toán eigenfaces để nhận dạng khuôn mặt tự động trên thời gian thực. Dù phương pháp này một phần nào bi hạn chế bởi yếu tố môi trường nhưng nó cũng góp phần đáng kể cho sự phát triển của công nghệ nhận dạng khuôn mặt.
  + Năm 2001, Paul Viola và Michael J. Jones đã cho ra đời đặc trưng Haar-Like, đó là những đặc trưng ảnh số cũng với phương pháp tích phân ảnh (integral images) để nâng cao khả năng rút trích đặc trưng trên thời gian thực.
  + Tháng 7-2003, Trần Phước Long và Nguyễn Văn Lượng dùng mạng Neural để dò tìm khuôn mặt trên ảnh, kết hợp phương pháp phân tích thành phần chính và biến đổi Cosine rời rạc để rút ra Vector đặc trưng làm đầu vào cho hai bộ nhận dạng SVM và HMM. Hệ thống tiếp cận từ các phương pháp học mạnh, nhưng nhiều trường hợp tối ưu cần thử nghiệm nhiều lần với nhiều bộ tham số khác nhau.
  + Năm 2005, Trần Lê Hồng Dũ sử dụng các đặc trưng lòi lõm để phát hiện khuôn mặt trên ảnh nhưng còn hạn chế về độ sáng, quan hệ hình học và các đặc trưng cùng mức.
  + Năm 2007, Lê Hồng Chuyên đã sử dụng mạng Neural để phát hiện khuôn mặt trên ảnh. Hệ thống đã nhận dạng được những khuôn mặt ở các tư thế: thẳng đứng, không thẳng đứng hoặc bị che mất một phần. Nhưng kết quả dò tìm không cao trong môi trường ảnh có nhiều người.
  + Năm 2013, Mã Trường Thành thực hiện đề tài “Điều khiển Robot Pioneer P3- DX bám sát đối tượng” đã sử dụng đặc trưng Haar-Like, bộ phân tầng Cascades of Boosted Classifiers và thuật toán Adaboost để nhận dạng đối tượng.
  + Năm 2013, Châu Ngân Khánh đã sử dụng đặc trưng Haar-Like, bộ phân tầng Cascades of Boosted Classifiers và thuật toán Adaboost để nhận dạng đối tượng bằng cách so khớp SIFT. Nhưng cơ sở dữ liệu không bao quát được hết sự thay đổi của khuôn mặt con người trên thời gian thực…

## **1.3 Phạm vi đề tài.**

* Đề tài tập trung nghiên cứu gói thư viện mã nguồn mở OpenCV để giải quyết bài toán quan sát, phát hiện và nhận dạng đối tượng theo thời gian thực bằng đặc trưng HaarLike Features – Cascades of Boosted Classifiers.
* Triển khai nghiên cứu bằng việc thu thập ảnh, huấn luyện và giải quyết bài toán thực tế về nhận dạng khuôn mặt người bằng cách lưu các hoạt động của đối tượng dưới dạng ảnh và video thông qua Webcam.

## **1.4 Phương pháp nghiên cứu**

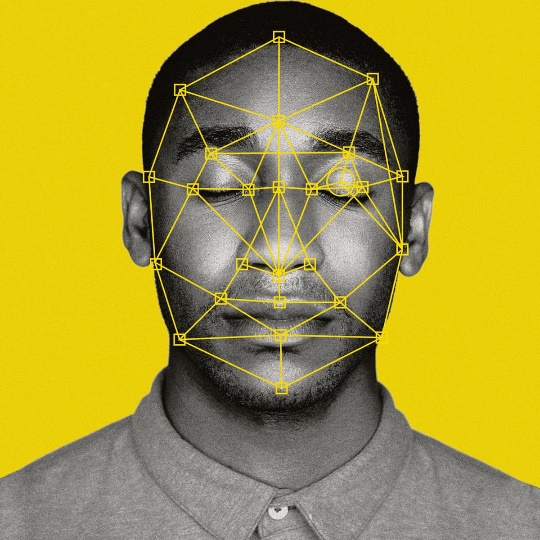
* + Nghiên cứu gói thư viện OpenCv – cụ thể là gói thư viện Emgu với ngôn ngữ được sử dụng là C# (C Sharp).
  + Sử dụng các hàm trong thư viện OpenCV bằng việc kết hợp đặc trưng HaarLike với thuật toán Adaboost và mô hình Cascades of Boosted Classifiers để thực hiện dò tìm đối tượng trên mặt phẳng ảnh theo thời gian thực để triển khai giải quyết bài toán đặt ra.

## **1.5 Bố Cục và nội dung**

* Bài báo cáo nghiên cứu về việc nhận dạng khuôn mặt người với OpenCV với những nội dung sau:
  + **CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI**: Giới thiệu tổng quan về đề tài, xác định vấn đề, bối cảnh và nội dung bài toán cần giải quyết. Trình bày về lịch sử giải quyết vấn đề, các nghiên cứu và thành quả của những người đi trước trong cùng lĩnh vực, phạm vi và phương pháp nghiên cứu đề tài.
  + **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT:** Giới thiệu về bài toán dò tìm khuôn mặt, đặc trưng Haar-Like Features, thuật toán AdaBoost và bộ phân tầng Cascade of Boosted Classifier. Tổng quan về gói thư viện OpenCV để nghiên cứu về nhận dạng mặt người, thư viện Emgu nền tảng của OpenCV trên nền ngôn ngữ C# và hệ thống gửi tin nhắn GSMComm.
  + **CHƯƠNG 3,4,5: NỘI DUNG VÀ KẾT QUẢ NGHIÊN CỨU:** Thể hiện cách giải quyết bài toán thông qua hệ thống thực nghiệm. Xây dựng một bộ huấn luyện nhằm nhận dạng khuôn mặt người trên mặt phẳng ảnh. Nêu lên các thiết kế về chương trình và trình bày về thực nghiệm chương trình kèm theo hướng dẫn

**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

**2.1 Giới thiệu về bài toán dò tìm khuôn mặt**



Hình 1.Dò tìm khuôn mặt

Bài toán dò tìm khuôn mặt (face detection) là một trong những bài toán quan trọng trong lĩnh vực thị giác máy tính và trí tuệ nhân tạo. Nó đặt ra mục tiêu nhận diện và phát hiện khuôn mặt trong một hình ảnh hoặc một đoạn video.

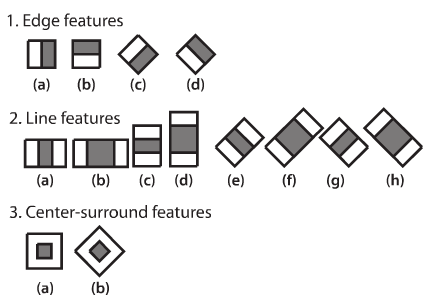
Bài toán này thường được giải quyết thông qua việc sử dụng các thuật toán học máy và deep learning để học và tìm kiếm các đặc trưng của khuôn mặt, ví dụ như hình dạng, màu sắc, kích thước và đặc điểm hình học khác.

Các ứng dụng của bài toán dò tìm khuôn mặt rất đa dạng, từ nhận dạng khuôn mặt để mở khóa điện thoại, nhận diện khuôn mặt để đăng nhập vào các ứng dụng, đến giám sát an ninh và phát hiện tội phạm. Tuy nhiên, bài toán này còn đối mặt với nhiều thách thức như sự biến đổi về ánh sáng, góc nhìn và vật liệu, cũng như sự đa dạng của các khuôn mặt trong thực tế.

**2.2 Đặc trưng Haar-Like Features**

Đặc trưng Haar-Like Features là một trong những phương pháp được sử dụng trong bài toán dò tìm khuôn mặt. Phương pháp này được đề xuất bởi Viola và Jones vào năm 2001 và đã trở thành một trong những phương pháp phổ biến trong lĩnh vực nhận dạng khuôn mặt.

Trong bài báo của tác giả *Paul Viola và Michael Jones* đã trình bày một phương pháp mới và nhanh hơn để xử lý hình ảnh và phát hiện khuôn mặt bằng cách sử dụng các đặc điểm hình chữ nhật như hình dưới đây. Các **đặc điểm hình chữ nhật** tương tự như nhân được sử dụng để phát hiện các đặc điểm khác nhau của khuôn mặt như mắt và các nốt như trong hình minh họa.



*Hình 2. Đặc trưng Haar-Like*

Các tính năng hình chữ nhật được chạy lần lượt trên hình ảnh và tổng số pixel nằm trong phần màu trắng được trừ cho tổng số pixel nằm trong phần màu đen.

Trong đó:

Edge features: Đặc trưng cạnh

Line features: Đặc trưng đường

Center-suround features: Đặc trưng trung tâm

Trong hình minh họa bên dưới, đặc điểm hình chữ nhật đầu tiên đang tính toán cho sự khác biệt về cường độ giữa vùng mắt và vùng má trên khuôn mặt. Và đặc điểm hình chữ nhật thứ hai là đo sự chênh lệchvề cường độ giữa hai vùng mắt và sống mũi. Bộ lọc Haar chỉ có thể nhìn cụ thể vào một vùng trong cửa sổ để nhận diện.



*Hình 3. Ví dụ*

* 1. **Trích đặc trưng Haar-Like từ ảnh**

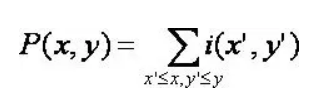
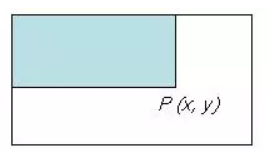
Đặc trưng Haar-like sẽ diễn đạt được tri thức về các đối tượng trong ảnh (bởi vì nó biểu diễn mối liên hệ giữa các bộ phận của đối tượng), điều mà bản thân từng điểm ảnh không diễn đạt được – dựa trên việc co giãn của khung cửa sổ tìm kiếm. Để tính giá trị các đặc trưng Haar-Like, ta tính sự chênh lệch giữa tổng của các pixel của các vùng đen và các vùng trắng như trong công thức:

****

Vì vậy chúng ta sẽ thấy rằng, với tổng các pixel trên ảnh *(pixel vùng trắng và pixel vùng đen)* sẽ cho ta các giá trị của đặc trưng Haar – Like. Nhưng để tính toán những giá trị của các đặc trưng Haar-like cho tất cả các vị trí trên ảnh đòi hỏi chi phí tính toán khá lớn, dẫn đến không thể đáp ứng được cho các ứng dụng đòi hỏi tính thời gian thực*(run-time)* trực tiếp từ Camera *(hay Webcam trên Laptop)*. Do đó Viola và Jones đề xuất một khái niệm gọi là ***“tich phân ảnh”*** *(Integral Image)* để tính toán nhanh cho các đặc trưng cơ bản của Haar-like.

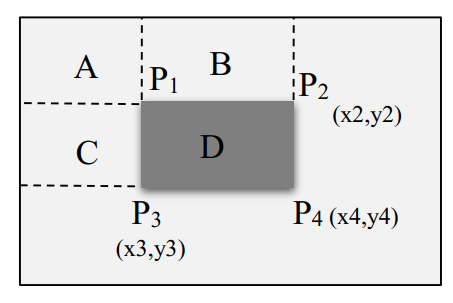
Tích phân ảnh được định nghĩa là 1 mảng 2 chiều có kích thước bằng kích thước ảnh cần tìm.

Mỗi phần tử trong mảng có giá trị bằng tổng các phần từ ở phái trên và bên trái của nó. Bắt đầu từ vị trí trên bên trái và kết thúc tại vị trí dưới bên phải của ảnh, tính lần lượt từ trái qua phải và từ trên xuống dưới.



*Hình 4. Hình minh họa tích phân ảnh và công thức tính tích phân ảnh*

Sau khi tính tích phân ảnh, ta tính tổng điểm ảnh theo các sau:



*Hình 5.Tính nhanh tổng điểm ảnh trong vùng D*

Tính điểm D:

Tổng = A + B + C + D

D =Tổng – (A+B) – (A+C) + A

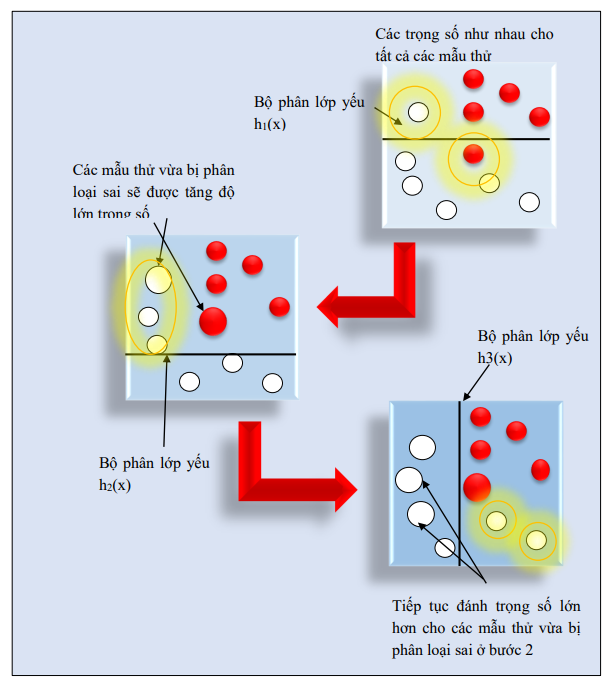
Với A + B + C + D chính là giá trị tại điểm P4 trên đảo hàm ảnh , tương tự như vậy A+B là giá trị tại điểm P2, A+C là giá trị tại điểm P3, và A là giá trị tại điểm P1.  
 Vậy ta có thể viết lại biểu thức tính D ở trên như sau:

D= P4-P2-P3+P1

D= (x4,y4)-(x2,y2)-(x3,y3)+(x1,y1)

## **2.4 Thuật toán AdaBoost**

Thuật toán AdaBoost (Adaptive Boosting) là một trong những thuật toán quan trọng và phổ biến nhất được sử dụng trong bài toán dò tìm khuôn mặt. AdaBoost kết hợp nhiều bộ phân loại yếu thành một bộ phân loại mạnh. Mỗi bộ phân loại yếu được huấn luyện để phân loại một tập con của dữ liệu và nhận được một trọng số ứng với độ chính xác của nó. AdaBoost sẽ tăng trọng số của các dữ liệu bị phân loại sai và giảm trọng số của các dữ liệu được phân loại đúng. Sau đó, các bộ phân loại yếu sẽ được sử dụng để tạo ra một bộ phân loại mạnh bằng cách kết hợp dự đoán của chúng theo một quy tắc đa số.



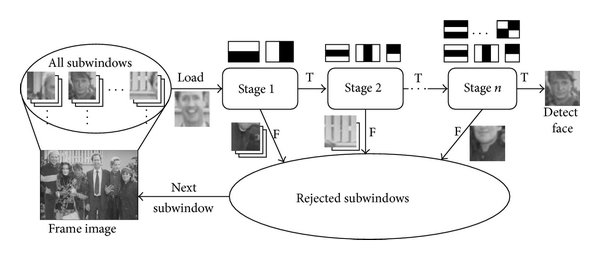
*Hình 6. Kỹ thuật Adaboost*

## **2.5 Bộ phân tầng Cascade of Boosted Classifier**

Bộ phân tầng Cascade of Boosted Classifier là một phương pháp dò tìm khuôn mặt cải tiến dựa trên AdaBoost. Nó bao gồm nhiều giai đoạn, mỗi giai đoạn sử dụng một bộ phân loại được huấn luyện để loại bỏ các vùng không phải là khuôn mặt. Ở mỗi giai đoạn, hình ảnh được chia thành nhiều vùng nhỏ hơn, các vùng này được xem xét bởi các bộ phân loại với độ chính xác khác nhau. Các vùng bị loại bỏ không phải là khuôn mặt sẽ không được xem xét trong các giai đoạn tiếp theo. Các khuôn mặt được phát hiện trong các giai đoạn sau sẽ được xác định chính xác hơn vì chỉ cần xem xét các vùng nhỏ hơn và đặt ưu tiên cho các vùng có khả năng cao hơn chứ không cần phân tích toàn bộ hình ảnh.

Bộ phân tầng Cascade of Boosted Classifier có nhiều ưu điểm, bao gồm tăng tốc độ dò tìm, cải thiện độ chính xác và giảm độ phức tạp tính toán. Tuy nhiên, cần phải cân nhắc kỹ lưỡng trong việc lựa chọn các tham số và bộ phân loại để đạt được hiệu quả cao nhất.

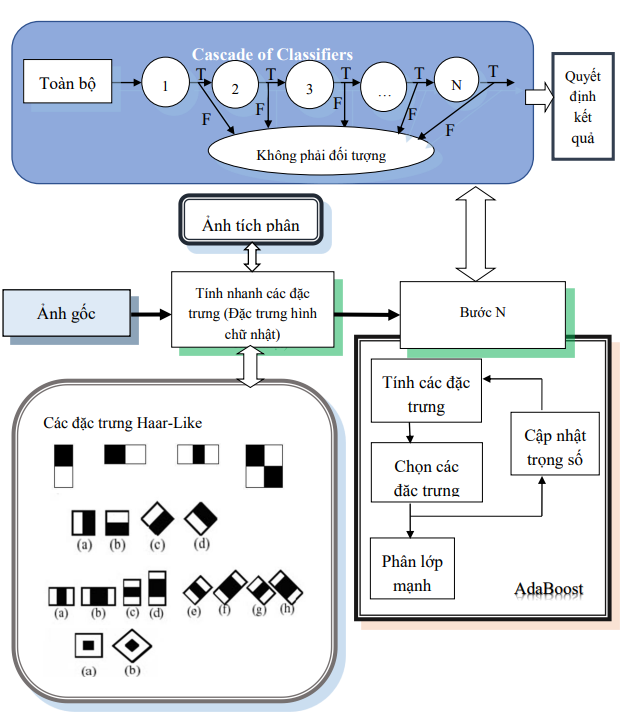
Để được nhận dạng là khuôn mặt, các cửa sổ ảnh nhỏ sẽ phải đi qua hết các stage của bộ phân loại. Các bộ lọc ở mỗi stage đằng sau sẽ được huấn luyện bằng các mẫu không có đối tượng mà các stage trước nó nhận định sai, do đó sự kết hợp các lớp lọc này sẽ giúp hệ thống giảm thiếu khả năng nhận định sai.



*Hình 7.Bộ phân tầng Haar Cascade*

* **Bước 1:** Hình ảnh (đã được gửi đến bộ phân loại) được **chia thành các phần nhỏ**(hoặc các cửa sổ con như trong hình minh họa).
* **Bước 2:** Chúng tôi đặt N bộ dò theo cách xếp tầng trong đó mỗi bộ phát hiện sự kết hợp của các loại **đặc trưng khác nhau** từ các hình ảnh (ví dụ: đường thẳng, cạnh, hình tròn, hình vuông) được truyền qua. Giả sử khi việc trích xuất đối tượng địa lý được thực hiện, mỗi phần phụ được gán một **giá trị tin cậy**.
* **Bước 3:** Hình ảnh (hoặc hình ảnh phụ) có độ tin cậy cao nhất được phát hiện dưới dạng khuôn mặt và được gửi đến **bộ tích lũy** trong khi phần còn lại bị từ chối. Do đó, Cascade tìm nạp khung hình / hình ảnh tiếp theo nếu còn lại và bắt đầu lại quá trình.

## **2.6 Định vị đối tượng**

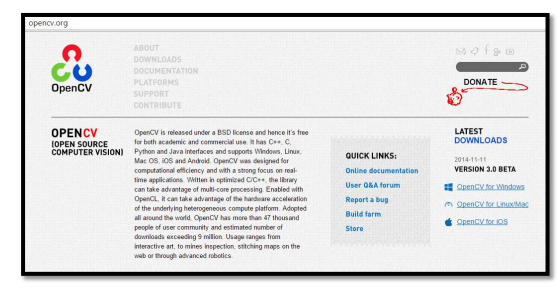


*Hình 8. Dò tìm đối tượng với Haar-Like Features và Cascade of Boosted Classifiers*

* Image Kernels là 1 ma trận nhỏ dùng để sử dụng các hiệu ứng chỉnh sửa. Trong mỗi ô của ma trận sẽ chứa giá trị tờ 0 đến 255 để biểu thị giá trị sang tối của từng pixel ảnh trong ảnh chính. Từ đó, người dùng sẽ áp dụng các công thức tính toán để làm mờ, làm rõ nét bằng các phép nhân ma trận. Hơn thế nữa, chúng ta có thể dùng phép tích phân ảnh để sử dụng trong việc nhận dạng khuôn mặt.
* Trình tự hoạt động: Sau khi quét ảnh từ camera, hệ thống sẽ chuyển hình ảnh quét được thành ảnh đen trắng. Sau đó chuyển hình ảnh đen trắng đã có thành dạng Image Kernels. Sau đó hệ thống trích các đặc trưng Haar-Like từ ảnh thông qua tích phân ảnh và sử dụng các hình chữ nhật đặc trưng để phát hiện gương mặt trong khung hình. Sau đó được đưa qua bộ điều chỉnh AdaBoost để loại bỏ bớt các đặc trưng không cần thiết. Các đặc trưng còn lại sẽ được đưa qua bộ phân lớp để quyết định xem có phải mặt người hay không.

## **2.7 Tổng quan về thư viện OpenCV để nghiên cứu về nhận dạng mặt người**

OpenCV (Open Source Computer Vision Library) là một thư viện mã nguồn mở cung cấp nhiều công cụ và thuật toán để xử lý ảnh và thị giác máy tính. OpenCV được viết bằng C++, nhưng cũng có thể sử dụng các ngôn ngữ khác như Python và Java thông qua các giao diện lập trình ứng dụng (API).



*Hình 9. Thư viện OpenCV*

OpenCV cung cấp nhiều tính năng để hỗ trợ nhận dạng mặt người, bao gồm:

1. Xử lý ảnh: OpenCV cung cấp các tính năng để xử lý ảnh như đọc, ghi, chuyển đổi, cắt, phóng to, thu nhỏ ảnh, làm mịn ảnh, tăng độ tương phản và cân bằng sáng tối.
2. Dò tìm khuôn mặt: OpenCV cung cấp các thuật toán để dò tìm khuôn mặt như Haar Cascades, LBP (Local Binary Pattern), và HOG (Histogram of Oriented Gradients).
3. Nhận dạng khuôn mặt: OpenCV cung cấp các thuật toán để nhận dạng khuôn mặt như Eigenfaces, Fisherfaces, và Local Binary Patterns Histograms (LBPH).
4. Ghi chú và vẽ hình trên ảnh: OpenCV cung cấp các công cụ để ghi chú và vẽ hình trên ảnh để đánh dấu khuôn mặt đã được tìm thấy hoặc hiển thị kết quả của thuật toán.

OpenCV cũng có thể được sử dụng để kết hợp nhiều thuật toán nhận dạng khuôn mặt và tạo ra một hệ thống hoàn chỉnh để nhận dạng khuôn mặt trong thời gian thực.

Việc sử dụng OpenCV để nghiên cứu về nhận dạng khuôn mặt là rất phổ biến vì thư viện này cung cấp nhiều tính năng và thuật toán cho bài toán này và có thể được sử dụng trên nhiều nền tảng khác nhau như Windows, Linux, MacOS, Android và iOS.

**2.8 Thư viện EmguCV**

Emgu CV là gói thư viện được viết hoàn toàn bằng ngôn ngữ C#. Nó bao gồm các chức năng của OpenCV, cho phép gọi các hàm của OpenCV từ Visual Studio bằng các ngôn ngữ .Net như: C#, VB, VC++, v.v... Vì Visual Studio/.Net không thể gọi trực tiếp các hàm của OpenCV nên Emgu là giải pháp cho vấn đề này. Emgu có thể chạy trên Windows, Linux, iOS, Mac OS X, Android và Windows Phone.



*Hình 10. Thư viện EmguCV*

**CHƯƠNG 3: YÊU CẦU CHỨC NĂNG VÀ MÔ HÌNH HÓA USECASE**

## **3.1 Yêu cầu chức năng**

**-Quản trị viên:**

**+** Đăng nhập

+ Thêm tài khoản

+ Sửa tài khoản

+ Xóa tài khoản

+ Lấy lại mật khẩu

+ Xử lý yêu cầu

**-Quản lý:**

+ Đăng nhập

+ Lấy lại mật khẩu

+ Quản lý nhân viên (thêm, sửa, xóa, tìm kiếm)

+ Xem báo cáo theo ngày và theo tháng

+ Gửi báo cáo qua email

+ Xuất báo cáo

+ Chấm công

+ Xem bảng lương

+ Gửi bảng lương qua email

+ Xuất bảng lương

+ Gửi yêu cầu

+ Quản lý nhân viên đi làm muộn (thêm, sửa, xóa, tìm kiếm)

## **3.2 Mô hình hóa CSDL**

### 3.2.1 Các yêu cầu về dữ liệu

Để xây dựng 1 hệ thống chấm công tự động cần có các yêu cầu sau:

- Hệ thống lưu trữ thông tin các tài khoản admin và nhân viên bao gồm: tên đăng nhập, mật khẩu, email, ảnh đại diện, chức vụ. phòng ban.

- Danh sách nhân viên: Tên nhân viên, id, ngày sinh,lương cơ bản của 1 ca làm việc, ảnh hổ sơ.

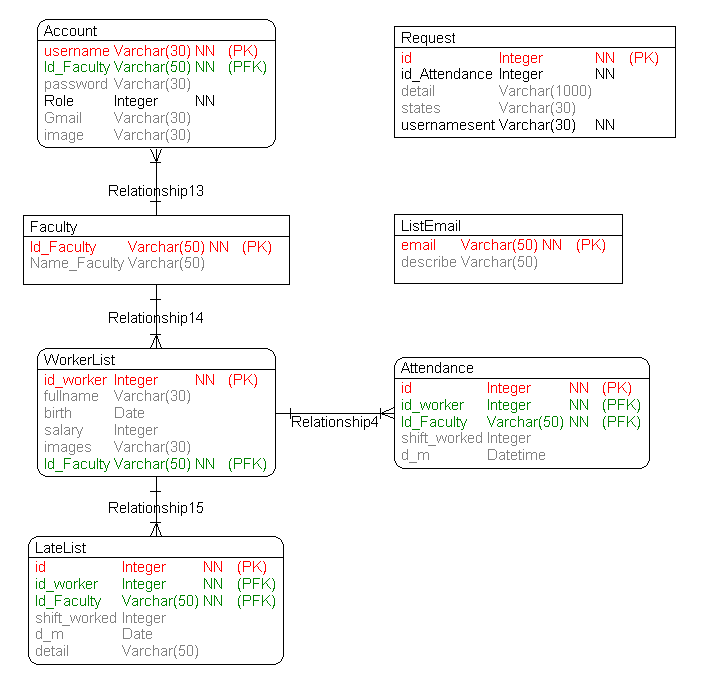
- Danh sách phòng ban: id, tên phòng ban.

-Danh sách email: tài khoản email, mô tả tài khoản.

-Danh sách điểm danh:id, id công nhân, id phòng ban, ca, ngày tháng.

-Danh sách đi muộn : id, id công nhân , id phòng ban , ca , ngày , lý do đi muộn

### 3.2.2 Biểu đồ thực thể liên kết

****

### 3.2.3 Mô tả chi tiết các bảng CSDL

-Bảng Account

|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| Username | Varchar(30) | Tên đăng nhập |
| Password | Varchar(30) | Mật khẩu |
| Id\_faculty | Varchar(50) | Id của phòng ban |
| Role | Int | Chức vụ |
| Gmail | Varchar(30) | Gmail cá nhân |
| Image | Varchar(30) | Ảnh đại diện |

-Bảng Faculty

|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| Id\_Faculty | Varchar(50) | Id của phòng ban |
| Name\_Falculty | Varchar(50) | Tên phòng ban |

-Bảng WorkerList

|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| Id | Int | Id nhân viên |
| Fullname | Varchar(30) | Tên nhân viên |
| Birth | Date | Ngày sinh |
| Salary | Int | Lương cơ bản của 1ca làm việc |
| Image | Varchar(30) | Ảnh hồ sơ |
| Id-Faculty | Varchar(50) | Id phòng ban |

-Bảng Attendance

|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| Id | Int | Id điểm danh |
| Id\_worker | Int | Id nhân viên |
| Id\_Faculty | Varchar(50) | Id phòng ban |
| Shift\_worked | Int | Ca |
| D\_m | DateTime | Ngày, giờ điểm danh |

-Bảng ListEmail

|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| Email | Varchar(50) | Email |
| Describe | Varchar(50) | Mô tả của email |

-Bảng LateList

|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| Id | Int | Id điểm danh |
| Id\_worker | Int | Id nhân viên |
| Id\_Faculty | Varchar(50) | Id phòng ban |
| Shift\_worked | Int | Ca |
| D\_m | Date | Ngày đi muộn |
| Detail | Varchar(50) | Lý do đi muộn |

-Bảng Request

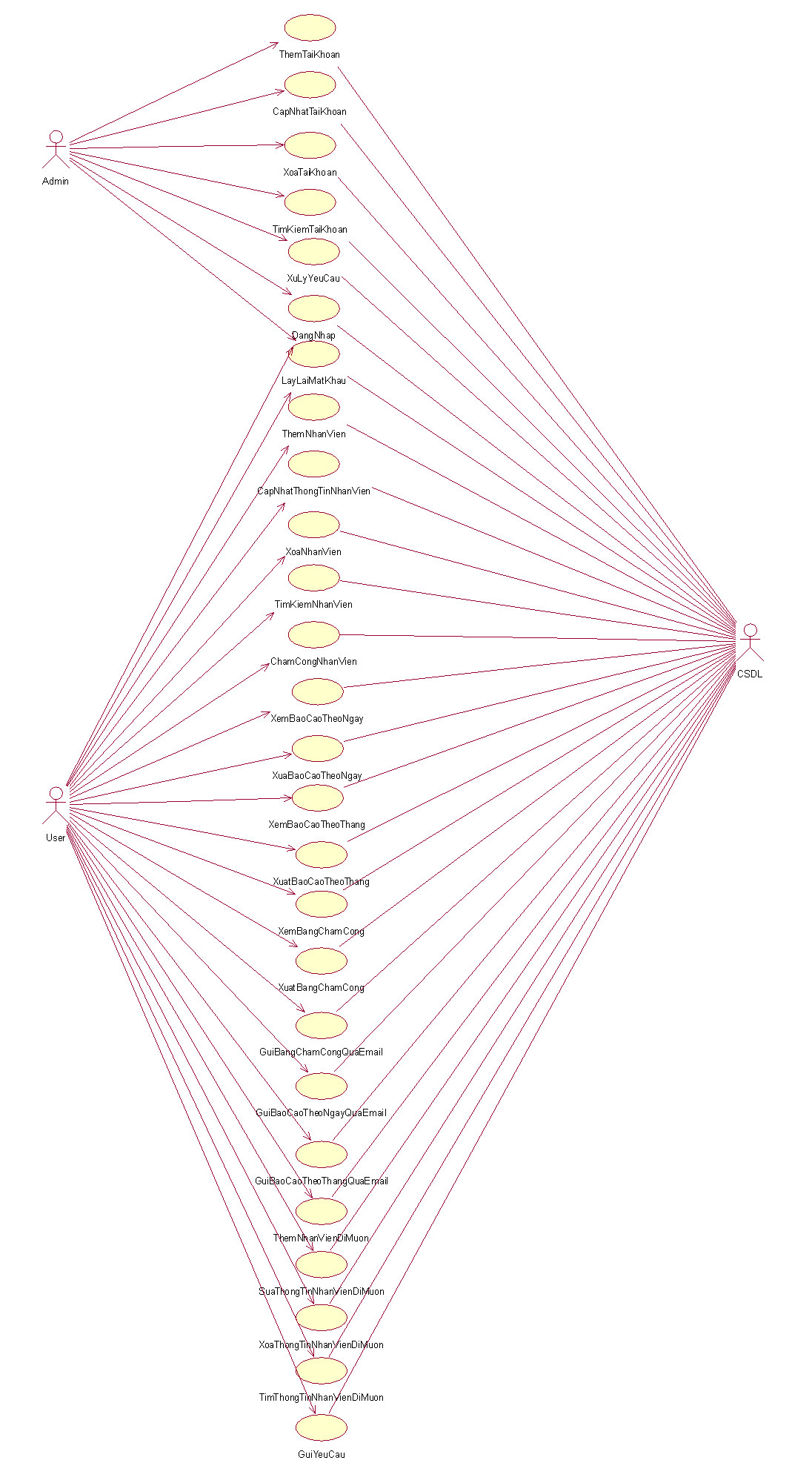
|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| Id | Int | Id yêu cầu |
| Id\_Attendance | Int | Id điểm danh |
| Detail | Varchar(1000) | Chi tiết yêu cầu |
| States | Varchar(30) | Trạng thái yêu cầu |
| Usernamesent | Varchar(30) | Người gửi yêu cầu |

# **CHƯƠNG 4: MÔ TẢ USECASE**

## **4.1 Danh sách Usecase**

|  |  |  |
| --- | --- | --- |
| **Tên UC** | **Mã UC** | **Chức năng** |
| Đăng nhập | UC01 | Đăng nhập vào hệ thống |
| Thêm tài khoản | UC02 | Thêm tài khoản |
| Cập nhật tài khoản | UC03 | Sửa thông tin tài khoản |
| Xóa tài khoản | UC04 | Xóa tài khoản |
| Tìm tài khoản | UC05 | Tìm kiếm thông tin tài khoản |
| Thêm nhân viên | UC06 | Thêm thông tin nhân viên |
| Cập nhật nhân viên | UC07 | Sửa thông tin nhân viên |
| Xóa nhân viên | UC08 | Xóa thông tin nhân viên |
| Tìm kiếm nhân viên | UC09 | Tìm kiếm thông tin nhân viên |
| Chấm công | UC10 | Chấm công nhân viên |
| Xem báo cáo theo ngày | UC11 | Xem danh sách nhân viên đi làm đúng giờ, đi làm muộn và vắng mặt theo ngày |
| Xuất báo cáo theo ngày | UC12 | Xuất danh sách nhân viên đi làm đúng giờ, đi làm muộn và vắng mặt theo ngày |
| Xem báo cáo theo tháng | UC13 | Xem danh sách nhân viên đi làm đúng giờ, đi làm muộn và vắng mặt theo tháng |
| Xuất báo cáo theo tháng | UC14 | Xuất file excel danh sách nhân viên đi làm đúng giờ, đi làm muộn và vắng mặt theo tháng |
| Lấy lại mật khẩu | UC15 | Lấy lại mật khẩu |
| Xem bảng lương | UC16 | Xem bảng lương |
| Xuất bảng chấm công | UC17 | Xuất file excel chứa bảng lương |
| Gửi báo cáo theo ngày | UC18 | Gửi file danh sách nhân viên đi làm đúng giờ, đi làm muộn và vắng mặt theo ngày |
| Gửi báo cáo theo tháng | UC19 | Gửi file danh sách nhân viên đi làm đúng giờ, đi làm muộn và vắng mặt theo tháng |
| Gửi bảng lương | UC20 | Gửi file bảng lương qua email |
| Thêm nhân viên đi muộn | UC21 | Thêm thông tin nhân viên đi làm muộn |
| Sửa thông tin nhân viên đi muộn | UC22 | Sửa thông tin nhân viên đi làm muộn |
| Xóa thông tin nhân viên đi muộn | UC23 | Xóa thông tin nhân viên đi làm muộn |
| Tìm thông tin nhân viên đi muộn | UC24 | Tìm kiếm thông tin nhân viên đi muộn |
| Gửi yêu cầu | UC25 | Gửi yêu cầu đến admin để xóa điểm danh |
| Xử lý yêu cầu | UC26 | Xử lý các yêu cầu được gửi đến |

## **4.2 Sơ đồ UC**

****

## **4.3 Mô tả chi tiết UC**

### UC01: Đăng nhập

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Login | **Code** | UC01 |
| **Description** | Cho phép actor đăng nhập vào hệ thống | | |
| **Actor** | Admin/Staff | **Trigger** | Actor bấm nút login |
| **Pre-condition** | Điền đủ vào ô tài khoản và mật khẩu | | |
| **Post condition** | Chuyển tới trang menu với role tương ứng | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Login thành công** | | | |
| 1 | Actor nhập tên đăng nhập/mật khẩu và click vào button Login trên trang đăng nhập ([SC01](#_SC01:_My_Profile)). |  |  |
|  |  | 2 | Kiểm tra tên đăng nhập/mật khẩu là chính xác sau đó chuyển tới trang default role tương ứng ([SC03](#_SC03:_My_Group), [SC02](#_SC02:_List_users)...) |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

|  |  |
| --- | --- |
| MS01 | “Tên Đăng Nhập và/hoặc Mật Khẩu của bạn không chính xác. Vui lòng kiểm tra và thử lại.”  Message thông báo khi actor nhập sai tên đăng nhập/mật khẩu |

### UC02: Thêm tài khoản

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Thêm tài khoản | **Code** | UC02 |
| **Description** | Cho phép actor thêm tài khoản staff/admin trong công ty | | |
| **Actor** | Admin | **Trigger** | Actor click button **[Add]** trên màn hình Quản lý tài khoản |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống | | |
| **Post condition** | Thêm thành công tài khoản | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Thêm tài khoản thành công** | | | |
| 1 | Actor click nút **[Add]** trên màn hình Account Manage |  |  |
|  |  | 2 | Hệ thống Load trang Thêm mới thông tin tài khoản |
| 3 | Nhập vào các thông tin được yêu cầu. |  |  |
|  |  | 4 | Validation các thông tin nhập vào. |
| 5 | Sửa lại những thông tin chưa đúng (nếu hệ thông check validation chưa đúng). |  |  |
|  |  | 6 | Lưu thông tin người dùng mới vào bảng Account trên CSDL, thông báo thành công và chuyển sang trang Quản lý tài khoản |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message: N/A**

### UC03: Cập nhật tài khoản

**Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Cập nhật thông tin tài khoản | **Code** | UC03 |
| **Description** | Cho phép actor cập nhật thông tin tài khoản trong hệ thống. | | |
| **Actor** | Admin | **Trigger** | Actor click button **[Edit]** ở màn hình Quản lý tài khoản |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống | | |
| **Post condition** | Cập nhật tài khoản thành công | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Cập nhật thông tin tài khoản thành công** | | | |
| 1 | Actor chọn 1 tài khoản rồi click button **[Edit]** ở màn hình Quản lý tài khoản |  |  |
|  |  | 2 | Hệ thống lấy thông tin của tài khoản cần sửa trong bảng Account trên CSDL rồi hiển thị lên trang Thông tin tài khoản |
| 3 | Cập nhật những thông tin cần thiết rồi ấn button **[Save]** |  |  |
|  |  | 4 | Validation các thông tin nhập vào. |
| 5 | Sửa lại những thông tin chưa đúng (nếu hệ thông check validation chưa đúng). |  |  |
|  |  | 6 | Hệ thống update thông tin tài khoản trong bảng WorkerList trên CSDL. Hệ thống thông báo thành công và chuyển về màn hình Quản lý tài khoản |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

**N/A**

### UC04: Xóa tài khoản

**Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Xóa tài khoản | **Code** | UC04 |
| **Description** | Cho phép actor xóa tài khoản trong công ty | | |
| **Actor** | Admin | **Trigger** | Actor click button **[Del]** ở màn hình Quản lý tài khoản |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống | | |
| **Post condition** | Xóa tài khoản thành công | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Xóa tài khoản thành công** | | | |
| 1 | Actor chọn 1 tài khoản rồi click button **[Del]** ở màn hình Quản lý tài khoản |  |  |
|  |  | 2 | Hệ thống hiển thị thông báo “Are you sure ?” |
| 3 | Actor click **[Yes]** |  |  |
|  |  | 4 | Hệ thống xóa thông tin tài khoản đó ra khỏi bảng Account và những bảng liên quan trên CSDL. Thông báo thành công và chuyển sang trang Quản lý tài khoản |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

|  |  |
| --- | --- |
| MS02 | “Xác nhận xóa  Message thông báo khi actor nhập xóa |

### UC05: Tìm kiếm tài khoản

**Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Tìm kiếm tài khoản | **Code** | UC05 |
| **Description** | Cho phép actor tìm kiếm tài khoản bằng tên đăng nhập hoặc email | | |
| **Actor** | Admin | **Trigger** | Actor click button **[Search]** bên cạnh thanh tìm kiếmtrên màn hình Quản lý tài khoản |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống | | |
| **Post condition** | Tìm kiếm tài khoản thành công | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Tìm kiếm tài khoản thành công** | | | |
| 1 | Actor nhập tên đăng nhập hoặc email của tài khoản cần tìm vào thanh tìm kiếm rồi click button **[Search]** trênmàn hình Quản lý tài khoản |  |  |
|  |  | 2 | Hệ thống tìm tài khoản có tên đăng nhập/ email trùng với tên đăng nhập/ email actor nhập trong bảng Account trên CSDL rồi hiển thị lên list tài khoản trên màn hình Quản lý tài khoản |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

**N/A**

### UC06: Thêm nhân viên

**Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Quản lý nhân viên | **Code** | UC06 |
| **Description** | Cho phép actor quản lý nhân viên trong công ty | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Add]** trên màn hình Quản lý nhân viên |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống | | |
| **Post condition** | Thêm thành công nhân viên | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Thêm nhân viên thành công** | | | |
| 1 | Actor chọn nút **[Add]** màn hình Worker Manage |  |  |
|  |  | 2 | Hệ thống Load trang Thêm mới thông tin người dùng |
| 3 | Nhập vào các thông tin được yêu cầu. |  |  |
|  |  | 4 | Validation các thông tin nhập vào. |
| 5 | Sửa lại những thông tin chưa đúng (nếu hệ thông check validation chưa đúng). |  |  |
|  |  | 6 | Lưu thông tin người dùng mới vào bảng WorkerList trên CSDL, thông báo thành công và chuyển sang trang Quản lý nhân viên |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

**N/A**

### UC07: Cập nhật thông tin nhân viên

**Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Cập nhật thông tin nhân viên | **Code** | UC07 |
| **Description** | Cho phép actor cập nhật thông nhân viên trong phòng ban. | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Edit]** trên màn hình Quản lý nhân viên |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống | | |
| **Post condition** | Cập nhật thông tin nhân viên thành công | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Cập nhật thông tin nhân viên thành công** | | | |
| 1 | Actor chọn 1 nhân viên rồi click button **[Edit]** ở hiển thị trên màn hình Quản lý nhân viên |  |  |
|  |  | 2 | Hệ thống lấy thông tin của nhân viên cần sửa trong bảng WorkerList trên CSDL rồi hiển thị lên trang Thông tin nhân viên |
| 3 | Cập nhật những thông tin cần thiết rồi ấn button **[Save]** |  |  |
|  |  | 4 | Validation các thông tin nhập vào. |
| 5 | Sửa lại những thông tin chưa đúng (nếu hệ thông check validation chưa đúng). |  |  |
|  |  | 6 | Hệ thống update thông tin nhân viên trong bảng WorkerList trên CSDL. Hệ thống thông báo thành công và chuyển về màn hình Quản lý nhân viên |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

**N/A**

### UC08: Xóa nhân viên

**Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Xóa nhân viên | **Code** | UC08 |
| **Description** | Cho phép actor xóa nhân viên trong công ty | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Del]** trên màn hình Quản lý nhân viên |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống | | |
| **Post condition** | Xóa nhân viên thành công | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Xóa nhân viên thành công** | | | |
| 1 | Actor chọn 1 nhân viên và click button **[Del]** ở trên màn hình Quản lý nhân viên |  |  |
|  |  | 2 | Hệ thống hiển thị thông báo “Are you sure ?” |
| 3 | Actor click **[Yes]** |  |  |
|  |  | 4 | Hệ thống xóa thông tin nhân viên đó ra khỏi bảng WorkerList và những bảng liên quan trên CSDL. Thông báo thành công và chuyển sang trang Quản lý nhân viên |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

**N/A**

### UC09: Tìm kiếm nhân viên

**Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Tìm kiếm nhân viên | **Code** | UC09 |
| **Description** | Cho phép actor tìm kiếm nhân viên trong phòng ban bằng mã ID | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Search]** bên cạnh thanh tìm kiếmtrên màn hình Quản lý nhân viên |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống | | |
| **Post condition** | Tìm kiếm nhân viên thành công | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Tìm kiếm nhân viên thành công** | | | |
| 1 | Actor nhập mã Id của nhân viên cần tìm vào thanh tìm kiếm rồi click button **[Search]** trênmàn hình Quản lý nhân viên |  |  |
|  |  | 2 | Hệ thống tìm nhân viên có mã ID trùng với ID actor nhập trong bảng WorkerList trên CSDL rồi hiển thị lên list nhân viên trên màn hình Quản lý nhân viên |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

**N/A**

### UC10: Chấm công nhân viên

**Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Chấm công nhân viên | **Code** | UC10 |
| **Description** | Cho phép actor chấm công nhân viên trong phòng ban. | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Attandance]** |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống | | |
| **Post condition** | Actor có được danh sách nhân viên đã chấm công trong ngày | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Danh sách nhân viên đã điểm danh trong ngày** | | | |
| 11 | Actor bấm “Switch” |  |  |
|  |  | 12 | Hệ thống chuyển từ đợt điểm danh ca 1 sang điểm danh ca 2 (ca sáng sang ca chiều) |
| 1 | Actor bấm “Start” |  |  |
|  |  | 2 | Hệ thống lấy list dữ liệu “image” trong bảng WorkerList trong CSDL. |
|  |  | 3 | Hệ thống quét khuôn mặt rồi so sánh trong dữ liệu vừa lấy. |
|  |  | 4 | Hệ thống thêm tên nhân viên và thời gian điểm danh vào danh sách nhân viên đã điểm danh được hiển thị trên màn hình. |
| 5 | Actor bấm “End” |  |  |
|  |  | 7 | Hệ thống hiển thị thông báo kết thúc điểm danh. |
|  |  | 8 | Hệ thống lọc id nhân viên trong danh sách nhân viên đã điểm danh trên màn hình và đưa ra danh sách nhân viên chưa chấm công. |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

**N/A**

### UC11: Xem báo cáo theo ngày

**Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Xem báo cáo theo ngày | **Code** | UC05 |
| **Description** | Cho phép actor tìm kiếm và xem danh sách thống kê nhân viên trong phòng ban đi làm đúng giờ, đi làm muộn và vắng mặt trong 2 ca theo ngày | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Search]** ở màn hình Report\_Day |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống | | |
| **Post condition** | Xem thông tin danh sách nhân viên đi làm đúng giờ, đi làm muộn và vắng mặt thành công | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Xem thông tin danh sách nhân viên đi làm đúng giờ, đi làm muộn và vắng mặt thành công** | | | |
| 1 | Từ màn hình bất kỳ, actor lựa chọn ở left menu “Report” |  |  |
|  |  | 2 | Hệ thống hiển thị giao diện màn hình Report\_Day |
| 3 | Tại màn hình Report\_Day, Actor chọn ngày rồi bấm nút “Search” |  |  |
|  |  | 4 | Hệ thống đọc dữ liệu trong bảng WorkerList, Attendance, LateList trong CSDL có thuộc tính id phòng ban trùng với id phòng ban của actor, d\_m trùng với ngày mà actor đã chọn, chia theo từng ca, ở mỗi ca lấy được 2 danh sách nhân viên đi làm đúng giờ, nhân viên đi làm muộn, từ 2 danh sách trên lọc ra những nhân viên không có trong 2 danh sách đó (danh sách nhân viên vắng mặt). |
|  |  | 5 | Hiển thị ra màn hình 3 danh sách của ca 1, ẩn các danh sách của ca 2 |
| 6 | Actor chọn ca 2 ở Combobox |  |  |
|  |  | 7 | Ẩn các danh sách của ca 1 và hiển thị các danh sách của ca 2 (và ngược lại nếu actor chọn lại ca ở combobox) |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

**N/A**

### UC12: Xuất báo cáo theo ngày

##### **Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Xuất danh sách theo ngày | **Code** | UC11 |
| **Description** | Cho phép actor có thông tin danh sách nhân viên nghỉ/ đi làm theo ngày | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Export]** trong trang Report\_Day |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống, điền thông tin ngày-tháng-năm | | |
| **Post condition** | Xuất báo cáo theo ngày thành công | | |

##### **Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Xuất báo cáo theo ngày thành công** | | | |
| 1 | Ở cửa sổ Report\_Day, sau khi dữ liệu mà actor muốn lưu đã có ở trên màn hình, actor bấm nút “Export” |  |  |
|  |  | 2 | Hệ thống hiện cửa sổ chọn vị trí lưu file báo cáo. |
| 3 | Người dùng chọn vị trí lưu file và tên file được để mặc định: “Report” (được phép chỉnh sửa), rồi bấm lưu |  |  |
|  |  | 4 | Hệ thống xuất file excel báo cáo vào đúng vị trí người dùng đã chọn. Trong file excel, thông tin của 2 ca được chia làm 2 sheet tương ứng, ở mỗi sheet chứa dữ liệu 3 bảng các danh sách nhân viên đi làm đúng giờ, đi làm muộn và vắng mặt. |

##### **Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

##### **System Message**

##### **N/A**

### UC13: Xem báo cáo theo tháng

**Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Xem báo cáo theo tháng | **Code** | UC08 |
| **Description** | Cho phép actor tìm kiếm và xem danh sách nhân viên trong phòng ban về số ca đi làm, đi làm muộn và vắng mặt theo ngày | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Search]** ở màn hình Report­­­­­­­­­­\_Month |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống | | |
| **Post condition** | Xem thông tin danh sách nhân viên trong phòng ban gồm các thông tin về số ca đi làm, đi làm muộn, vắng mặt thành công | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Xem thông tin danh sách nhân viên trong phòng ban gồm các thông tin về số ca đi làm, đi làm muộn, vắng mặt thành công** | | | |
| 1 | Từ màn hình bất kỳ, actor lựa chọn ở left menu “Report”/ “Month” |  |  |
|  |  | 2 | Hệ thống hiển thị giao diện màn hình Report\_Month |
| 3 | Tại màn hình Report\_Month, Actor nhập tháng, năm rồi bấm nút “Search” |  |  |
|  |  | 4 | Hệ thống đọc dữ liệu trong bảngWorkerList, Attendance, LateList trong CSDL có thuộc tính id phòng ban trùng với id phòng ban của actor, thuộc tính d\_m trùng với tháng và năm mà actor đã chọn, từ đó đếm số ca đi làm (là số ca đi làm đúng giờ lấy từ bảng Attendance và số ca đi làm muộn lấy từ bảng LateList), số ca đi làm muộn và số ca ko đi làm (bằng tổng số ca của cả tháng – số ca đi làm đã tính trước đó) của từng nhân viên trong phòng ban, rồi hiển thị danh sách nhân viên với thống kê vừa tính được ra màn hình |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

**N/A**

### UC14: Xuất báo cáo theo tháng

##### **Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Xuất danh sách theo ngày | **Code** | UC12 |
| **Description** | Cho phép actor có thông tin danh sách nhân viên và số ca đi làm theo tháng | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Export]** trong trang Report\_Month |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống, điền thông tin tháng muốn xem | | |
| **Post condition** | Xuất báo cáo tho tháng thành công | | |

##### **Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Xuất báo cáo theo tháng thành công** | | | |
| 1 | Ở cửa sổ Report\_Month, sau khi dữ liệu mà actor muốn lưu đã có ở trên màn hình, actor bấm nút “Export” |  |  |
|  |  | 2 | Hệ thống hiện cửa sổ chọn vị trí lưu file báo cáo. |
| 3 | Người dùng chọn vị trí lưu file và tên file được để mặc định: “Report” (được phép chỉnh sửa), rồi bấm lưu |  |  |
|  |  | 4 | Hệ thống xuất file excel báo cáo vào đúng vị trí người dùng đã chọn. Trong file excel là thông tin danh sách thống kê các ca đi làm của nhân viên mà actor đã tìm kiếm trước đó |

##### **Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

##### **System Message**

##### **N/A**

### UC15: Lấy lại mật khẩu

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Register | **Code** | UC13 |
| **Description** | Cho phép actor đăng ký tài khoản vào hệ thống | | |
| **Actor** | Staff | **Trigger** | Actor bấm nút Cant login |
| **Pre-condition** | Điền đủ vào ô tài khoản và mật khẩu, email | | |
| **Post condition** | Chuyển tới trang đăng nhập | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Đăng ký thành công** | | | |
| 1 | Người dùng truy cập vào trang đăng ký trên hệ thống. |  |  |
| 2 | Người dùng nhập thông tin email, tên đăng nhập và họ và tên, và bấm nút Recovery |  |  |
|  |  | 3 | Hệ thống kiểm tra tính hợp lệ của thông tin người dùng. Nếu thông tin hợp lệ, hệ thống sẽ gửi 1 mã mật khẩu mới qua email |
|  |  | 4 | Hệ thống chuyển đến trang Đăng nhập. |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

|  |  |
| --- | --- |
| MS05 | “Vui lòng điền đủ thông tin cá nhân để lấy lại tài khoản.”  Message thông báo khi actor không nhập đủ thông tin được yêu cầu |
| MS06 | “Lấy mật khẩu thành công”  Message thông báo khi actor điền đủ thông tin và hợp lệ sau khi hệ thống kiểm tra |

### UC16: Xem bảng chấm công

**Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Xem bảng chấm công | **Code** | UC11 |
| **Description** | Cho phép actor xem bảng chấm công của phòng ban theo tháng, năm | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Accountant]** ở left menu |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống | | |
| **Post condition** | Xem bảng chấm công thành công | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Xem bảng lương thành công** | | | |
| 1 | Từ màn hình bất kỳ, actor lựa chọn ở left menu “Accountant” |  |  |
|  |  | 2 | Hệ thống hiển thị giao diện màn hình Accountant với dữ liệu được chọn, nhập từ trước là tháng và năm của hiện tại. |
|  |  | 4 | Hệ thống lấy dữ liệu từ bảng WorkerList, Accountant và LateList trong CSDL để tạo 1 bảng lương của phòng ban gồm id nhân viên, tên nhân viên, số ca của các ngày trong tháng, tổng số ca đi làm, tổng số ca đi muộn, lương của nhân viên tương ứng với tháng, năm đã được chọn, nhập từ trước đó rồi hiển thị bảng lương lên màn hình |
| 3 | Tại màn hình Accountant, Actor thay đổi tháng và năm |  |  |
|  |  | 4 | Khi người thay đổi, hệ thống lấy bảng lương tương ứng với tháng, năm mà actor đã thay đổi rồi hiển thị ra màn hình. |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

**N/A**

### UC17: Xuất bảng chấm công

##### **Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Xuất bảng chấm công | **Code** | UC14 |
| **Description** | Cho phép actor lấy bảng chấm công theo phòng ban và tháng | | |
| **Actor** | Accoutant | **Trigger** | Actor click button **[Export Excel]** trong trang Accountant |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống, điền tên của phòng ban và tháng muốn lấy thông tin | | |
| **Post condition** | Không có | | |

##### **Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Gửi danh sách theo ngày qua email** | | | |
| 1 | Ở màn hình Accountant, sau khi dữ liệu mà actor muốn lưu đã có ở trên màn hình, actor bấm nút “Export” |  |  |
|  |  | 2 | Hệ thống hiện cửa sổ chọn vị trí lưu file báo cáo. |
| 3 | Người dùng chọn vị trí lưu file và tên file được để mặc định: “Report” (được phép chỉnh sửa), rồi bấm lưu |  |  |
|  |  | 4 | Hệ thống xuất file excel báo cáo vào đúng vị trí người dùng đã chọn. Trong file excel là thông tin của bảng lương |

##### **Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

##### **System Message**

**N/A**

### UC18: : Gửi báo cáo theo ngày qua email

##### **Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Gửi báo cáo theo ngày qua email | **Code** | UC15 |
| **Description** | Cho phép actor gửi báo cáo danh sách nhân viên nghỉ/ đi làm theo ngày qua email | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Send]** trong trang Report\_Day |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống, điền thông tin ngày-tháng-năm | | |
| **Post condition** | Gửi báo báo theo ngày qua email thành công | | |

##### **Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Gửi báo cáo theo ngày qua email thành công** | | | |
| 1 | Ở cửa sổ Report\_Day, sau khi đã xuất file báo cáo actor bấm nút “Send” |  |  |
|  |  | 2 | Hệ thống hiển thị cửa số Send lên màn hình |
| 3 | Tại cửa sổ Send, actor chọn email trong danh sách muốn gửi (nếu không có thì nhập email vào thanh input), nhập các thông tin cần thiết |  |  |
| 4 | Actor bấm vào nút “chọn file” |  |  |
|  |  | 5 | Hệ thống mở cửa sổ chọn file |
| 6 | Tại của sổ chọn file, actor chọn file báo cáo đã xuất từ trước (có thể chọn nhiều file) rồi bấm “Open” |  |  |
|  |  | 7 | Hệ thống lấy được danh sách vị trí file muốn gửi, hiển thị lên cửa sổ Send |
| 8 | Tại của sổ Send, actor bấm “Send” |  |  |
|  |  | 9 | Hệ thống sẽ truy cập vào email của công ty sau đó gửi mail đã được nhập/chọn trước chứa những thông tin và file mà actor đã điền và chọn. |

##### **Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

##### **System Message**

|  |  |
| --- | --- |
| MS07 | “Gửi thành công”  Message thông báo khi hệ thống đã gửi thành công tới các email |
| MS08 | “Gửi thất bại”  Message thông báo khi hệ thống không gửi được file tới các email |

### UC19: Gửi báo cáo theo tháng qua email

##### **Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Gửi báo cáo theo tháng qua email | **Code** | UC16 |
| **Description** | Cho phép actor gửi báo cáo danh sách nhân viên và số ca đi làm theo tháng qua email | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Send]** trong trang Report\_Month |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống, điền thông tin tháng muốn xem | | |
| **Post condition** | Gửi báo cáo theo tháng qua email thành công | | |

##### **Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Gửi báo cáo theo tháng qua emai thành công** | | | |
| 1 | Ở cửa sổ Report\_Month, sau khi đã xuất file báo cáo actor bấm nút “Send” |  |  |
|  |  | 2 | Hệ thống hiển thị cửa số Send lên màn hình |
| 3 | Tại cửa sổ Send, actor chọn email trong danh sách muốn gửi (nếu không có thì nhập email vào thanh input), nhập các thông tin cần thiết |  |  |
| 4 | Actor bấm vào nút “chọn file” |  |  |
|  |  | 5 | Hệ thống mở cửa sổ chọn file |
| 6 | Tại của sổ chọn file, actor chọn file báo cáo đã xuất từ trước (có thể chọn nhiều file) rồi bấm “Open” |  |  |
|  |  | 7 | Hệ thống lấy được danh sách vị trí file muốn gửi, hiển thị lên cửa sổ Send |
| 8 | Tại của sổ Send, actor bấm “Send” |  |  |
|  |  | 9 | Hệ thống sẽ truy cập vào email của công ty sau đó gửi mail đã được nhập/chọn trước chứa những thông tin và file mà actor đã điền và chọn. |

##### **Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

##### **System Message**

|  |  |
| --- | --- |
| MS09 | “Gửi thành công”  Message thông báo khi hệ thống đã gửi thành công tới các email |
| MS10 | “Gửi thất bại”  Message thông báo khi hệ thống không gửi được file tới các email |

### UC20: Gửi bảng chấm công qua Email

##### **Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Xuất bảng chấm công | **Code** | UC17 |
| **Description** | Cho phép actor gửi file bảng chấm công theo phòng ban cho email được chọn | | |
| **Actor** | Accoutant | **Trigger** | Actor click button **[Send]** trong trang Accountant |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống, điền tên của phòng ban và tháng muốn lấy thông tin | | |
| **Post condition** | Gửi bảng chấm công qua email thành công | | |

##### **Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Gửi bảng chấm lương qua email thành công** | | | |
| 1 | Ở màn hình Accountant, sau khi đã xuất file báo cáo actor bấm nút “Send” |  |  |
|  |  | 2 | Hệ thống hiển thị cửa số Send lên màn hình |
| 3 | Tại cửa sổ Send, actor chọn email trong danh sách muốn gửi (nếu không có thì nhập email vào thanh input), nhập các thông tin cần thiết |  |  |
| 4 | Actor bấm vào nút “chọn file” |  |  |
|  |  | 5 | Hệ thống mở cửa sổ chọn file |
| 6 | Tại của sổ chọn file, actor chọn file báo cáo đã xuất từ trước (có thể chọn nhiều file) rồi bấm “Open” |  |  |
|  |  | 7 | Hệ thống lấy được danh sách vị trí file muốn gửi, hiển thị lên cửa sổ Send |
| 8 | Tại của sổ Send, actor bấm “Send” |  |  |
|  |  | 9 | Hệ thống sẽ truy cập vào email của công ty sau đó gửi mail đã được nhập/chọn trước chứa những thông tin và file mà actor đã điền và chọn. |

##### **Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

##### **System Message**

**N/A**

### UC21: Thêm nhân viên đi muộn

**Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Thêm nhân viên đi muộn | **Code** | UC01 |
| **Description** | Cho phép actor thêmthông tin nhân viên đi làm muộn trong phòng ban | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Add]** trên màn hình Late Manage |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống | | |
| **Post condition** | Thêm thành công thông tin nhân viên đi làm muộn | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Thêm thành công nhân viên đi làm muộn** | | | |
| 1 | Từ màn hình bất kỳ, actor lựa chọn ở left menu “Late Manage”/ “Add” |  |  |
|  |  | 2 | Hệ thống hiển thị màn hình AddLate và hiển thị form nhập thông tin cần thiết |
| 3 | Actor nhập các thông tin cần thiết |  |  |
|  |  | 4 | Validation các thông tin nhập vào. |
| 5 | Sửa lại những thông tin chưa đúng (nếu hệ thông check validation chưa đúng). |  |  |
|  |  | 6 | Lưu thông tin nhân viên đó đi làm muộn vào bảng LateList trên CSDL, thông báo thành công. |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

**N/A**

### UC22: Sửa thông tin nhân viên đi muộn

**Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Sửa thông tin nhân viên đi muộn | **Code** | UC02 |
| **Description** | Cho phép actor sửa thông nhân viên đi làm muộn trong phòng ban. | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Edit]** ở màn hình LateManage |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống | | |
| **Post condition** | Sửa thông tin nhân viên đi muộn thành công | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Sửa thông tin nhân viên đi làm muộn thành công** | | | |
| 1 | Từ màn hình LateManage, actor chọn thông tin nhân viên đi muộn trong phòng ban trên DataGrid rồi bấm nút “Edit” |  |  |
|  |  | 2 | Hệ thống hiển thị màn hình EditLate và hiển thị form nhập thông tin đã được điền sẵn thông tin của nhân viên đi muộn đó |
| 3 | Actor sửa lại các thông tin cần thiết |  |  |
|  |  | 4 | Validation các thông tin nhập vào. |
| 5 | Sửa lại những thông tin chưa đúng (nếu hệ thông check validation chưa đúng). |  |  |
|  |  | 6 | Sửa lại thông tin nhân viên đó đi làm muộn ở bảng LateList trên CSDL, thông báo thành công. |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

**N/A**

### UC23: Xóa thông tin nhân viên đi muộn

**Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Xóa nhân viên đi làm muộn | **Code** | UC03 |
| **Description** | Cho phép actor xóa thông tin nhân viên đi làm muộn trong phòng ban | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Delete]** ở màn hình LateManage |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống | | |
| **Post condition** | Xóa thông tin nhân viên đi làm muộn thành công | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Xóa thông tin nhân viên đi làm muộn thành công** | | | |
| 1 | Từ màn hình LateManage, actor chọn thông tin nhân viên đi muộn trong phòng ban trên DataGrid rồi bấm nút “Delete” |  |  |
|  |  | 2 | Hệ thống hiển thị cửa sổ thống báo xác nhận xóa |
| 3 | Actor click button “OK” |  |  |
|  |  | 4 | Hệ thống xóa dữ liệu thông tin nhân viên đó đi làm muộn trong bảng LateList trên CSDL, rồi thông báo thành công |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

**N/A**

### UC24: Tìm thông tin nhân viên đi muộn

**Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Tìm thông tin nhân viên đi muộn | **Code** | UC04 |
| **Description** | Cho phép actor tìm kiếm thông tin nhân viên hoặc các nhân viên trong phòng ban đi muộn bằng tên nhân viên , tháng , năm hoặc ngày cụ thể | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Search]** ở màn hình LateManage |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống | | |
| **Post condition** | Tìm kiếm thông tin thành công | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Tìm kiếm nhân viên thành công** | | | |
| 1 | Ở màn hình LateManage, actor nhập tên của nhân viên cần tìm vào thanh tìm kiếm, chọn tháng trong comboBox và nhập năm hoặc chọn ngày rồi click button **[Search]** ở trên màn hình |  |  |
|  |  | 2 | Hệ thống tìm thông tin nhân viên hoặc các nhân viên trong phòng ban đi muộn có thông tin trùng khớp với thông tin mà actor đã nhập trong bảng LateList trên CSDL rồi hiển thị lên bảng danh sách trên màn hình LateList |

**Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

**System Message**

**N/A**

### UC25: Gửi yêu cầu

##### **Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Gửi yêu cầu | **Code** |  |
| **Description** | Cho phép actor gửi yêu cầu xóa điểm danh của nhân viên | | |
| **Actor** | Staff | **Trigger** | Actor click button **[Send]** trong trang Request của Staff |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống, điền thông tin nhân viên bị điểm danh sai | | |
| **Post condition** | Gửi thành công yêu cầu cho admin | | |

##### **Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Gửi bảng chấm công qua email thành công** | | | |
| 1 | Người dùng truy cập vào hệ thống và chọn chức năng “Request". |  |  |
|  |  | 2 | Hệ thống hiển thị trang Request |
| 3 | Người dùng nhập thông tin được yêu cầu gồm tên công nhân, id, ngày điểm danh, ca điểm danh |  |  |
| 4 | Người dùng bấm Send |  |  |
|  |  | 5 | Hệ thống truy xuất dữ liệu của công nhân bị điểm danh nhầm, và ghi vào csdl để gửi cho admin |
|  |  |  | Hệ thống hiển thị thông báo gửi thành công nếu tìm được công nhân đó nếu không, hệ thống hiển thị thông báo yêu cầu nhập lại |

##### **Business Rules**

|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

##### **System Message**

|  |  |
| --- | --- |
| MS | “Request sent successful”  Message thông báo khi hệ thống truy xuất được đúng dữ liệu của nhân viên bị điểm danh sai |
| MS | “Request sent failure”  Message thông báo khi hệ thống không truy xuất được dữ liệu của nhân viên bị điểm danh sai |

### UC26: Xử lý yêu cầu

##### **Use Case Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Xử lý yêu cầu | **Code** |  |
| **Description** | Cho phép actor xử lý các yêu cầu từ staff | | |
| **Actor** | Admin | **Trigger** | Actor click button **[Accept] hoặc[Deny]** trong trang Request của Admin |
| **Pre-condition** | Actor đã đăng nhập vào hệ thống, chọn 1 yêu cầu | | |
| **Post condition** | Xử lý thành công yêu cầu | | |

##### **Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Gửi bảng chấm công qua email thành công** | | | |
| 1 | Người dùng truy cập vào hệ thống và chọn chức năng “Request". |  |  |
|  |  | 2 | Hệ thống hiển thị trang Request |
| 3 | Người dùng chọn 1 yêu cầu đang ở trạng thái”Idle” và chọn Accept hoặc Deny |  |  |
|  |  | 4 | Nếu người dùng chọn Accept, hệ thống sẽ xóa chấm công của nhân viên được gửi đến trong yêu cầu từ staff và chuyển trạng thái của yêu cầu thành Accepted, hiển thị thông báo yêu cầu được chấp thuật. Nếu chọn deny, hệ thống sẽ chuyển trạng thái của yêu cầu thành denied và hiển thị thông báo yêu cầu bị từ chối |
|  |  | 5 | Hệ thống hiển thị lại bảng yêu cầu |

##### **Business Rules**

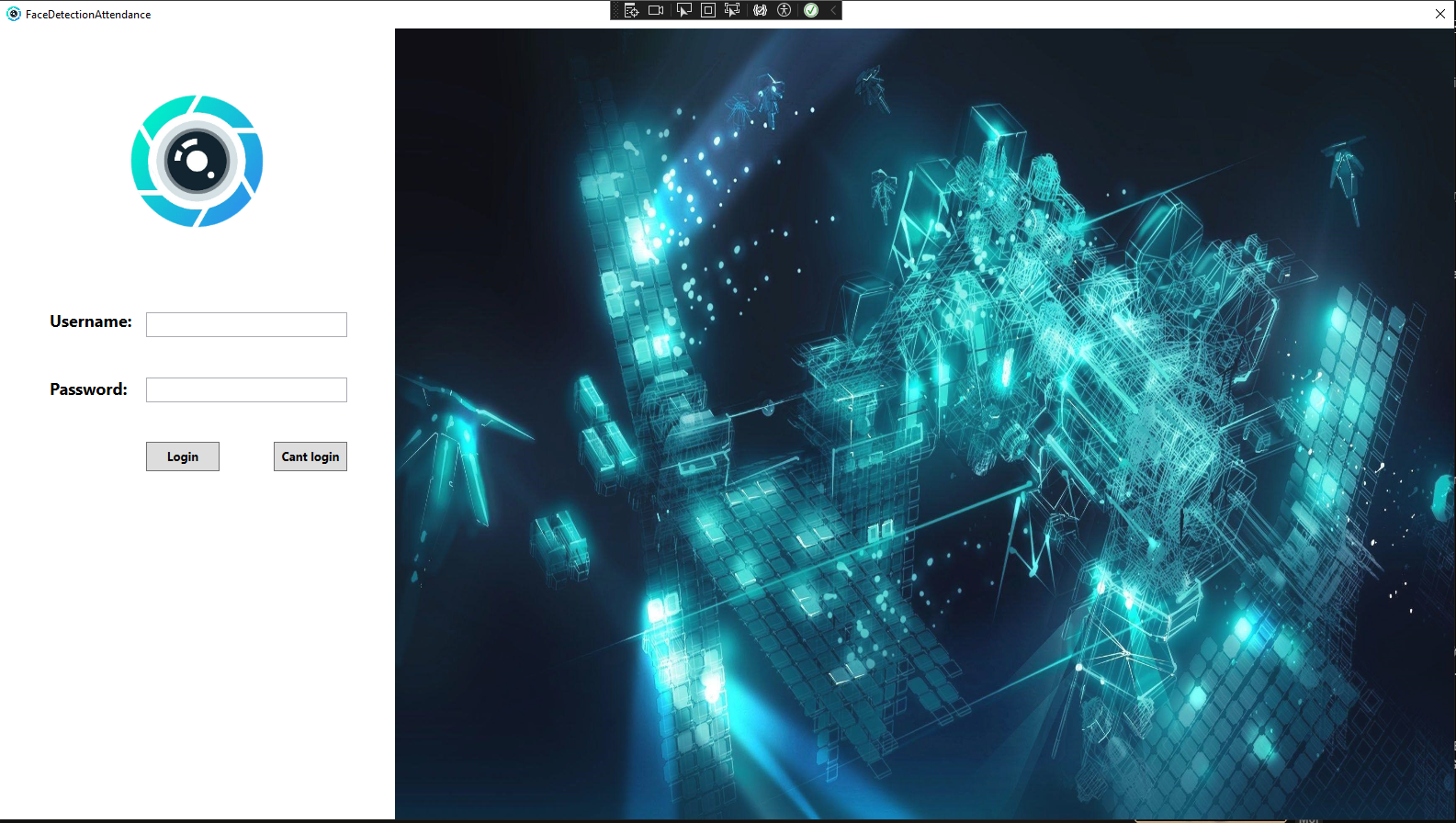
|  |  |  |
| --- | --- | --- |
| **Rule No.** | **Rule** | **Description** |
|  |  |  |

##### **System Message**

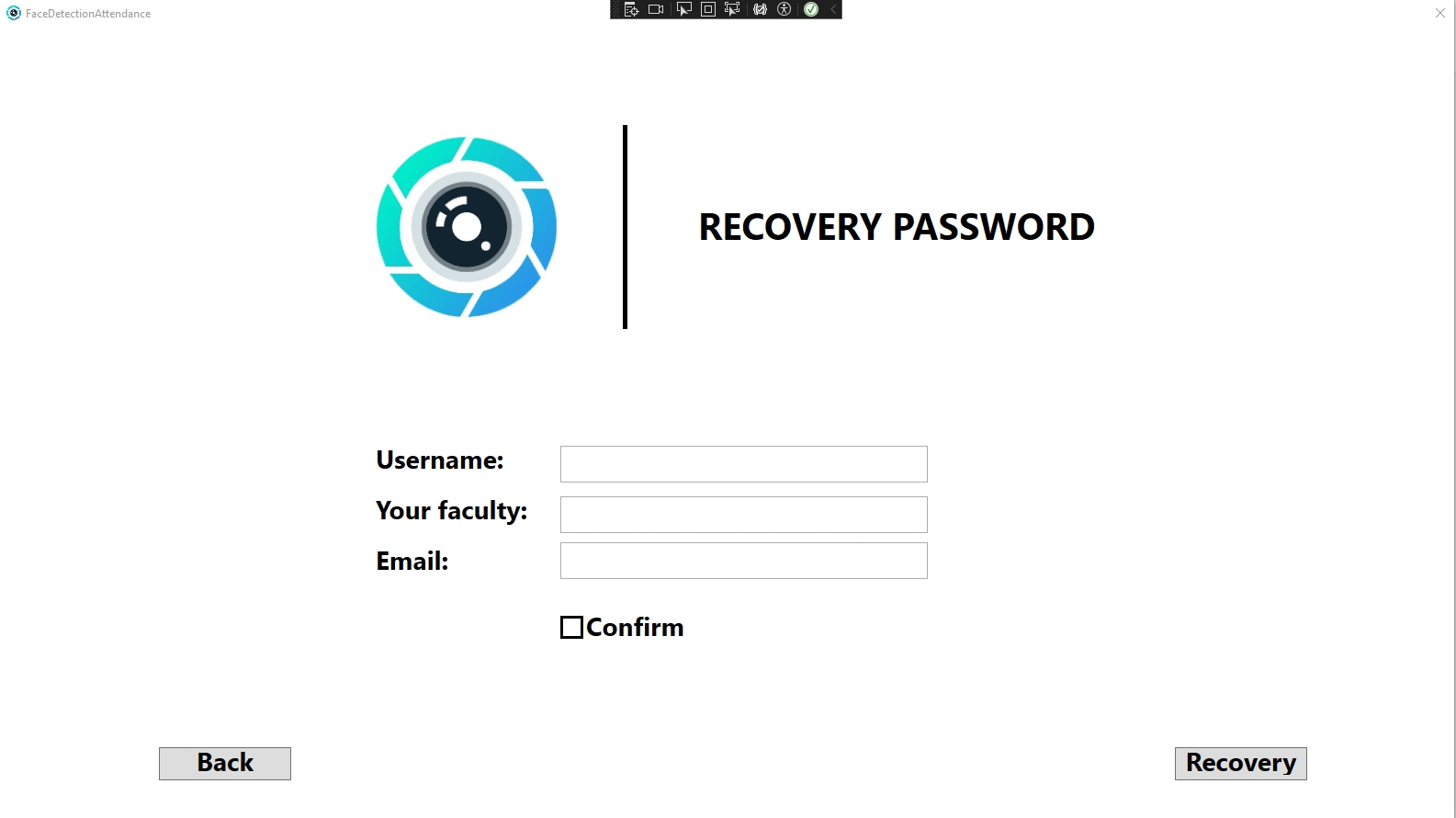
|  |  |
| --- | --- |
| MS | “Request accepted”  Message thông báo khi hệ thống xóa thành công nhân viên bị điểm danh sai |
| MS | “Request sent failure”  Message thông báo khi hệ thống chuyển thành công trạng thái của request thành Denied |

**CHƯƠNG 5: GIAO DIỆN NGƯỜI DÙNG**

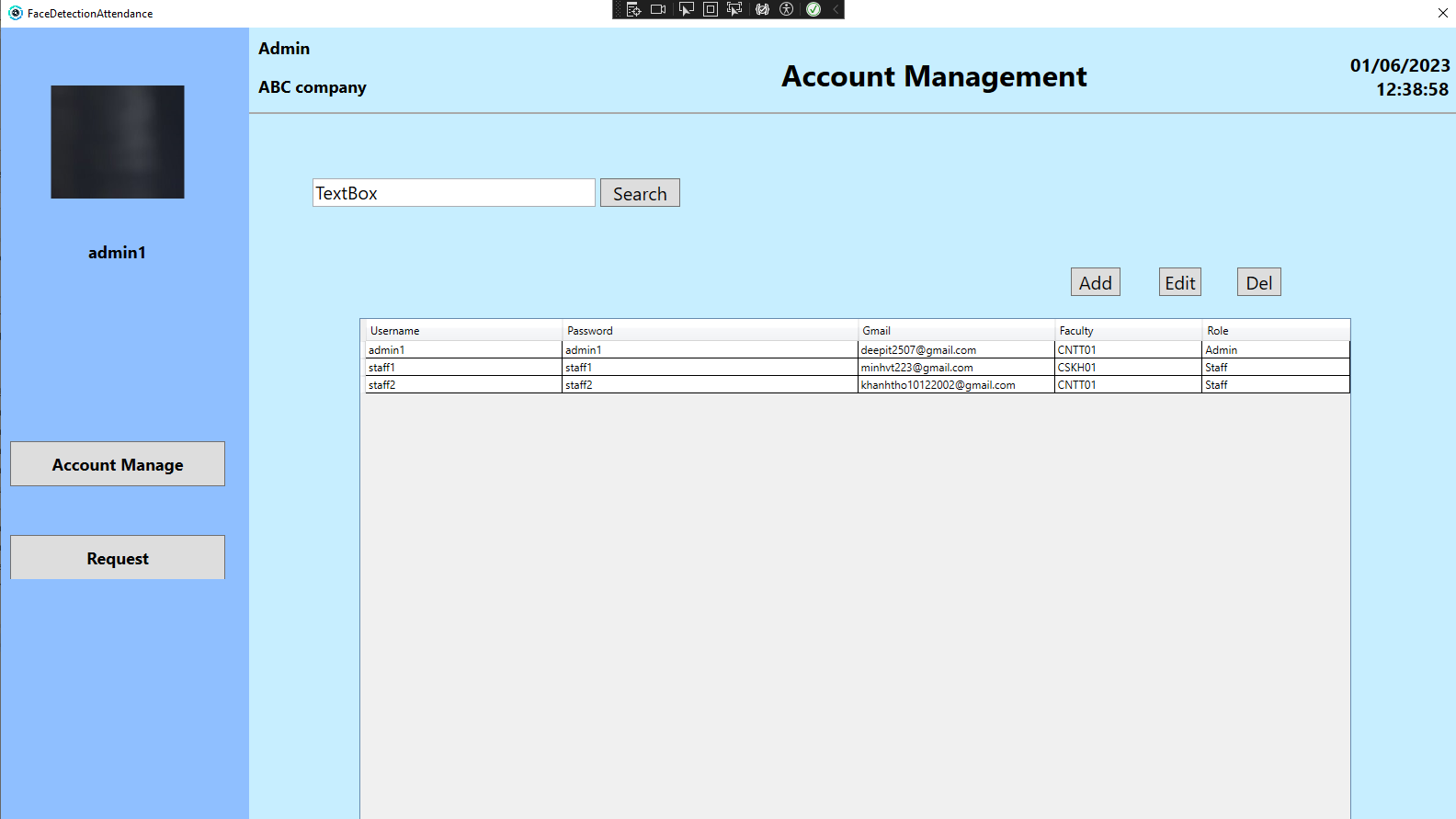
**5.1 Login**

****

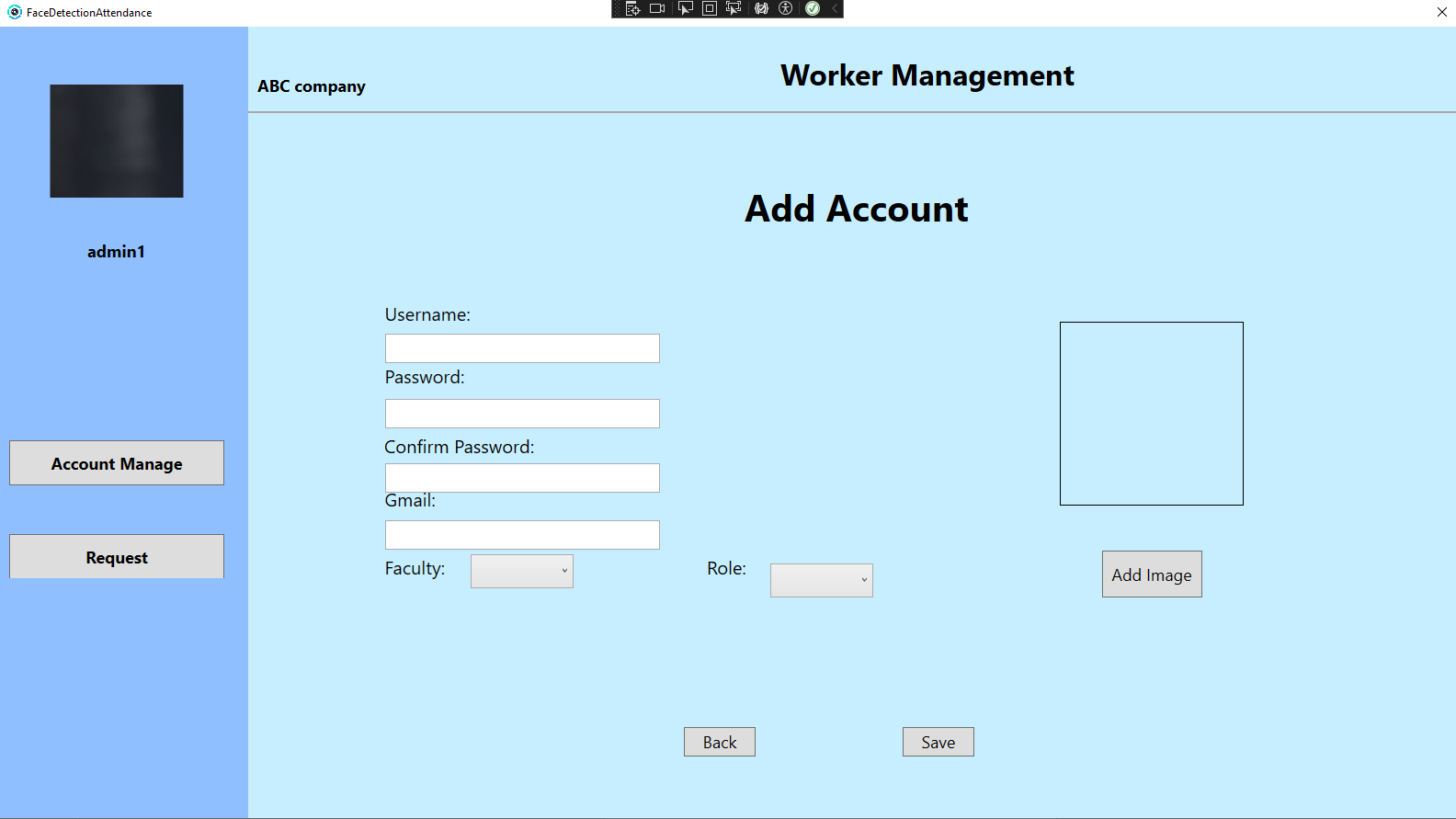
**5.2 Recovery**

****

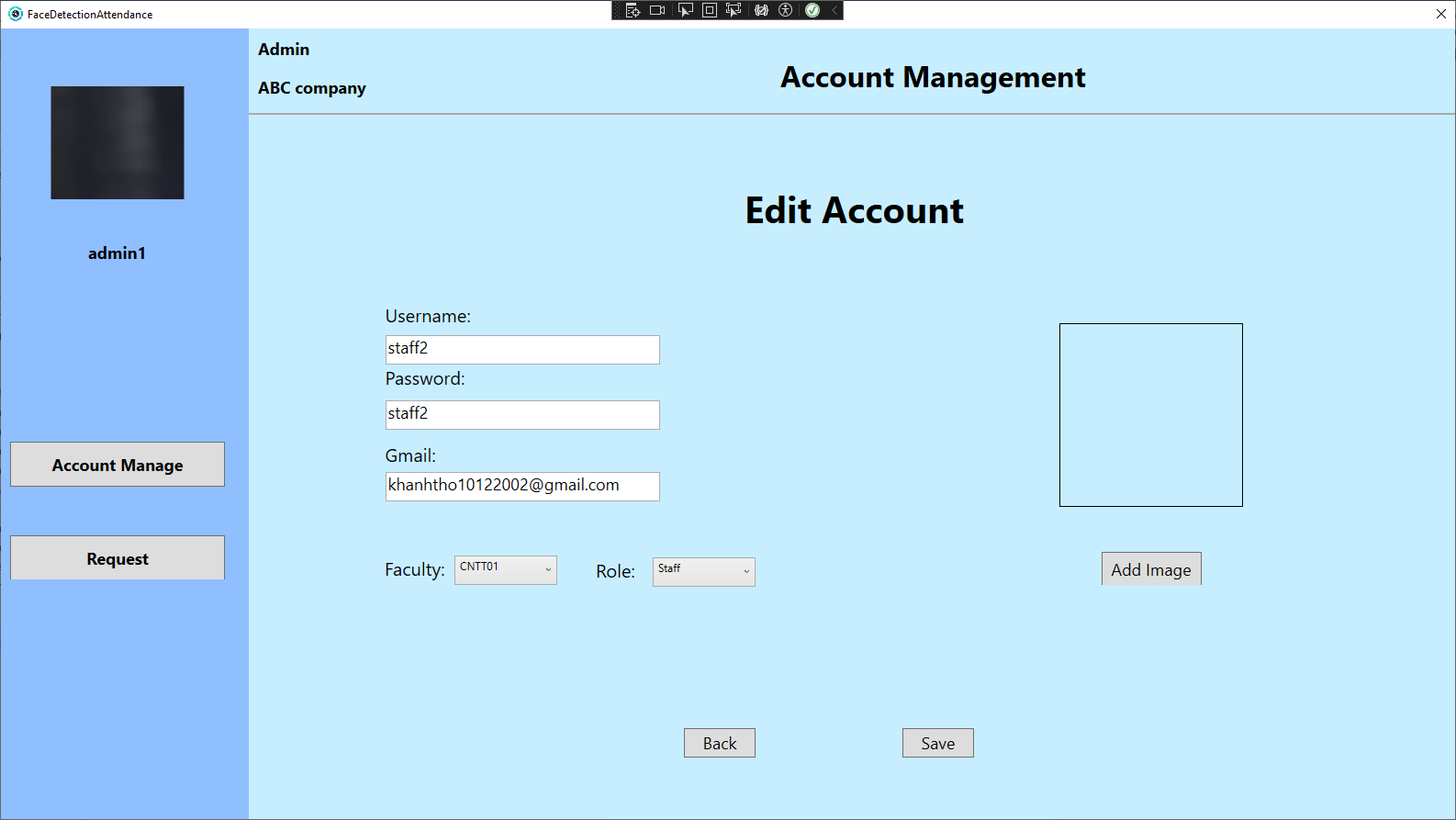
**5.3 Account manage**

****

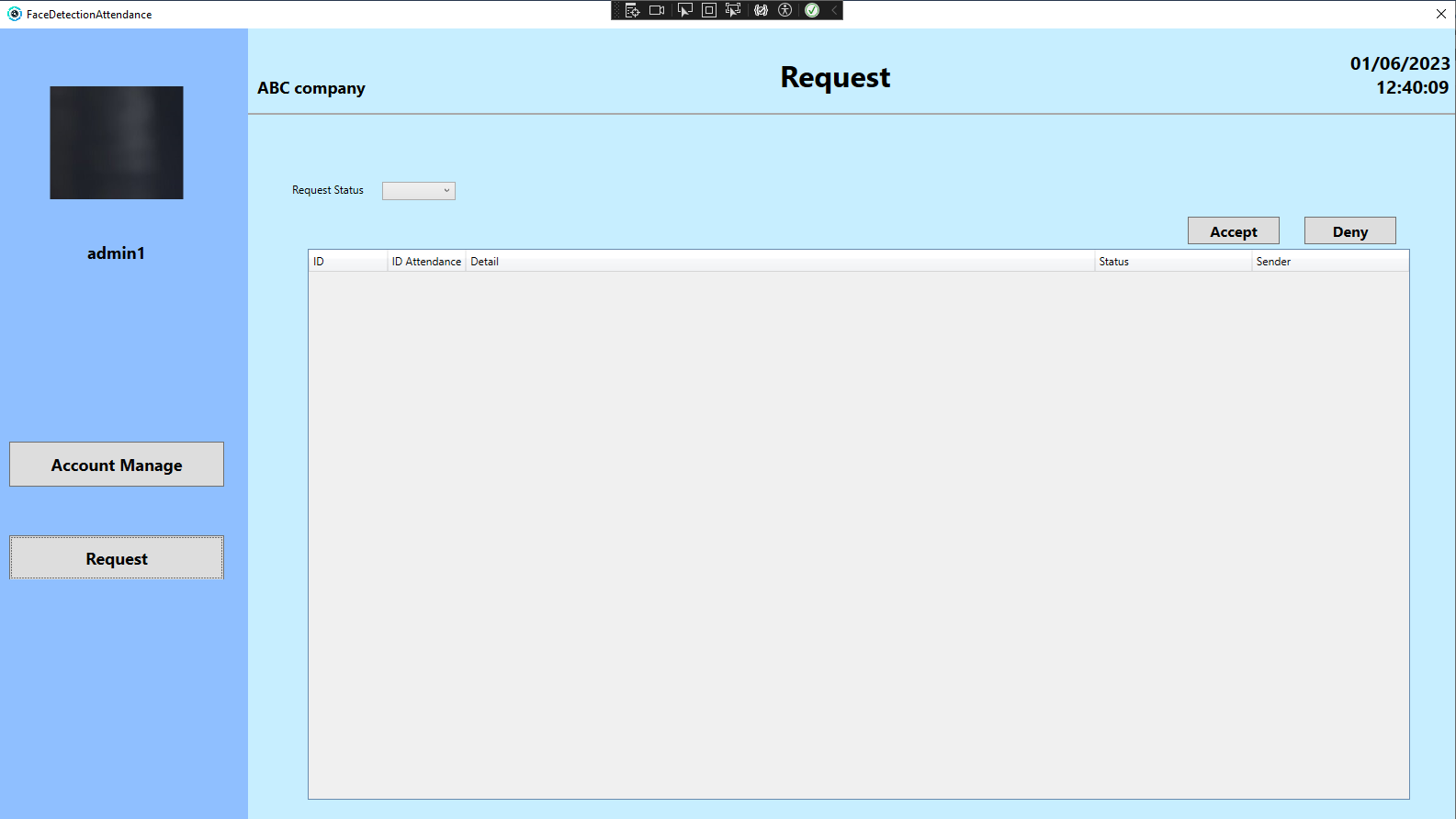
**5.4 Add account**

****

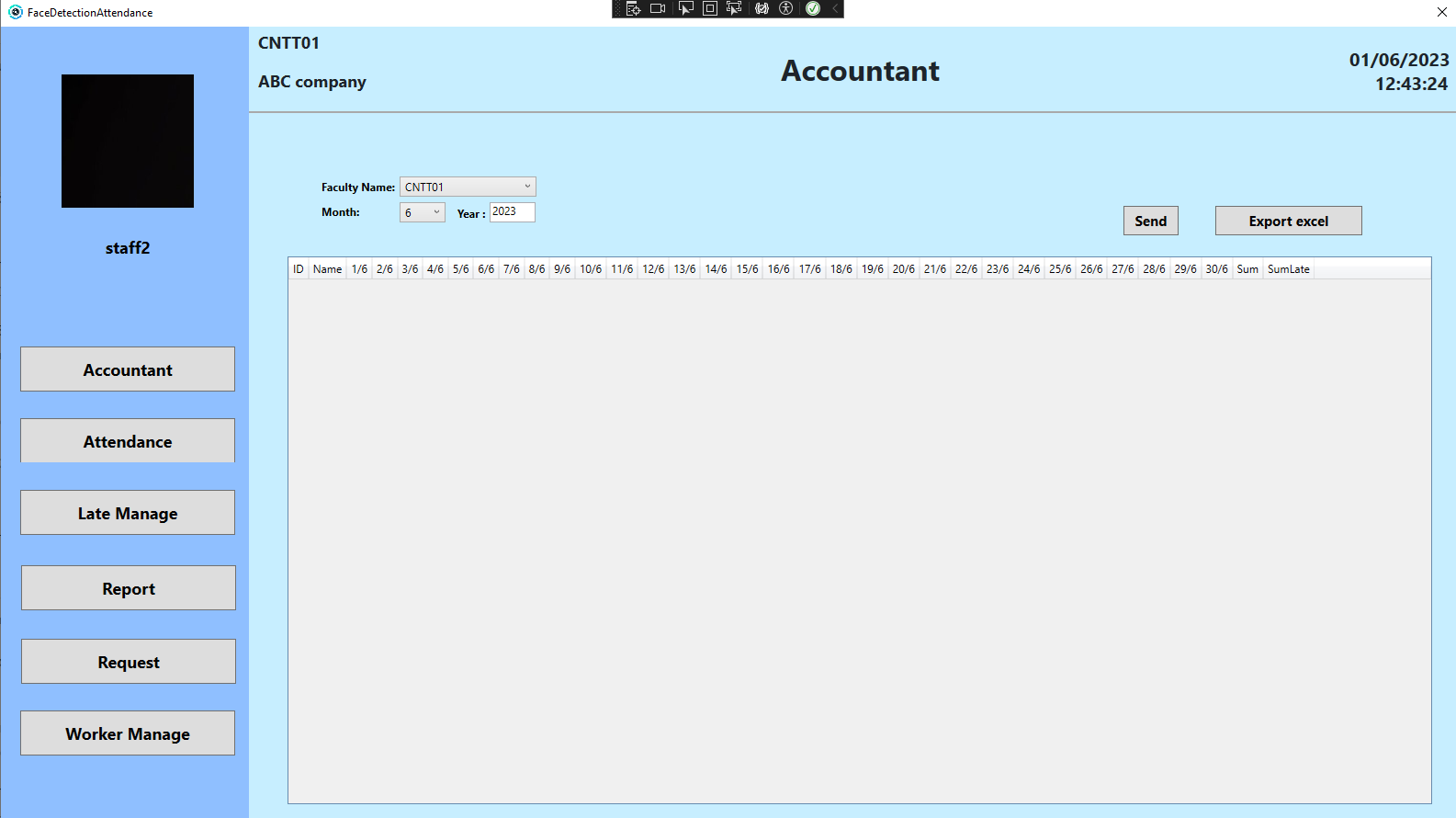
**5.5 Edit account**

****

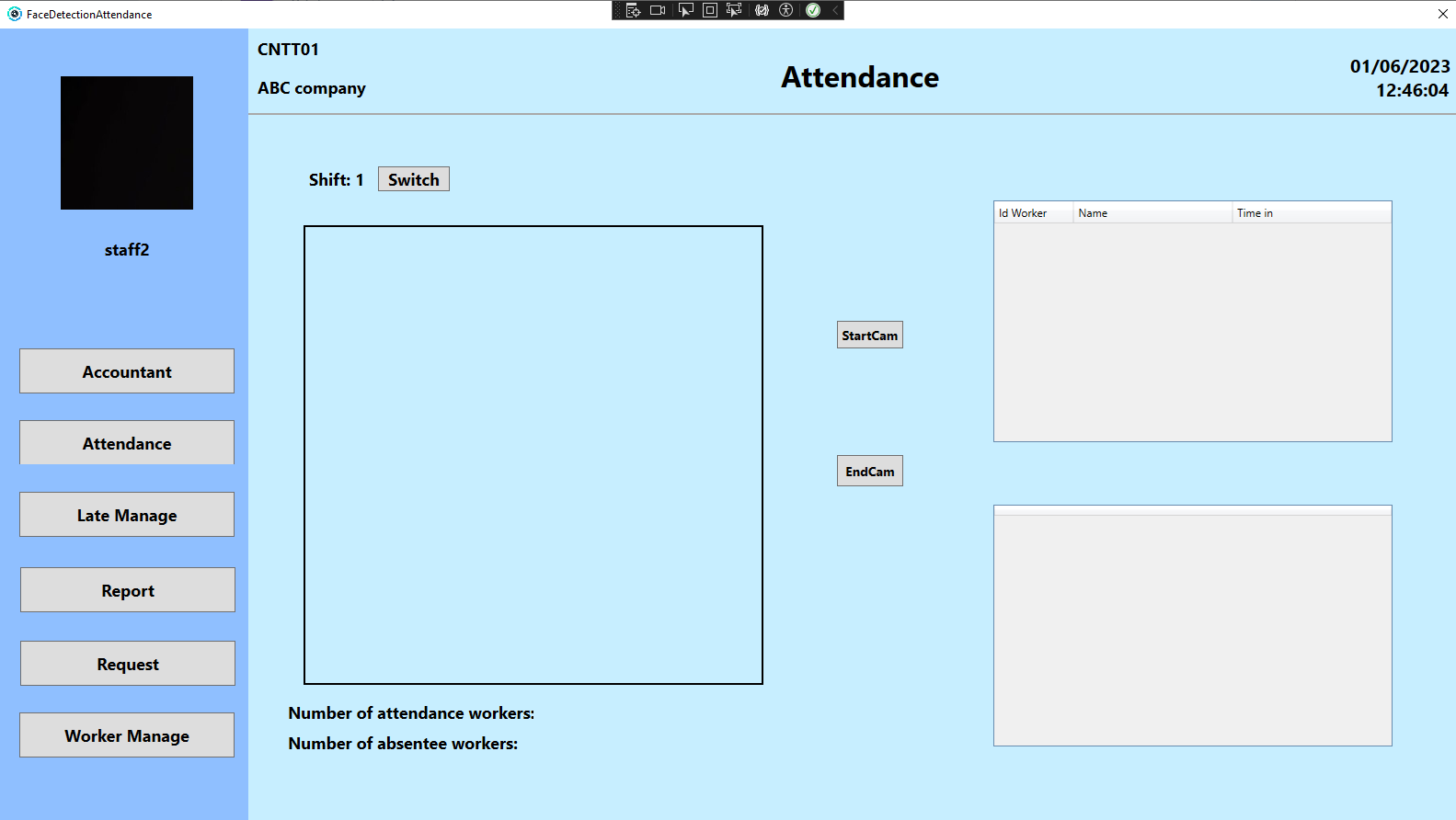
**5.5 Request(Admin)**

****

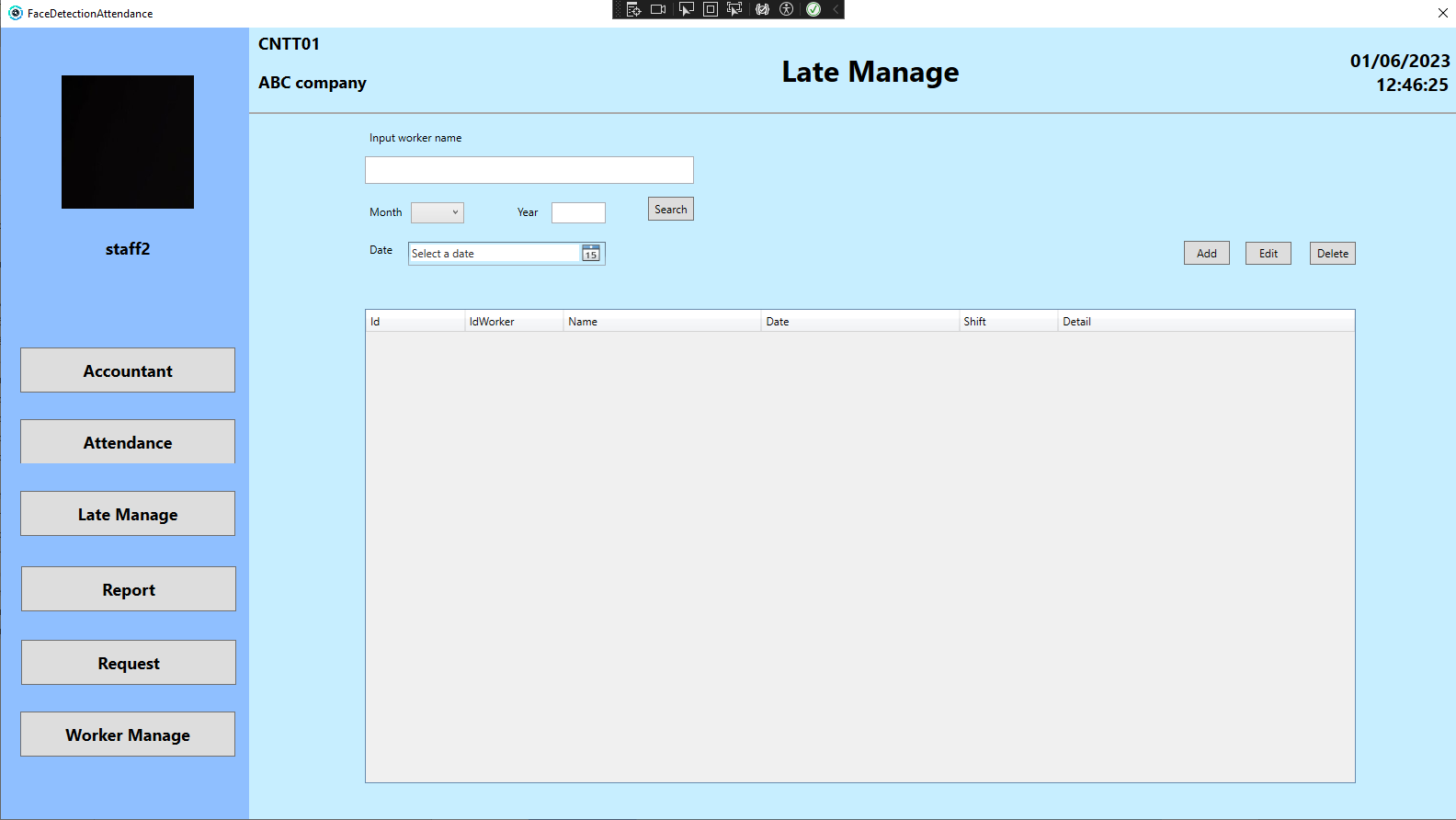
**5.6 Accountant**

****

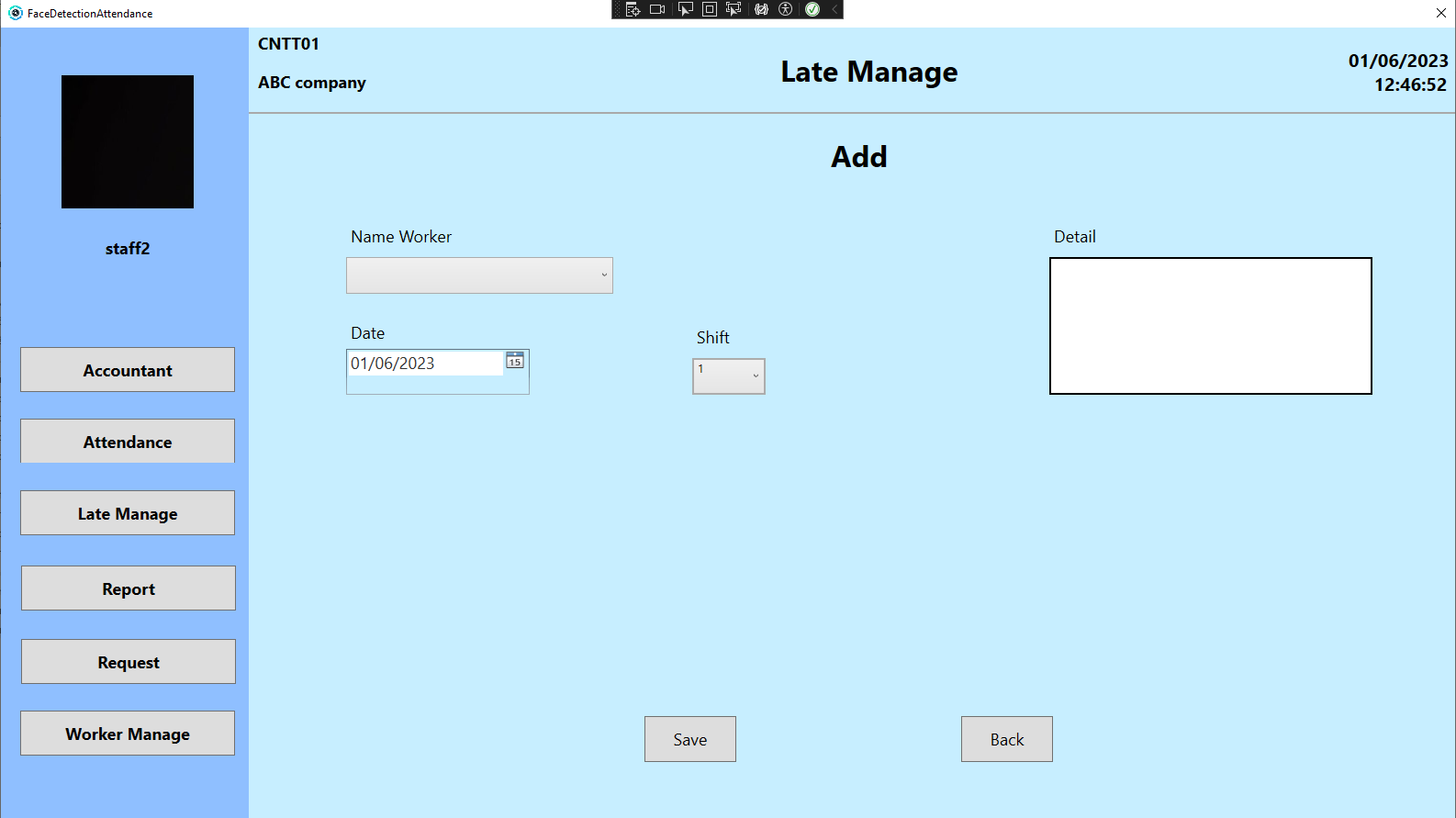
**5.7 Attendance**

****

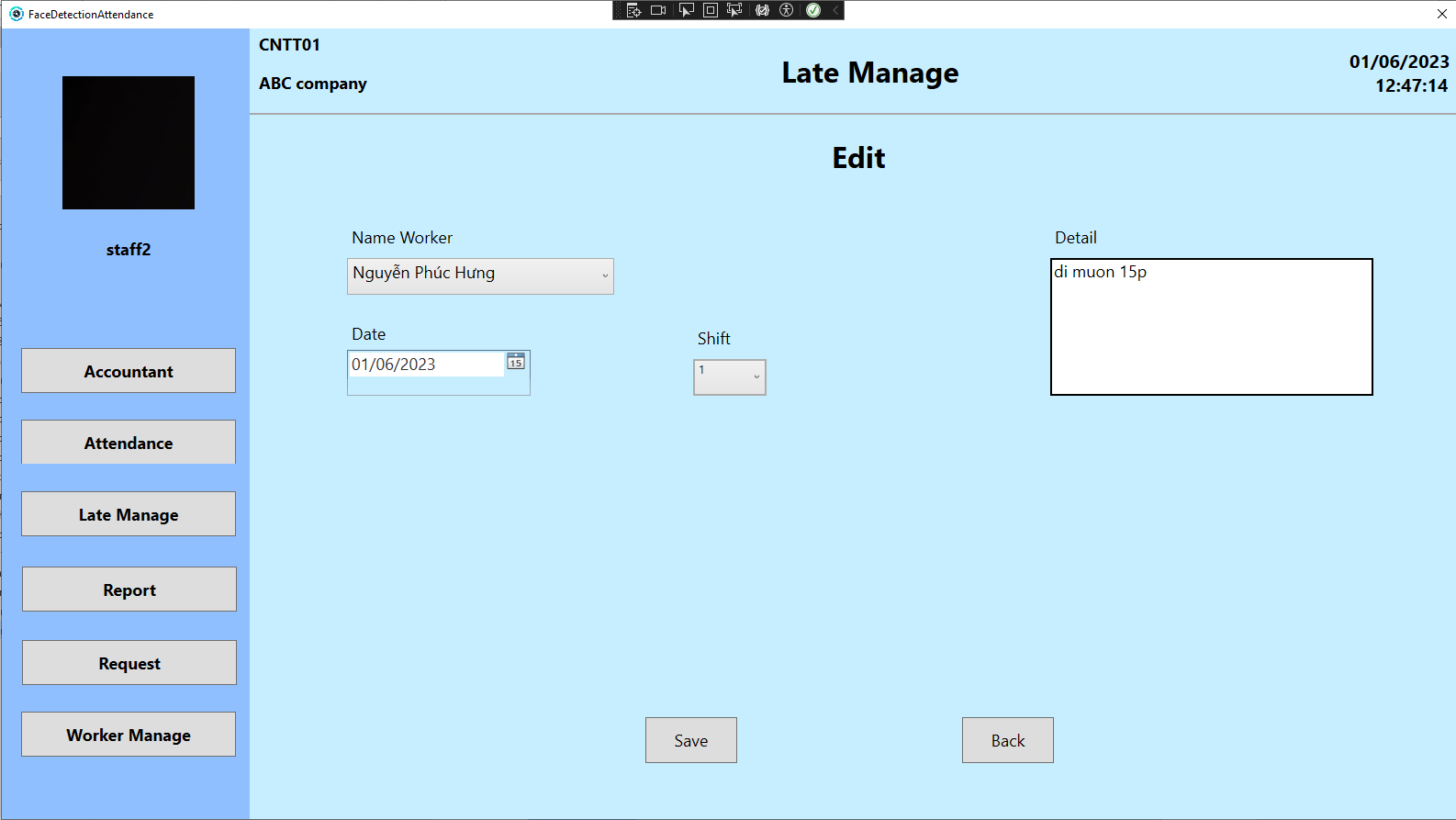
**5.8 Late manage**

****

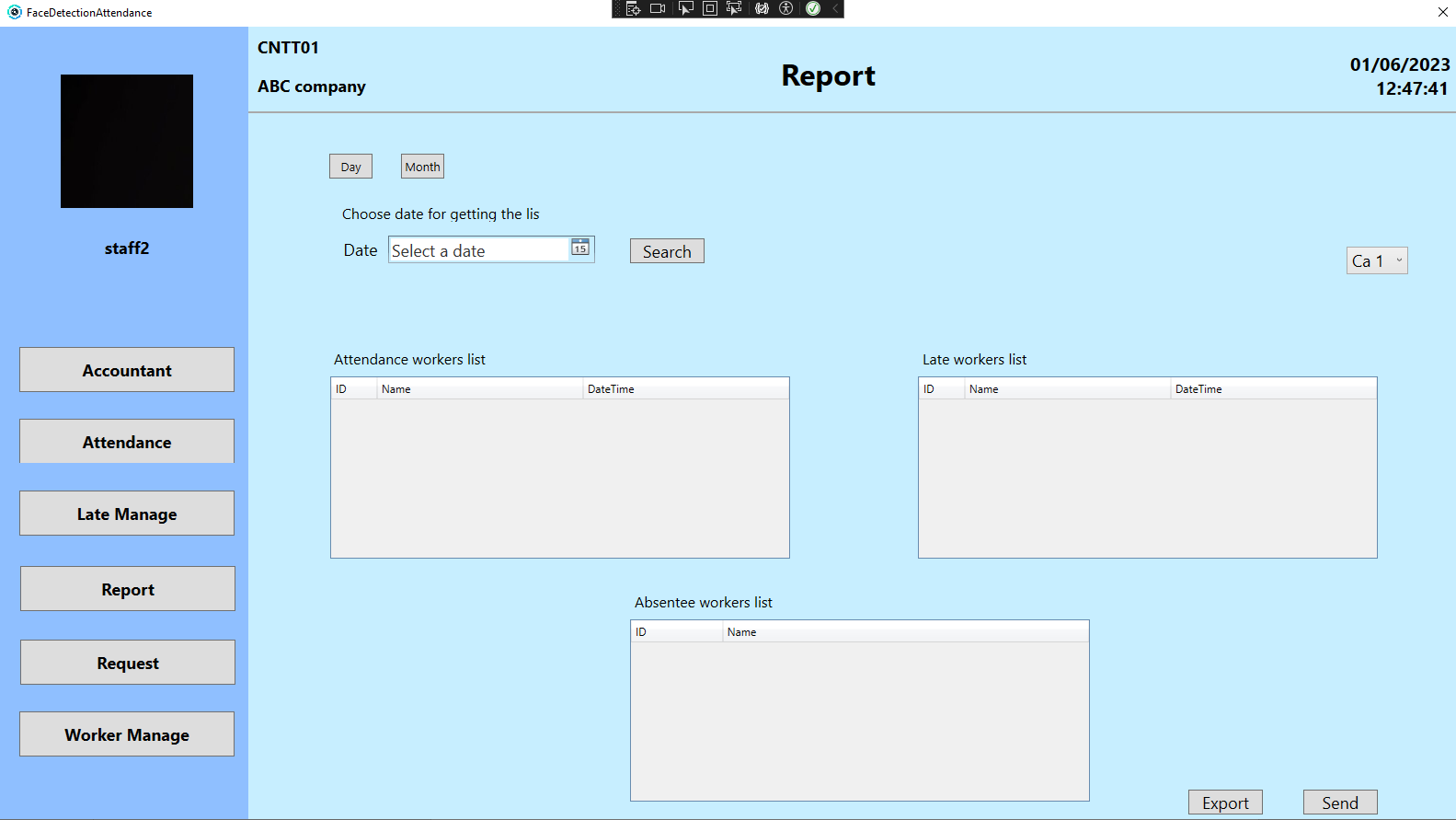
**5.9 Add late**

****

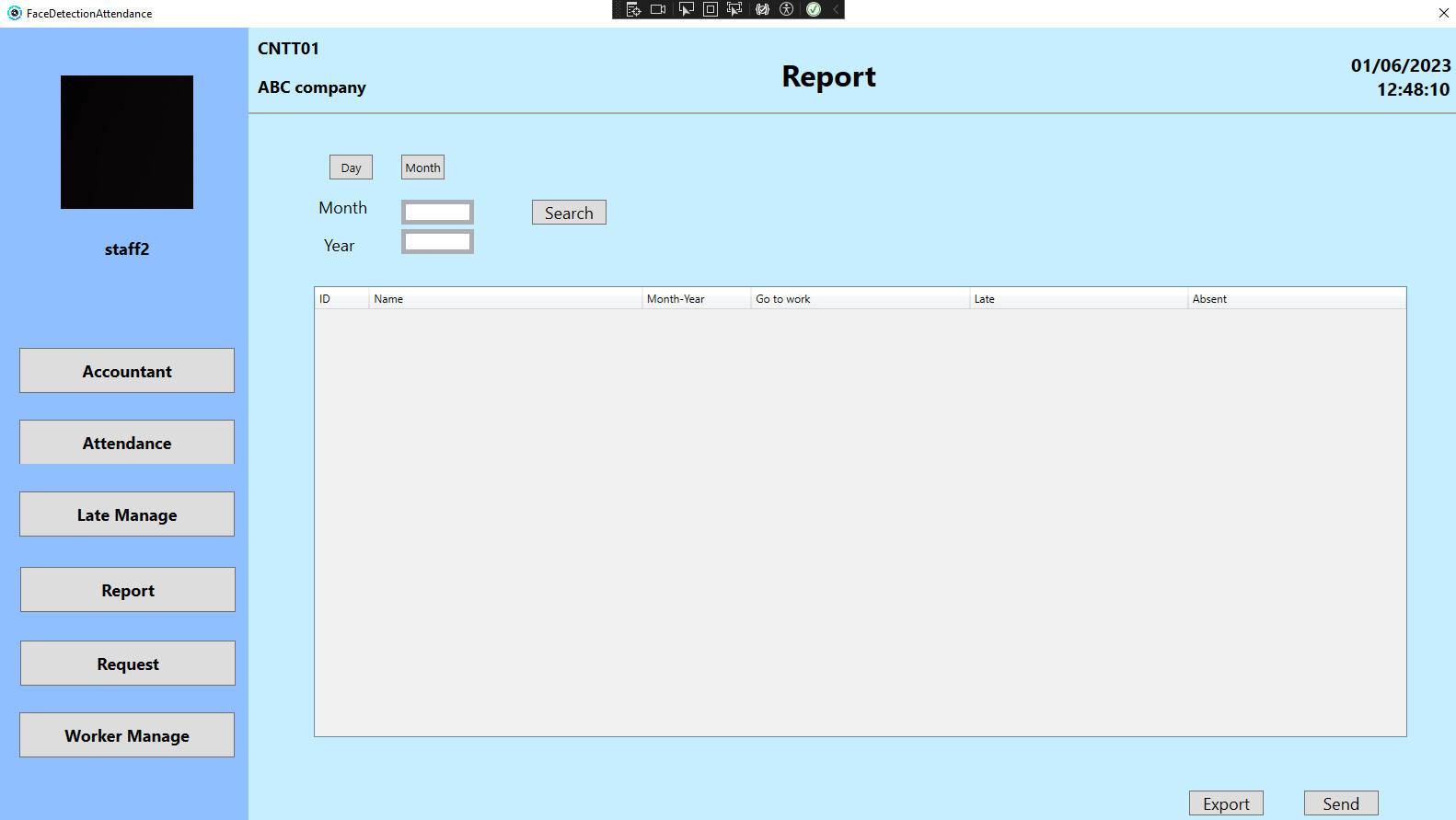
**5.10 Edit late**

****

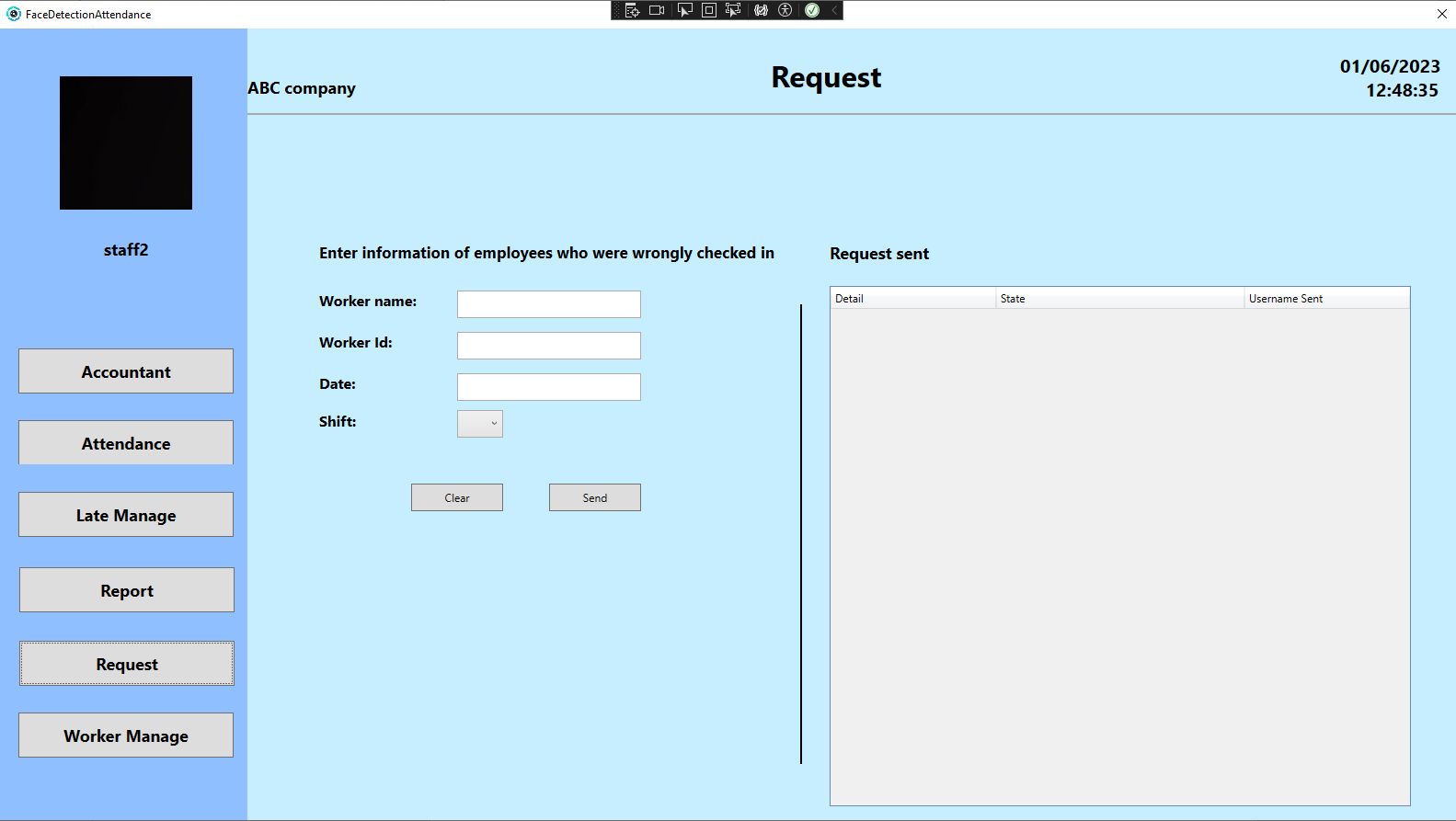
**5.11 Report day**

****

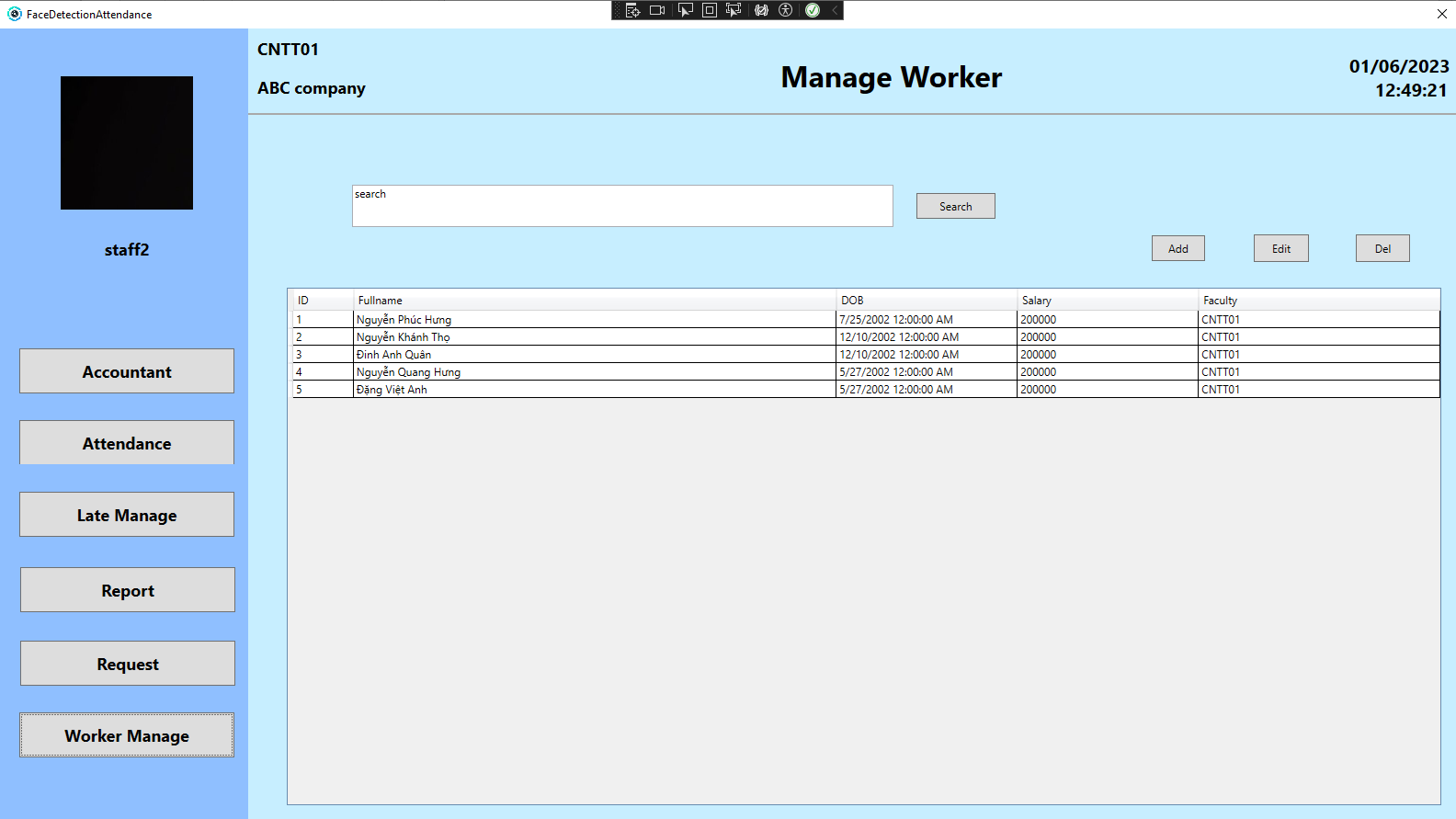
**5.12 Report month**

****

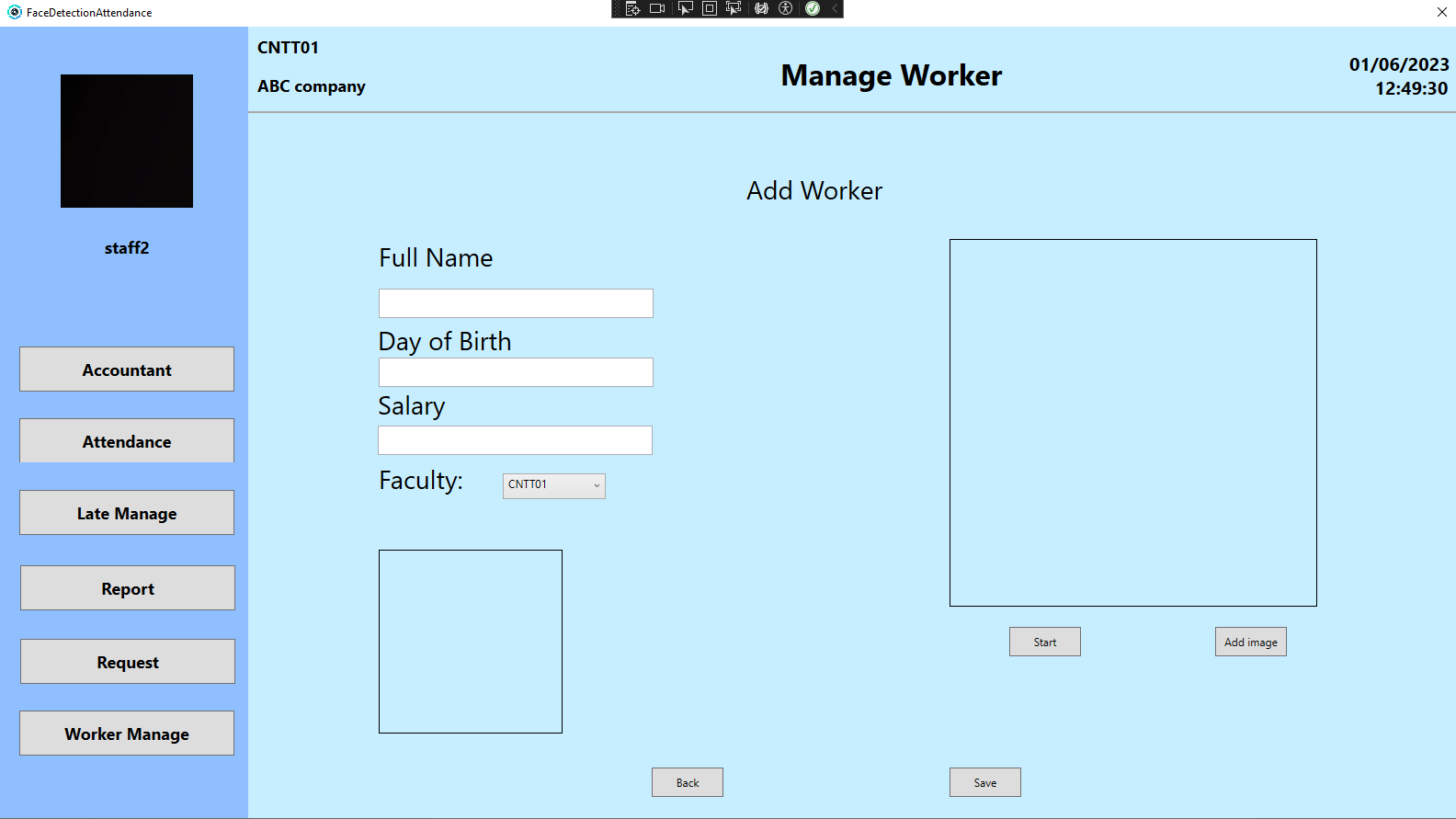
**5.13 Request(Staff)**

****

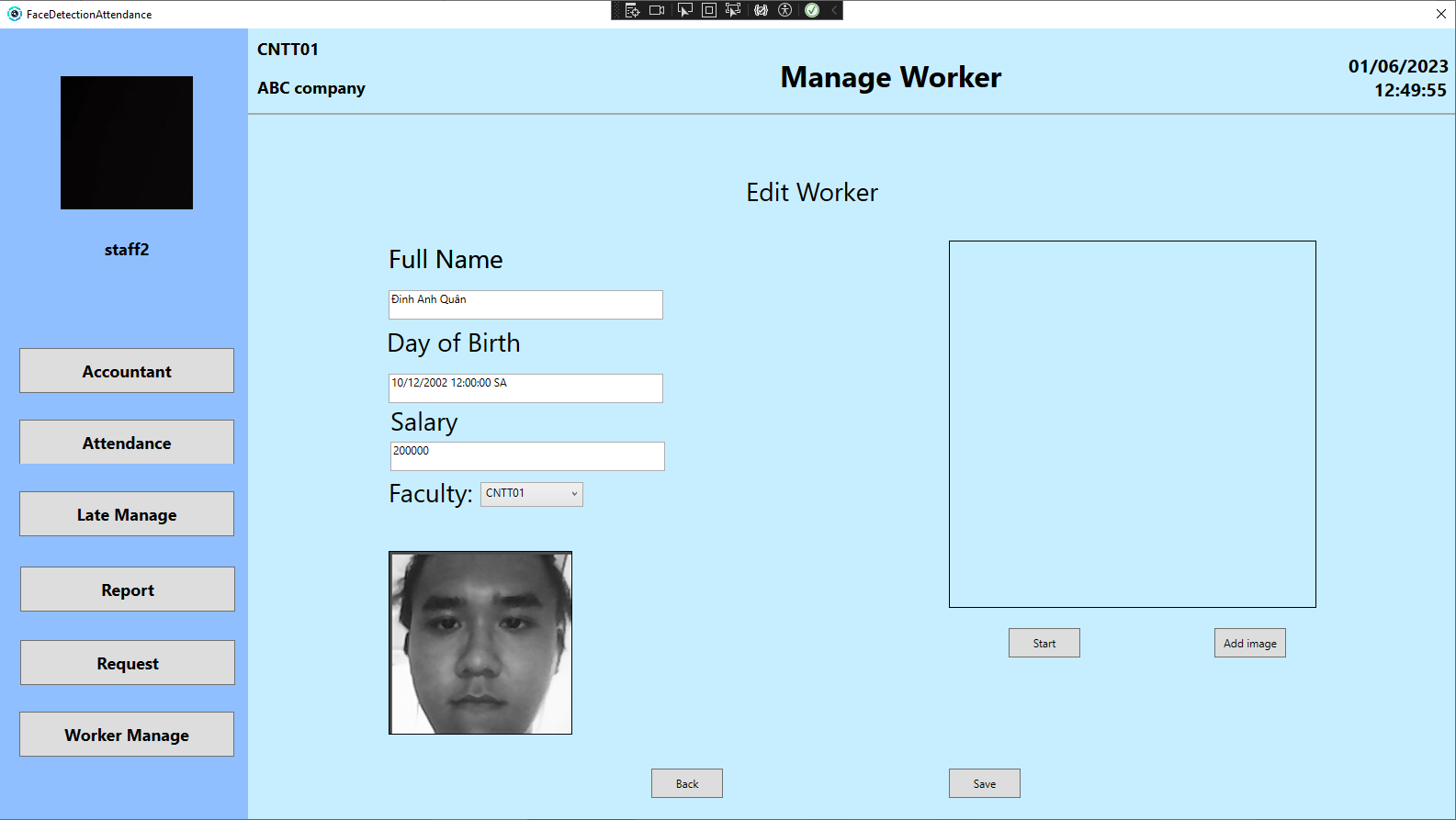
**5.14 Worker manage**

****

**5.15 Add worker**

****

**5.16 Edit worker**

****

**CHƯƠNG 6: THỰC HIỆN BÀI TOÁN**

**6.1 Đặng Việt Anh**

**Màn hình Account manage:**

+ Giao diện: Chỉ hiển thị danh sách những tài khoản có trong cơ sở dữ liệu của hệ thống.

+ Mã lệnh:

private void Loaddata()

{

string querry = "Select username,passwords,fid, gmail,roles From Account";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

dtc.GetConnection().Open();

try{

command = new SqlCommand(querry, dtc.GetConnection());

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

int temp = reader.GetInt32(4);

string role;

if (temp == 1)

{

role = "Admin";

}

else

{

role = "Staff";

}

string Username = reader.GetString(0);

string Passwords = reader.GetString(1);

string Fid = reader.GetString(2);

string Gmail = reader.GetString(3);

string Roles = role;

Accountdtg.Items.Add(new { username = Username, password = Passwords, fid = Fid, gmail = Gmail, roles = Roles });

}

}catch(Exception ex){

MessageBox.Show(ex.Message);

}

}

Hàm trên được sử dụng để lấy dữ liệu từ trong cơ sở dữ liệu rồi ánh xạ lên datagrid. Câu lệnh SqlCommand được dung để kết nối ứng dụng đến cơ sở dữ liệu và thực hiện câu lệnh query, sử dụng thêm try catch để bắt lỗi những trường hợp sai câu lệnh sql, sai bảng hoặc kết nối từ ứng dụng đến cơ sở dữ liệu có sự cố và hiển thị lỗi trên màn hình.

**Màn hình Add account:**

+ Giao diện: Nhận vào tài khoản, mật khẩu, chức vụ, gmail, ảnh, phòng ban mà người dùng nhập vào. Hiển thị lên thông báo thêm thành công và ảnh đại diện cho account khi người dùng thêm vào.

+ Mã lệnh:

private void Savebtn\_Click(object sender, RoutedEventArgs e)

{

if (string.IsNullOrEmpty(Usernametxb.Text) ||

Passwordtxb.Text.Length == 0 ||

Confirmtxb.Text.Length == 0 ||

Gmailtxb.Text == "" ||

facultycbb.SelectedItem == null ||

Rolecbb.SelectedItem == null ||

Imagebd.Background == null)

{

MessageBox.Show("Please enter enough information.");

return;

}

if (Passwordtxb.Text != Confirmtxb.Text)

{

MessageBox.Show("Please re-enter the password.");

return;

}

Đoạn mã trên dung để kiểm tra xem người dung đã nhập đủ thông tin hay chưa và kiểm tra xem phần Confirm Password đã giống với mật khẩu người dung nhập vào hay chưa.

string username = Usernametxb.Text;

string password = new System.Net.NetworkCredential(string.Empty, Passwordtxb.Text).Password;

string gmail = Gmailtxb.Text;

string faculty = facultycbb.SelectedItem.ToString();

string role = Rolecbb.SelectedItem.ToString();

BitmapImage image = ((ImageBrush)Imagebd.Background).ImageSource as BitmapImage;

string avt = resourceFolderPath + @$"\Avatar\{username}.png";

File.Copy("C:\\FDA temp\\temp.png", avt);

string querry = "Insert into Account Values(@username,@passwords,@roles, @gmail,@images,@fid)";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

dtc.GetConnection().Open();

int temp;

if (role == "Admin")

{

temp=1;

}

else

{

temp=0;

}

try

{

SQLcommand = new SqlCommand(querry, dtc.GetConnection());

SQLcommand.Parameters.Add("@username", SqlDbType.NVarChar).Value = username;

SQLcommand.Parameters.Add("@passwords", SqlDbType.NVarChar).Value = password;

SQLcommand.Parameters.Add("@fid", SqlDbType.NVarChar).Value = faculty;

SQLcommand.Parameters.Add("@gmail", SqlDbType.NVarChar).Value = gmail;

SQLcommand.Parameters.Add("@roles", SqlDbType.NVarChar).Value = temp;

SQLcommand.Parameters.Add("@images", SqlDbType.NVarChar).Value = username;

SQLcommand.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

Đoạn mã trên dùng để lưu thông tin tài khoản vào cơ sở dữ liệu và lưu ảnh địa diện vào thư mục Resource của ứng dụng. Sử dụng SqlCommand để thực hiện câu lệnh Insert trong database của hệ thống. Câu lệnh: File.Copy được dùng để chuyển ảnh người dùng đã thêm vào để sử dụng làm ảnh đại diện vào đúng folder của hệ thống.

**Màn hình Edit account:**

+ Giao diện: Nhận vào dữ liệu của tài khoản mà người dùng muốn sửa và hiển thị chúng lên màn hình, sau khi người dùng bấm save thì hiển thị thông báo chỉnh sửa thành công.

+ Mã lệnh:

string avt = resourceFolderPath + @$"\Avatar\{username}.png";

string querry = "update Account set username = @username, passwords=@passwords, gmail=@gmail, images=@images, fid=@fid where username = @U ";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

dtc.GetConnection().Open();

int temp;

File.Delete(avt);

File.Copy("C:\\FDA temp\\avatar.png",avt );

if (role == "Admin")

{

temp = 1;

}

else

{

temp = 0;

}

try

{

command = new SqlCommand(querry, dtc.GetConnection());

command.Parameters.Add("@username", SqlDbType.NVarChar).Value = username;

command.Parameters.Add("@passwords", SqlDbType.NVarChar).Value = password;

command.Parameters.Add("@fid", SqlDbType.NVarChar).Value = faculty;

command.Parameters.Add("@gmail", SqlDbType.NVarChar).Value = gmail;

command.Parameters.Add("@roles", SqlDbType.NVarChar).Value = temp;

command.Parameters.Add("@images", SqlDbType.NVarChar).Value = username;

command.Parameters.Add("@U", SqlDbType.NVarChar).Value = Currentusername;

command.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

Đoạn mã trên dùng để lưu thông tin tài khoản vào cơ sở dữ liệu và lưu ảnh địa diện vào thư mục Resource của ứng dụng. Sử dụng SqlCommand để thực hiện câu lệnh Update trong database của hệ thống. Câu lệnh: File.Copy() được dùng để chuyển ảnh người dùng đã thêm vào để thay thế ảnh cũ để làm ảnh đại diện vào đúng folder của hệ thống và xóa ảnh cũ với câu lệnh File.Delete().

**Màn hình Request(Admin):**

+Giao diện: Hiển thị thông tin về các yêu cầu được gửi tới và hiển thị thông báo khi xử lý xong 1 yêu cầu.

+Mã lệnh:

string query = "select \* from Request";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

{

dtc.GetConnection().Open();

}

try

{

cmd = new SqlCommand(query, dtc.GetConnection());

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

Request request = new Request();

request.id = reader.GetInt32(0);

request.id\_attendance = reader.GetInt32(1);

request.detail = reader.GetString(2);

request.states = reader.GetString(3);

request.usernamesent = reader.GetString(4);

Requestdtg.Items.Add(new {id = request.id ,id\_attendance = request.id\_attendance, detail = request.detail, status = request.states, usernamesent = request.usernamesent});

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

Mã lênh trên được dùng để lấy những yêu cầu có trong cơ sở dữ liệu của hệ thống và hiển thị lên datagrid. Câu lệnh SqlCommand được dung để kết nối ứng dụng đến cơ sở dữ liệu và thực hiện câu lệnh query, sử dụng thêm try catch để bắt lỗi những trường hợp sai câu lệnh sql, sai bảng hoặc kết nối từ ứng dụng đến cơ sở dữ liệu có sự cố và hiển thị lỗi trên màn hình.

var temp = Requestdtg.SelectedItem;

int id = 0;

if (temp != null)

{

dynamic selected = temp;

id = selected.id;

}

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

dtc.GetConnection().Open();

string query = "Update Request set states = 'Accepted' where id = @id";

try

{

cmd = new SqlCommand(query, dtc.GetConnection());

cmd.Parameters.AddWithValue("@id", id);

cmd.ExecuteNonQuery();

Requestdtg.Items.Refresh();

string query2 = "Delete from Attendance where id = @id";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

dtc.GetConnection().Open();

try

{

cmd = new SqlCommand(query2, dtc.GetConnection());

cmd.Parameters.AddWithValue("@id", id);

cmd.ExecuteNonQuery();

dtc.GetConnection().Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

Requestdtg.Items.Refresh();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

Đoạn mã trên được dùng để thực hiện yêu cầu của quản lý là xóa phần điểm danh của 1 nhân viên và chuyển trạng thái của yêu cầu thành Accepted/Deny. Khi admin chọn 1 yêu cầu và bấm Accept/Deny thì hệ thống sẽ kết nối đến cơ sở dữ liệu thông qua SqlCommand và và thực hiện các yêu cầu hoặc không. Bắt lỗi nếu kết nối đến cơ sở dữ liệu bị lỗi hoặc thông tin về nhân viên muốn xóa bị sai và hệ thống không tìm thấy đúng dữ liệu để xóa.

**6.2 Nguyễn Phúc Hưng**

**Màn hình Login:**

+ Giao diện: Nhận vào các thông tin người dùng điền vào các ô. Hiển thị các ô để người dùng nhập tài khoản và mật khẩu và ảnh chờ và các nút để người dùng tương tác với hệ thống.

+ Mã lệnh:

string querry = "Select Count(1) from Account where username =@username and passwords=@password";

try

{

if (Dataconnecttion.GetConnection().State == System.Data.ConnectionState.Closed)

Dataconnecttion.GetConnection().Open();

SqlCommand cmd = new SqlCommand(querry, Dataconnecttion.GetConnection());

cmd.Parameters.AddWithValue("@username", acc.username);

cmd.Parameters.AddWithValue("@password", acc.password);

int check = Convert.ToInt32(cmd.ExecuteScalar());

if (check == 1)

{

string querry2 = "Select Roles from Account where username=@username";

try

{

SqlCommand cmd2 = new SqlCommand(querry2, Dataconnecttion.GetConnection());

cmd2.Parameters.AddWithValue("@username", acc.username);

int roles = Convert.ToInt32(cmd2.ExecuteScalar());

Dataconnecttion.GetConnection().Close();

MessageBoxChange ms = new MessageBoxChange(roles, acc.username);

ms.WindowStartupLocation = WindowStartupLocation.CenterScreen;

ms.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

else

{

MessageBox.Show("Wrong username or password");

MessageBox.Show(acc.password);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

Đoạn mã trên được dùng để kiểm tra thông tin đăng nhập và nhận viết chức vụ của mỗi tài khoản để đưa đến đúng cửa sổ làm việc của từng chức vụ. Khi thực hiện lệnh Sql, ExcuteScalar() sẽ trả về 1 giá trị nếu giá trị đó bằng 1 thì tài khoản và mật khẩu đúng sau đó hệ thống sẽ thực hiện 1 câu lệnh Sql nữa để lấy ra role của tài khoản đó và đưa đến đúng cửa sổ làm việc. Bắt lỗi sai mật khẩu hoặc tài khoản, lỗi liên quan đến cơ sở dữ liệu.

**Màn hình Recover:**

+ Giao diện: Nhận vào email, tài khoản, mã phòng ban. Hiển thị logo app và các nút để người dùng tương tác và thông báo khi gửi thành công.

+ Mã lệnh:

string querry = "Select count(1) from Account inner join Faculty on Account.fid = Faculty.id\_faculty where id\_faculty = @faculty and gmail=@email and username=@username";

try

{

if(dataconnecttion.GetConnection().State == System.Data.ConnectionState.Closed)

dataconnecttion.GetConnection().Open();

SqlCommand cmd = new SqlCommand(querry, dataconnecttion.GetConnection());

cmd.Parameters.AddWithValue("@username", UsernameBox.Text);

cmd.Parameters.AddWithValue("@faculty",FalcultyBox.Text);

cmd.Parameters.AddWithValue("@email",EmailBox.Text);

int check = Convert.ToInt32(cmd.ExecuteScalar());

if (check == 1)

{

string querry2 = "select passwords from Account where username =@username";

cmd = new SqlCommand(querry2, dataconnecttion.GetConnection());

cmd.Parameters.AddWithValue("@username", UsernameBox.Text);

string to = EmailBox.Text;

string from = "imhunggg02@gmail.com";

string subject = "Recovery password";

string body = @"Your password is: " + Convert.ToString(cmd.ExecuteScalar());

string password = "stfhexhuhwbogtjc";

MailMessage message = new MailMessage();

message.To.Add(to);

message.From = new MailAddress(from);

message.IsBodyHtml = true;

message.Body = body;

message.Subject = subject;

SmtpClient smtp = new SmtpClient("smtp.gmail.com");

smtp.EnableSsl = true;

smtp.Port = 587;

smtp.DeliveryMethod = SmtpDeliveryMethod.Network;

smtp.UseDefaultCredentials = false;

smtp.Credentials = new NetworkCredential(from, password);

try

{

smtp.Send(message);

MessageBox.Show("Email sent");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

else

MessageBox.Show("Incorrect information, please fill in again");

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

Đoạn mã trên được dùng để kiểm tra thông tin người dùng nhập và gửi email gồm mật khẩu của người dùng thông qua email.

**Màn hình Attendance:**

+ Giao diện: Hiển thị thông tin nhân viên đi làm và thông tin nhân viên vắng mặt và số lượng nhân viên đi làm và vắng mặt. Nhận vào hình ảnh của nhân viên thông qua webcam.

+ Mã lệnh:

var frame = \_videoSource.QueryFrame().ToImage<Bgr, byte>().Resize(640, 480, Inter.Cubic);

var grayFrame = frame.Convert<Gray, byte>();

// Detect faces in current frame

var faces = \_faceClassifier.DetectMultiScale(grayFrame, 1.2, 5);

foreach (var face in faces)

{

// Extract face region of interest

var faceRect = new Rectangle(face.X, face.Y, face.Width, face.Height);

var faceImage = grayFrame.Copy(faceRect).Resize(200, 200, Inter.Cubic);

// Recognize face

var result = \_recognizer.Predict(faceImage);

// Display result

var label = result.Label.ToString();

if (checkId(Int32.Parse(label)))

{

//Display worker name if recognized

var worker = workerLabels.FirstOrDefault(w => w.Id == Int32.Parse(label));// take worker id

if (worker != null)

{

Dispatcher.Invoke(() =>

{

//KHi nhan dien ra cong nhan

if (CheckAttandance(Int32.Parse(label),\_faculty,shift))

{

string querry = "INSERT INTO Attendance VALUES (@id\_worker, @date, @fid, @shift)";

if (dataconnecttion.GetConnection().State == System.Data.ConnectionState.Closed)

dataconnecttion.GetConnection().Open();

command = new SqlCommand(querry, dataconnecttion.GetConnection());

//command.Parameters.AddWithValue("@id", id);

command.Parameters.AddWithValue("@id\_worker", Int32.Parse(label));

command.Parameters.AddWithValue("@date", DateTime.Now);

command.Parameters.AddWithValue("@fid", \_faculty);

command.Parameters.AddWithValue("@shift", shift);

command.ExecuteNonQuery();

dataconnecttion.GetConnection().Close();

}

AttendanceWorker temp = new AttendanceWorker();

temp.name = worker.Name;

temp.id = Int32.Parse(label);

temp.date = DateTime.Now;

Attendance.Items.Add(new { id\_worker = worker.Id, Name = worker.Name, TimeIn = DateTime.Now });

workerLabels.Remove(worker);

AttendanceTxt.Text = Attendance.Items.Count.ToString();

});

}

}

// Draw a rectangle around the face

frame.Draw(faceRect, new Bgr(0, 0, 255), 2);

}

// Display the processed frame in the image control

var bitmap = frame.ToBitmap();

var bitmapSource = Imaging.CreateBitmapSourceFromHBitmap(

bitmap.GetHbitmap(), IntPtr.Zero, Int32Rect.Empty, BitmapSizeOptions.FromEmptyOptions());

VideoDisplay.Source = bitmapSource;

}

Đoạn mã trên được sử dụng để nhận diện nhân viên và hiển thị họ lên bảng. Khi quản lý bật vào màn hình thì hệ thống sẽ load dữ liệu của nhân viên bao gồm id và ảnh, sau đó hệ thống sẽ dùng 2 List để lưu trữ 2 mảng giá trị id và ảnh của nhân viên theo 1 cặp trị số giống nhau. Ở đây, ta dùng List để dễ dàng thêm và xóa phần tử cũng như không phải cấp phát bộ nhớ trước khi lưu dữ liệu. Khi hệ thống nhận diện được mặt người sẽ ghi dữ liệu của nhân viên đó vào cơ sở dữ liệu để tránh mất mát dữ liệu và xóa id và ảnh của nhân viên đó ra khỏi 2 List đã tạo để tránh trùng lặp nhân viên khi chấm công. Ngoài ra, để nhận được dữ liệu từ webcam, ta sử dụng lớp VideoCapture để có thể kết nối đến webcam và bật/tắt tùy ý.

**Màn hình Request(Staff):**

+ Giao diện: Nhận vào thông tin người dùng nhập gồm: id nhân viên, tên, ngày làm, ca làm và danh sách các yêu cầu đã gửi. Hiển thị thông báo khi tìm dữ liệu về nhân viên cần được xóa chấm công và gửi thành công yêu cầu.

+ Mã lệnh:

string query = "Select detail, states, usernamesent from Request where usernamesent = @username";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

dtc.GetConnection().Open();

try

{

cmd = new SqlCommand(query, dtc.GetConnection());

cmd.Parameters.AddWithValue("@username", username);

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

string detail = reader.GetString(0);

string states = reader.GetString(1);

string usernamesent = reader.GetString(2);

RequestSentdtg.Items.Add(new { detail = detail, states = states, usernamesent = usernamesent });

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

Đoạn mã trên được dùng để hiển thị lên những yêu cầu của người dùng đã gửi. Bắt lỗi dữ liệu truyền vào, lỗi khi liên kết đến cơ sở dữ liệu.

int attendanceId = getID();

if(attendanceId != null)

{

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

dtc.GetConnection().Open();

string query = "Insert into Request values(@attendanceId, @detail, @states, @usernamesent)";

try

{

string detail = "Delete worker: " + Nametxb.Text + " Id: " + Idtxb.Text + " At: " + Datetxb.Text + " Shift: " + Shiftcbb.SelectedItem.ToString();

cmd = new SqlCommand(query, dtc.GetConnection());

cmd.Parameters.AddWithValue("@attendanceId", attendanceId);

cmd.Parameters.AddWithValue("@detail", detail);

cmd.Parameters.AddWithValue("states", "Idle");

cmd.Parameters.AddWithValue("@usernamesent", \_username);

cmd.ExecuteNonQuery();

MessageBox.Show("Successful sent request");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

else

{

MessageBox.Show("Wrong worker's information");

}

Đoạn mã trên dùng để kiểm tra thông tin khi người dùng nhập vào và gửi yêu cầu nếu như thông tin người dùng nhập vào là đúng và hiển thị thống báo gửi yêu cầu thành công.

**6.3 Nguyễn Quang Hưng**

**Màn hình Worker manage:**

+ Giao diện: Hiển thị danh sách nhân viên, khung tìm kiếm và các nút thêm, sửa, xóa.

+Mã lệnh:

string query = "Select id,fullname, birth, salary ,fid from WorkerList Where fid = @fid";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

dtc.GetConnection().Open();

cmd = new SqlCommand(query, dtc.GetConnection());

try

{

cmd.Parameters.AddWithValue("@fid", \_faculty);

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

int id = Convert.ToInt32(reader.GetInt32(0));

string fullname = reader.GetString(1);

DateTime dob = reader.GetDateTime(2);

int salary = reader.GetInt32(3);

string fid = reader.GetString(4);

Workertxt.Items.Add(new { id = id, fullname = fullname, dob = dob,salary = salary, fid = fid });

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

Đoạn mã trên lấy ra thông tin của nhân viên và hiển thị danh sách lên datagrid. Bắt lỗi kết nối đến cơ sở dữ liệu.

**Màn hình Add worker:**

+Giao diện: Nhận các thông tin nhân viền mà người dùng nhập vào gồm: tên, ngày sinh, lương, phòng ban, ảnh. Hiển thị các ô trống để người dùng điền thông tin và nút lưu, chụp ảnh, bật cam. Hiển thị thêm các khung webcam và khung ảnh kết quả.

+ Mã lệnh:

if (iscapturing == true)

{

try

{

// Crop the face image

Rectangle face = faces[0];

Image<Gray, byte> faceImage = image.Convert<Gray, byte>().Copy(face);

string nameimg = FullNametxt.Text;

int idNext = getIdWorker\_Next();

string imagePath = $"{resourceFolderPath}\\WorkerImage\\{Falcutybox.SelectedItem.ToString()}\\{nameimg}{idNext}.png";

if (File.Exists(imagePath))

{

ResultImg.Source = null;

File.Delete(imagePath);

}

Resize(faceImage).Save(imagePath);

BitmapImage bitmap = new BitmapImage();

bitmap.BeginInit();

bitmap.CacheOption = BitmapCacheOption.OnLoad;

bitmap.UriSource = new Uri(imagePath, UriKind.Absolute);

bitmap.EndInit();

ResultImg.Source = bitmap;

MessageBox.Show("Add worker picture done.");

}

catch(Exception ex)

{

MessageBox.Show("Please, try add picture again", "Error add picture", MessageBoxButton.OK);

}

Đoạn mã trên để chụp ảnh nhân viên bằng webcam. Khi người dùng bấm chụp thì hệ thống sẽ nhận diện khuôn mặt của nhân viên và cắt ảnh chỉ còn mặt của nhân viên và hiện thị thông báo khi chụp ảnh thành công. Bắt lỗi khi trong anh chụp không có khuôn mặt nào.

string querry = "Insert into WorkerList Values(@fullname, @birth,@salary ,@images, @fid)";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

dtc.GetConnection().Open();

int id\_worker = getIdWorker\_Next();

try

{

SQLcommand = new SqlCommand(querry, dtc.GetConnection());

SQLcommand.Parameters.Add("@fullname", SqlDbType.NVarChar).Value = fullname;

SQLcommand.Parameters.Add("@birth", SqlDbType.Date).Value = DoB.Date.ToString();

SQLcommand.Parameters.Add("@fid", SqlDbType.NVarChar).Value = faculty;

SQLcommand.Parameters.Add("@images", SqlDbType.NVarChar).Value = fullname+id\_worker;

SQLcommand.Parameters.AddWithValue("@salary", Int32.Parse(Salarytxt.Text));

SQLcommand.ExecuteNonQuery();

MessageBox.Show("Worker added successfully.");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

Đoạn mã trên được dùng để ghi dữ liệu của nhân viên vừa được thêm vào cơ sở dữ liệu và lưu ảnh chụp vào đúng folder của hệ thống. Bắt lỗi liên kết đến cơ sở dữ liệu, sai định dạng dữ liệu.

**Màn hình Edit worker:**

+ Giao diện: Nhận vào thông tin người dùng muốn sửa, thông tin người dùng đã sửa. Hiển thị thông tin người dùng muốn sửa và thông báo khi sửa thành công.

+ Mã lệnh:

FullNametxt.Text = worker.Fullname;

Dobtxt.Text = worker.Birth.ToString();

Facultycbb.SelectedItem = worker.Fid;

Salarytxt.Text = worker.Salary.ToString();

string query = "Select images, id from WorkerList where fullname =@fullname";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

dtc.GetConnection().Open();

try{

cmd = new SqlCommand(query, dtc.GetConnection());

cmd.Parameters.AddWithValue("@fullname", fullname);

SqlDataReader reader = cmd.ExecuteReader();

string imageName = " ";

int idReader = 0;

while (reader.Read())

{

imageName = reader.GetString(0);

idReader = reader.GetInt32(1);

}

reader.Close();

}catch(Exception ex){

MessageBox.Show(ex.Message);

}

id = idReader;

string currentImage = $"{resourceFolderPath}\\WorkerImage\\{\_faculty}\\{imageName}.png";

BitmapImage bitmap = new BitmapImage();

bitmap.BeginInit();

bitmap.CacheOption = BitmapCacheOption.OnLoad;

bitmap.UriSource = new Uri(currentImage, UriKind.Absolute);

bitmap.EndInit();

ResultImg.Source = bitmap;

dtc.GetConnection().Close();

Đoạn mã trên được dùng để hiển thị các thông tin của tài khoản mà người dùng muốn sửa bao gồm cả ảnh hồ sơ. Dùng DataReader để lấy ra từng dữ liệu sau khi thực thi câu lệnh sql. Bắt lỗi kết nối đến cơ sở dữ liệu và lỗi không tìm được thông tin.

string query = "Update WorkerList set fullname = @fullname, Birth = @dob, fid =@fid, images = @image, salary = @salary where id = @id";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

dtc.GetConnection().Open();

cmd = new SqlCommand(query, dtc.GetConnection());

cmd.Parameters.AddWithValue("@fullname", FullNametxt.Text);

cmd.Parameters.AddWithValue("@dob", DoB);

cmd.Parameters.AddWithValue("@fid", Facultycbb.SelectedItem.ToString());

cmd.Parameters.AddWithValue("@image", FullNametxt.Text + id);

cmd.Parameters.AddWithValue("@id", id);

cmd.Parameters.AddWithValue("@salary", Int32.Parse(Salarytxt.Text.ToString()));

cmd.ExecuteNonQuery();

//Image processing

if (File.Exists($"D:\\{FullNametxt.Text}{id}.png") || FullNametxt.Text != fullname)

{

//change everything -> delete old image and replace

//Delete old image

string oldpath = $"{resourceFolderPath}\\WorkerImage\\{\_faculty}\\{fullname}{id}.png";

File.Delete(oldpath);

//Copy new image to directed folder

string targetpath = $"{resourceFolderPath}\\WorkerImage\\{Facultycbb.SelectedItem.ToString()}\\{FullNametxt.Text}{id}.png";

File.Copy($"D:\\{FullNametxt.Text}.png", targetpath);

File.Delete($"D:\\{FullNametxt.Text}{id}.png");

}

else if (FullNametxt.Text == fullname && Facultycbb.SelectedItem.ToString() == \_faculty)

{

//only change dob ->do not thing

}

else

{

//only change falculty-> Move old iamge to new folder

string oldpath = $"{resourceFolderPath}\\WorkerImage\\{\_faculty}\\{fullname}{id}.png";

string targetpath = $"{resourceFolderPath}\\WorkerImage\\{Facultycbb.SelectedItem.ToString()}\\{FullNametxt.Text}{id}.png";

File.Move(oldpath, targetpath);

}

MessageBox.Show("Successful change worker's information");

Đoạn mã trên được sử dụng để lưu thông tin vừa sửa đổi vào cơ sở dữ liệu. Xử lý file ảnh tùy thuộc vào các thay đổi về nhân viên đó. Khi chỉ thay đổi ngày sinh thì không thay đổi đến file ảnh. Khi thay đổi tên nhân viên nên tên ảnh thay đổi, ta xóa ảnh cũ đi và lưu ảnh mới vào đúng thư mục. Còn khi chỉ thay đổi phòng ban thì ta chỉ cần chuyển file ảnh cũ sang đúng folder. Ta sử dụng File.Delete() để xóa file còn File.Copy() để di chuyển file sang folder khác.

**6.2 Nguyễn Khánh Thọ**

**Màn hình Report\_Day:**

+ Giao diện: Hiển thị các button với các chức năng tương ứng để actor tương tác, hiển thị ô chọn ngày Date Ficker để chọn ngày, sau khi chọn ngày và bấm tìm kiếm, đổ dữ liệu ra các bảng DataGrid, hiển thị ComboBox để chọn ca 1, ca 2 (hiển thị bảng danh sách của ca tương ứng).

+ Mã lệnh:

*Hàm tìm kiếm dữ liệu:*

private void Search\_Button\_Click(object sender, RoutedEventArgs e)

{

if (Date\_DatePicker.SelectedDate != null)

{

DateTime date = Convert.ToDateTime(Date\_DatePicker.SelectedDate);

AttandanceWorkers\_DataGrid\_1.Items.Clear();

AttandanceWorkers\_DataGrid\_2.Items.Clear();

LateWorkers\_DataGrid\_1.Items.Clear();

LateWorkers\_DataGrid\_2.Items.Clear();

AbsenteeWorkers\_DataGrid\_1.Items.Clear();

AbsenteeWorkers\_DataGrid\_2.Items.Clear();

try

{

for (int i = 1; i <= 2; i++) // 2 ca

{

string query1 = "SELECT id\_worker,fullname,d\_m " +

"FROM Attendance " +

"INNER JOIN WorkerList " +

"ON Attendance.id\_worker = WorkerList.id " +

"WHERE id\_faculty = @fid " +

"AND shift\_worked = @i " +

"AND CONVERT(date,d\_m) = @date";

string query2 = "SELECT id\_worker,fullname,d\_m " +

"FROM LateList " +

"INNER JOIN WorkerList " +

"ON LateList.id\_worker = WorkerList.id " +

"WHERE id\_faculty = @fid " +

"AND shift\_worked = @i " +

"AND CONVERT(date,d\_m) = @date";

string query3 = "SELECT id, fullname " +

"FROM WorkerList " +

"WHERE id NOT IN (" +

"SELECT id\_worker " +

"FROM Attendance " +

"WHERE id\_faculty = @fid " +

"AND shift\_worked = @i " +

"AND CONVERT(date,d\_m) = @date " +

") " +

"AND id NOT IN (" +

"SELECT id\_worker " +

"FROM LateList " +

"WHERE id\_faculty = @fid " +

"AND shift\_worked = @i " +

"AND CONVERT(date,d\_m) = @date " +

") ";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

{

dtc.GetConnection().Open();

}

cmd = new SqlCommand(query1, dtc.GetConnection());

cmd.Parameters.AddWithValue("@fid", this.fid);

cmd.Parameters.AddWithValue("@i", i);

cmd.Parameters.AddWithValue("@date", date.Date);

SqlDataReader read1 = cmd.ExecuteReader();

while (read1.Read())

{

int id = read1.GetInt32(0);

string name = read1.GetString(1);

DateTime dt = read1.GetDateTime(2);

if (i == 1)

{

AttandanceWorkers\_DataGrid\_1.Items.Add(new { id = id, name = name, dt = dt });

}

else

{

AttandanceWorkers\_DataGrid\_2.Items.Add(new { id = id, name = name, dt = dt });

}

}

read1.Close();

cmd = new SqlCommand(query2, dtc.GetConnection());

cmd.Parameters.AddWithValue("@fid", this.fid);

cmd.Parameters.AddWithValue("@i", i);

cmd.Parameters.AddWithValue("@date", date.Date);

SqlDataReader read2 = cmd.ExecuteReader();

while (read2.Read())

{

int id = read2.GetInt32(0);

string name = read2.GetString(1);

DateTime dt = read2.GetDateTime(2);

if (i == 1)

{

LateWorkers\_DataGrid\_1.Items.Add(new { id = id, name = name, dt = dt });

}

else

{

LateWorkers\_DataGrid\_2.Items.Add(new { id = id, name = name, dt = dt });

}

}

read2.Close();

cmd = new SqlCommand(query3, dtc.GetConnection());

cmd.Parameters.AddWithValue("@fid", this.fid);

cmd.Parameters.AddWithValue("@i", i);

cmd.Parameters.AddWithValue("@date", date.Date);

SqlDataReader read3 = cmd.ExecuteReader();

while (read3.Read())

{

int id = read3.GetInt32(0);

string name = read3.GetString(1);

if (i == 1)

{

AbsenteeWorkers\_DataGrid\_1.Items.Add(new { id = id, name = name });

}

else

{

AbsenteeWorkers\_DataGrid\_2.Items.Add(new { id = id, name = name });

}

}

read3.Close();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error");

}

}

else MessageBox.Show("Please choose a date", "Message");

}

Đoạn mã trên được dùng để lấy các danh sách các nhân viên trong phòng ban đi làm đúng giờ, đi làm muộn, vắng mặt trong ngày đã chọn trước đó. Đoạn code trên sử dụng 3 câu lệnh sql với mỗi lần chạy trong vòng for với i chạy từ 1 tới 2 tương đương với 2 ca làm việc. 3 lệnh truy xuất dữ liệu tương đương với dữ liệu cả bảng nhân viên đi làm đúng giờ, đi làm muộn, vắng mặt của mỗi ca. Sử dụng SqlCommand để thực thi câu lệnh rồi dùng SqlDataReader để đọc dữ liệu sau khi thực thi lệnh sql. Từ đó đưa các thành phần vào danh sách của bảng tương ứng. Bắt lỗi nếu có lỗi trong khi truy vấn dữ liệu bị lỗi và hiển thị lên màn hình, và nếu khi chưa chọn ngày thì sẽ đưa ra thông báo lỗi yêu cầu chọn ngày.

*Hàm xuất file excel báo cáo theo ngày:*

private void Export\_Button\_Click(object sender, RoutedEventArgs e)

{

if (AttandanceWorkers\_DataGrid\_1.Items.Count == 0 &&

LateWorkers\_DataGrid\_1.Items.Count == 0 &&

AbsenteeWorkers\_DataGrid\_1.Items.Count == 0)

{

MessageBox.Show("Data in the table is empty", "Error", MessageBoxButton.OK);

}

else

{

SaveFileDialog savefile = new SaveFileDialog();

savefile.DefaultExt = ".xlsx";

savefile.Filter = "Excel Files|\*xlsx;\*xls;\*xlsm";

savefile.FileName = "ReportDay.xlsx";

if (savefile.ShowDialog() == true)

{

try

{

var workbook = new XLWorkbook();

workbook.Style.Font.FontSize = 11;

workbook.Style.Font.FontName = "Times New Roman";

var sheet1 = workbook.Worksheets.Add("Shift 1");

int rowWrite = 1;

Type t;

PropertyInfo[] p;

IXLCell cell;

//Attendance

cell = sheet1.Cell(rowWrite, 1);

cell.Value = "Attandance workers list";

cell.Style.Font.Bold= true;

cell.Style.Font.FontSize = 14;

rowWrite++;

for (int col = 0; col < AttandanceWorkers\_DataGrid\_1.Columns.Count; col++)

{

cell = sheet1.Cell(rowWrite, col + 1);

cell.Value = AttandanceWorkers\_DataGrid\_1.Columns[col].Header.ToString();

cell.Style.Fill.BackgroundColor = XLColor.Aqua;

cell.Style.Font.Bold = true;

cell.Style.Border.OutsideBorder = XLBorderStyleValues.Thin;

}

rowWrite++;

foreach (var item in AttandanceWorkers\_DataGrid\_1.Items)

{

for (int col = 0; col < AttandanceWorkers\_DataGrid\_1.Columns.Count; col++)

{

TextBlock Value = AttandanceWorkers\_DataGrid\_1.Columns[col].GetCellContent(item) as TextBlock;

cell = sheet1.Cell(rowWrite, col + 1);

cell.Value = Value.Text;

cell.Style.Fill.BackgroundColor = XLColor.FromHtml("#F2F2F2");

cell.Style.Border.OutsideBorder = XLBorderStyleValues.Thin;

}

rowWrite++;

}

rowWrite++;

//Late

cell = sheet1.Cell(rowWrite, 1);

cell.Value = "Late workers list";

cell.Style.Font.Bold = true;

cell.Style.Font.FontSize = 14;

rowWrite++;

for (int col = 0; col < LateWorkers\_DataGrid\_1.Columns.Count; col++)

{

cell = sheet1.Cell(rowWrite, col + 1);

cell.Value = LateWorkers\_DataGrid\_1.Columns[col].Header.ToString();

cell.Style.Fill.BackgroundColor = XLColor.Aqua;

cell.Style.Font.Bold = true;

cell.Style.Border.OutsideBorder = XLBorderStyleValues.Thin;

}

rowWrite++;

foreach (var item in LateWorkers\_DataGrid\_1.Items)

{

for (int col = 0; col < LateWorkers\_DataGrid\_1.Columns.Count; col++)

{

TextBlock Value = LateWorkers\_DataGrid\_1.Columns[col].GetCellContent(item) as TextBlock;

cell = sheet1.Cell(rowWrite, col + 1);

cell.Value = Value.Text;

cell.Style.Fill.BackgroundColor = XLColor.FromHtml("#F2F2F2");

cell.Style.Border.OutsideBorder = XLBorderStyleValues.Thin;

}

rowWrite++;

}

rowWrite++;

//Absentee

cell = sheet1.Cell(rowWrite, 1);

cell.Value = "Absentee workers list";

cell.Style.Font.Bold = true;

cell.Style.Font.FontSize= 14;

rowWrite++;

for (int col = 0; col < AbsenteeWorkers\_DataGrid\_1.Columns.Count; col++)

{

cell = sheet1.Cell(rowWrite, col + 1);

cell.Value = AbsenteeWorkers\_DataGrid\_1.Columns[col].Header.ToString();

cell.Style.Fill.BackgroundColor = XLColor.Aqua;

cell.Style.Font.Bold = true;

cell.Style.Border.OutsideBorder = XLBorderStyleValues.Thin;

}

rowWrite++;

foreach (var item in AbsenteeWorkers\_DataGrid\_1.Items)

{

for (int col = 0; col < AbsenteeWorkers\_DataGrid\_1.Columns.Count; col++)

{

TextBlock Value = AbsenteeWorkers\_DataGrid\_1.Columns[col].GetCellContent(item) as TextBlock;

cell = sheet1.Cell(rowWrite, col + 1);

cell.Value = Value.Text;

cell.Style.Fill.BackgroundColor = XLColor.FromHtml("#F2F2F2");

cell.Style.Border.OutsideBorder = XLBorderStyleValues.Thin;

}

rowWrite++;

}

rowWrite++;

sheet1.Columns().AdjustToContents(); // adjust column width auto fit with contents

////////// sheet2 ///////////////////////////////////////////////////////////

rowWrite = 1;

var sheet2 = workbook.Worksheets.Add("shift 2");

//Attandance

cell = sheet2.Cell(rowWrite, 1);

cell.Value = "Attandance workers list";

cell.Style.Font.Bold = true;

cell.Style.Font.FontSize = 14;

rowWrite++;

for (int col = 0; col < AttandanceWorkers\_DataGrid\_2.Columns.Count; col++)

{

cell = sheet2.Cell(rowWrite, col + 1);

cell.Value = AttandanceWorkers\_DataGrid\_2.Columns[col].Header.ToString();

cell.Style.Fill.BackgroundColor = XLColor.Aqua;

cell.Style.Font.Bold = true;

cell.Style.Border.OutsideBorder = XLBorderStyleValues.Thin;

}

rowWrite++;

foreach (var item in AttandanceWorkers\_DataGrid\_2.Items)

{

for (int col = 0; col < AttandanceWorkers\_DataGrid\_2.Columns.Count; col++)

{

TextBlock Value = AttandanceWorkers\_DataGrid\_2.Columns[col].GetCellContent(item) as TextBlock;

cell = sheet2.Cell(rowWrite, col + 1);

cell.Value = Value.Text;

cell.Style.Fill.BackgroundColor = XLColor.FromHtml("#F2F2F2");

cell.Style.Border.OutsideBorder = XLBorderStyleValues.Thin;

}

rowWrite++;

}

rowWrite++;

//Late

cell = sheet2.Cell(rowWrite, 1);

cell.Value = "Late workers list";

cell.Style.Font.Bold = true;

cell.Style.Font.FontSize = 14;

rowWrite++;

for (int col = 0; col < LateWorkers\_DataGrid\_2.Columns.Count; col++)

{

cell = sheet2.Cell(rowWrite, col + 1);

cell.Value = LateWorkers\_DataGrid\_2.Columns[col].Header.ToString();

cell.Style.Fill.BackgroundColor = XLColor.Aqua;

cell.Style.Font.Bold = true;

cell.Style.Border.OutsideBorder = XLBorderStyleValues.Thin;

}

rowWrite++;

foreach (var item in LateWorkers\_DataGrid\_2.Items)

{

for (int col = 0; col < LateWorkers\_DataGrid\_2.Columns.Count; col++)

{

TextBlock Value = LateWorkers\_DataGrid\_2.Columns[col].GetCellContent(item) as TextBlock;

cell = sheet2.Cell(rowWrite, col + 1);

cell.Value = Value.Text;

cell.Style.Fill.BackgroundColor = XLColor.FromHtml("#F2F2F2");

cell.Style.Border.OutsideBorder = XLBorderStyleValues.Thin;

}

rowWrite++;

}

rowWrite++;

//Absentee

cell = sheet2.Cell(rowWrite, 1);

cell.Value = "Absentee workers list";

cell.Style.Font.Bold = true;

cell.Style.Font.FontSize = 14;

rowWrite++;

for (int col = 0; col < AbsenteeWorkers\_DataGrid\_2.Columns.Count; col++)

{

cell = sheet2.Cell(rowWrite, col + 1);

cell.Value = AbsenteeWorkers\_DataGrid\_2.Columns[col].Header.ToString();

cell.Style.Fill.BackgroundColor = XLColor.Aqua;

cell.Style.Font.Bold = true;

cell.Style.Border.OutsideBorder = XLBorderStyleValues.Thin;

}

rowWrite++;

foreach (var item in AbsenteeWorkers\_DataGrid\_2.Items)

{

for (int col = 0; col < AbsenteeWorkers\_DataGrid\_2.Columns.Count; col++)

{

TextBlock Value = AbsenteeWorkers\_DataGrid\_2.Columns[col].GetCellContent(item) as TextBlock;

cell = sheet2.Cell(rowWrite, col + 1);

cell.Value = Value.Text;

cell.Style.Fill.BackgroundColor = XLColor.FromHtml("#F2F2F2");

cell.Style.Border.OutsideBorder = XLBorderStyleValues.Thin;

}

rowWrite++;

}

rowWrite++;

sheet2.Columns().AdjustToContents();

workbook.SaveAs(savefile.FileName);

MessageBox.Show("Done", "Message", MessageBoxButton.OK);

workbook.Dispose();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

Đoạn mã trên được dùng để xuất các danh sách các nhân viên trong phòng ban đi làm đúng giờ, đi làm muộn, vắng mặt trong ngày đã chọn trước đó ra file excel. Đoạn code trên sử dụng thư viện ClosedXML để hỗ trợ tạo tùy chỉnh file excel. Bắt đầu khi bấm button Export trên màn hình, hệ thống sử dụng SaveFileDiaLog để mở 1 của sổ cho actor chọn chỗ lưu file và chỉnh sửa tên file. Sau khi chọn xong, sử dụng ClosedXML để tạo ra file excel gồm 2 sheet tương đương với 2 ca, ở mỗi sheet sẽ có 3 bảng dữ liệu danh sách như đã nói. Ở đây để lấy dữ liệu của từng cột trong DataGrid thì ta dùng hàm GetCellContent() của lớp Columns với giá trị truyền vào là item của DataGrid , lấy dữ liệu thông qua TextBlock. Bắt lỗi nếu có lỗi trong khi quá trình chạy bị lỗi và hiển thị lên màn hình.

*Hàm gửi file báo cáo theo ngày qua email:*

private void Send\_Button\_Click(object sender, RoutedEventArgs e)

{

SendEmailUI win = new SendEmailUI();

win.Show();

}

Đoạn mã trên được dùng để gửi báo cáo qua email, cách dùng là gọi cửa sổ SendEmail hiển thị lên màn hình , đoạn code này được áp dụng tương tự đối với tất cả các chức năng gửi email trên hệ thống (chức năng xem báo cáo theo tháng và chức năng xem chấm công.

**Màn hình Report\_Month:**

+ Giao diện: Hiển thị các button với các chức năng tương ứng để actor tương tác, hiển thị ô nhập dữ liệu cho tháng và năm, hiển thị bảng dữ liệu các nhân viên với thông tin số ca đi làm đúng giờ, ca đi muộn và ca vắng mặt.

+ Mã lệnh:

*Hàm tìm kiếm dữ liệu:*

private void Search\_Button\_Click(object sender, RoutedEventArgs e)

{

if (Check\_DMY())

{

MonthCount\_DataGrid.Items.Clear();

int month = int.Parse(Month\_TextBox.Text.ToString());

int year = int.Parse(Year\_TextBox.Text.ToString());

int CountDay = DateTime.DaysInMonth(year, month);

try

{

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

{

dtc.GetConnection().Open();

}

string querry = "SELECT \* FROM WorkerList WHERE fid = '" + this.fid + "' ";

cmd = new SqlCommand(querry, dtc.GetConnection());

List<WorkerList> listWorker = new List<WorkerList>();

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

WorkerList item = new WorkerList();

item.Id = reader.GetInt32(0);

item.Fullname = reader.GetString(1);

item.Birth = reader.GetDateTime(2);

item.Salary = reader.GetInt32(3);//fix

item.Images= reader.GetString(4);

item.Fid = reader.GetString(5);

listWorker.Add(item);

}

reader.Close();

string monthYear = month+"/"+ year;

foreach (WorkerList item in listWorker)

{

string querryAttend = "SELECT COUNT(shift\_worked) FROM Attendance " +

"WHERE id\_worker = @id\_worker " +

"AND MONTH(d\_m) = @month " +

"AND YEAR(d\_m) = @year ";

cmd = new SqlCommand(querryAttend, dtc.GetConnection());

cmd.Parameters.AddWithValue("@id\_worker", item.Id);

cmd.Parameters.AddWithValue("@month", month);

cmd.Parameters.AddWithValue("@year", year);

int attend = Convert.ToInt32(cmd.ExecuteScalar());

string querryLate = "SELECT COUNT(shift\_worked) FROM LateList " +

"WHERE id\_worker = @id\_worker " +

"AND MONTH(d\_m) = @month " +

"AND YEAR(d\_m) = @year ";

cmd = new SqlCommand(querryLate, dtc.GetConnection());

cmd.Parameters.AddWithValue("@id\_worker", item.Id);

cmd.Parameters.AddWithValue("@month", month);

cmd.Parameters.AddWithValue("@year", year);

int late = Convert.ToInt32(cmd.ExecuteScalar());

// all of shift in month - all of worked = absent (shift not working)

int absent = CountDay \* 2 - (attend + late);

MonthCount\_DataGrid.Items.Add(new {id = item.Id , name = item.Fullname , monthYear = monthYear , worked = attend+late , late = late, absent = absent});

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error");

}

}

}

Đoạn mã trên được dùng để lấy các danh sách thống kê số ca đi làm của các nhân viên trong phòng ban. Đoạn code trên sử dụng SqlCommand và SqlDataReader để chạy và lấy dữ liệu các nhân viên từ bảng WorkerList. Với mỗi nhân viên ta lấy thông tin id\_worker, tháng và năm mà actor đã nhập từ màn hình để đếm số ca mà nhân viên đó đi làm đúng giờ, đi làm muộn. Từ đó lấy tổng số ca đi làm cả tháng trừ cho số tổng số ca đi làm thực tế để lấy được số ca vắng mặt. Rồi hiển thị danh sách dữ liệu tính được ra bảng. Bắt lỗi nếu có lỗi trong khi truy vấn dữ liệu bị lỗi và hiển thị lên màn hình, và nếu khi chưa chọn ngày thì sẽ đưa ra thông báo lỗi yêu cầu chọn ngày.

.

*Hàm Export file excel và hàm Send file qua email* thì gọi hàm từ class ExportEmail (trình bày phía dưới) đã tạo và cửa sổ gửi email đã tạo (tương tự với đoạn mã của hàm Send trên màn hình Report\_Day

**Màn hình Send\_Email:**

+ Giao diện: Hiển thị form nhập thông tin dữ liệu cần thiết để gửi file. Với đầu vào gồm những thông tin: email người nhận, tiêu đề, nội dung, địa chỉ file trong máy. và chương trình sẽ thực hiện gửi email rồi đưa ra thông báo

+ Mã lệnh:

*Hàm gửi email:*

private void Send\_Button\_Click(object sender, RoutedEventArgs e)

{

string toAddress = ToEmail\_TextBox.Text;

string subject = Subject\_TextBox.Text;

string content = Content\_TextBox.Text;

string[] fileName = ChooseFile\_TextBlock.Text.Split('\n', StringSplitOptions.RemoveEmptyEntries); // remove null or "" path in string

SendEmail send = new SendEmail();

send.SendEmail\_FileChoose(toAddress, subject, content, fileName);

}

Đoạn mã trên lấy dữ liệu được nhập truyền vào hàm SendEmail\_FileChoose của class SendEmail đã được tạo từ trước để thực hiện việc xuất file

public void SendEmail\_FileChoose(string toAddress, string subject, string content, string[] filename)

{

try

{

MailMessage mail = new MailMessage();

SmtpClient smtp = new SmtpClient("smtp.gmail.com");

mail.From = new MailAddress("imhunggg02@gmail.com");

mail.To.Add(toAddress);

mail.Subject = subject;

mail.Body = content;

foreach (string file in filename)

{

Attachment attachFile = new Attachment(file);

mail.Attachments.Add(attachFile);

}

smtp.Port = 587;

smtp.DeliveryMethod = SmtpDeliveryMethod.Network;

smtp.Credentials = new System.Net.NetworkCredential("imhunggg02@gmail.com", "stfhexhuhwbogtjc");//username and password

smtp.EnableSsl = true;

smtp.Send(mail);

MessageBox.Show("Mail send successfully");

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

Đoạn code trên gọi namespace System.NET.Mail gọi lớp MailMessage để tạo 1 mail để gửi, rồi dùng lớp SmtpClient để gửi email qua giao thức SMTP, từ đó ta lấy thông tin truyền vào nhập cho mail. Với mảng string[] filename là mảng chuỗi chứa thông tin địa chỉ file trong máy mà actor muốn đính kèm vào mail, ta gọi thuộc tính Attachments của mail để thêm file đính kèm đó vào. Sau đó set cấu hình cho giao thức rồi gửi email. Bắt lỗi nếu có lỗi trong quá trình thực hiện bị lỗi rồi hiển thị lỗi lên màn hình

**Hàm ExportExcel\_DataGrid trong lớp ExportExcel:**

+ Mã lệnh:

*Hàm xuất file email 1 bảng DataGrid bất kì:*

public void ExportExcel\_DataGrid(DataGrid temp)

{

if (temp.Items.Count > 0)

{

SaveFileDialog saveDialog = new SaveFileDialog();

saveDialog.DefaultExt= ".xlsx";

saveDialog.Filter = "Excel Files|\*xlsx;\*xls;\*xlsm";

saveDialog.FileName = "Report.xlsx";

if(saveDialog.ShowDialog()==true)

{

try

{

var workbook = new XLWorkbook();

var sheet = workbook.Worksheets.Add(temp.Name);

int rowWrite = 1;

IXLCell cell;

cell = sheet.Cell(rowWrite, 1);

cell.Value= temp.Name;

cell.Style.Font.Bold = true;

cell.Style.Font.FontSize = 14;

rowWrite++;

for(int col = 0; col< temp.Columns.Count; col++)

{

cell = sheet.Cell(rowWrite, col + 1);

cell.Value = temp.Columns[col].Header.ToString();

cell.Style.Fill.BackgroundColor = XLColor.Aqua;

cell.Style.Font.Bold = true;

cell.Style.Border.OutsideBorder = XLBorderStyleValues.Thin;

}

rowWrite++;

foreach (var item in temp.Items)

{

for(int col = 0; col < temp.Columns.Count; col++)

{

TextBlock Value = temp.Columns[col].GetCellContent(item) as TextBlock;

cell = sheet.Cell(rowWrite, col + 1);

cell.Value = Value.Text;

cell.Style.Fill.BackgroundColor = XLColor.FromHtml("#F2F2F2");

cell.Style.Border.OutsideBorder = XLBorderStyleValues.Thin;

}

rowWrite++;

}

workbook.SaveAs(saveDialog.FileName);

MessageBox.Show("Export to excel successfull", "Message");

workbook.Dispose();

}

catch(Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

}

else

{

MessageBox.Show("Data in Table is nothing","Error",MessageBoxButton.OK);

}

}  
Đoạn code trên làm tương tự với phần xuất file báo cáo theo ngày. Hàm có tham số truyền vào là 1 DataGrid. Gọi cửa sổ SaveFileDiaLog để actor đặt tên file rồi chọn vị trí lưu. Đoạn code sử dụng thư viện ClosedXML để xuất ra file excel. Trong file Excel sẽ có thông tin tên của datagrid ở ô đầu tiên và sau đó là bảng DataGrid, trong bảng đó là tiêu đề của các cột ở hàng tiếp theo, và sau đó là thông tin của các item của DataGrid được lấy bằng hàm GetCellContent() của DataGird.Column được lấy qua TextBlock. Các ô được set Style như trong hàm. Bắt lỗi nếu có lỗi trong quá trình thực hiện bị lỗi rồi hiển thị lỗi lên màn hình.

**Màn hình Late\_Manage:**

+ Giao diện: Hiển thị các button với các chức năng tương ứng để actor tương tác, hiển thị các ô nhập dữ liệu, với đầu vào gồm tên nhân viên, tháng, năm hoặc là chọn ngày, chương trình sẽ hiển thị danh sách nhân viên đi muộn thỏa mãn điều kiện.

+ Mã lệnh:

*Hàm tìm kiếm bảng theo ngày:*

private void Date\_DatePicker\_SelectedDateChanged(object sender, SelectionChangedEventArgs e)

{

if (Date\_DatePicker.SelectedDate != null)

{

try

{

Late\_DataGrid.Items.Clear();

DateTime dateSearch = Convert.ToDateTime(Date\_DatePicker.SelectedDate.Value);

string querry = "SELECT LateList.id, LateList.id\_worker, " +

"fullname, d\_m, shift\_worked, detail " +

"FROM LateList INNER JOIN WorkerList ON " +

"LateList.id\_worker = WorkerList.id " +

"AND id\_faculty = @faculty " +

"WHERE d\_m = @date ";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

{

dtc.GetConnection().Open();

}

cmd = new SqlCommand(querry, dtc.GetConnection());

cmd.Parameters.AddWithValue("@date", dateSearch.Date);

cmd.Parameters.AddWithValue("@faculty", this.faculty);

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

int id = reader.GetInt32(0);

int id\_worker = reader.GetInt32(1);

string name\_worker = reader.GetString(2);

DateTime date = reader.GetDateTime(3);

int shift = reader.GetInt32(4);

string detail = reader.GetString(5);

Late\_DataGrid.Items.Add(new { Id = id, IdWorker = id\_worker, NameWorker = name\_worker, DateTime = date.ToString(), Shift = shift, Detail = detail });

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error");

}

}

else setLate\_DataGrid();

}

Đoạn mã trên được dùng để tìm kiếm danh sách nhân viên đi muộn theo ngày. Bắt lỗi nếu có lỗi trong khi truy vấn dữ liệu bị lỗi và hiển thị lên màn hình, và nếu khi chưa chọn ngày thì sẽ đưa ra thông báo lỗi yêu cầu chọn ngày.

*Hàm hiển thị dữ liệu lên DataGrid*

public void setLate\_DataGrid()

{

try

{

Late\_DataGrid.Items.Clear();

string querry = "SELECT LateList.id, LateList.id\_worker, " +

"fullname, d\_m, shift\_worked, detail " +

"FROM LateList INNER JOIN WorkerList ON " +

"LateList.id\_worker = WorkerList.id " +

"AND id\_faculty = '" + this.faculty + "' ";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

{

dtc.GetConnection().Open();

}

cmd = new SqlCommand(querry, dtc.GetConnection());

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

int id = reader.GetInt32(0);

int id\_worker = reader.GetInt32(1);

string name\_worker = reader.GetString(2);

DateTime date = reader.GetDateTime(3);

int shift = reader.GetInt32(4);

string detail = reader.GetString(5);

Late\_DataGrid.Items.Add(new { Id = id, IdWorker = id\_worker, NameWorker = name\_worker, DateTime = date.ToString(), Shift = shift, Detail = detail });

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error");

}

}

Đoạn hàm này giúp ta hiển thị dữ liệu danh sách tất cả những nhân viên đi làm muộn lên datagrid khi actor mở giao diện của chương trình. Bắt lỗi khi có lỗi trong khi chạy và hiển thị lỗi lên màn hình.

**Màn hình AddLate:**

+ Giao diện: Hiển thị các button với các chức năng tương ứng để actor tương tác, hiển thị các ô nhập dữ liệu, với đầu vào gồm tên nhân viên, ngày, ca, lý do đi muộn, chương trình sẽ thêm dữ liệu đó vào CSDL rồi đưa ra thông báo.

+ Mã lệnh:

*Hàm tìm kiếm bảng theo ngày:*

private void Add\_Button\_Click(object sender, RoutedEventArgs e)

{

if (checkInput())

{

int id\_worker = Convert.ToInt32(NameWorker\_ComboBox.SelectedValue);

DateTime date= Convert.ToDateTime(Date\_DatePicker.SelectedDate.Value);

int shift = Convert.ToInt32(Shift\_ComboBox.SelectedValue);

string detail = Detail\_TextBox.Text.Trim();

if (!checkAttendanced(id\_worker, date, shift) && !checkExist(id\_worker, date, shift))

{

try

{

if(dtc.GetConnection().State == System.Data.ConnectionState.Closed)

{

dtc.GetConnection().Close();

}

string querry = "INSERT INTO LateList VALUES (@id\_worker, @id\_faculty, @d\_m , @shift\_worked , @detail )";

cmd = new SqlCommand(querry, dtc.GetConnection());

cmd.Parameters.AddWithValue("@id\_worker", id\_worker);

cmd.Parameters.AddWithValue("@id\_faculty", this.faculty);

cmd.Parameters.AddWithValue("@d\_m", date.Date);

cmd.Parameters.AddWithValue("@shift\_worked", shift);

cmd.Parameters.AddWithValue("@detail", detail);

cmd.ExecuteNonQuery();

MessageBox.Show("Successful !", "Message");

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

}

Đoạn mã trên được dùng để thêm nhân viên đi muộn vào CSDL, sau khi kiểm tra điều kiện đầu vào có thỏa mãn (checkInput()) , kiểm tra điều kiện nhân viên đó đã điểm danh chưa (checkAttendanced()), kiểm tra điều kiện dữ liệu nhân viên đó đi làm muộn vào ngày đó ,ca đó có chưa (checkExist()) . Nếu điều kiện đầu vào chính xác và nhân viên chưa điểm danh (xác nhận là đi muộn), chưa tồn tại trong CSDL thì bắt đầu thêm vào CSDL. Dùng SqlCommad để thực hiện câu lệnh với ExecuteNonQuery() không trả về giá trị. Bắt lỗi nếu trong quá trình chạy có bất kì lỗi nào rồi hiển thị lỗi lên màn hình.

**6.3 Đinh Anh Quân**

**Màn hình Accountant:**

+ Giao diện: Hiển thị các button với các chức năng tương ứng để actor tương tác, hiển thị ô TextBlock, ComboBox nhập đầu vào: tháng, năm, phòng ban. Chương trình sẽ hiển thị bảng chấm công của nhân viên trong phòng ban đó lên màn hình gồm các dữ liệu : id nhân viên , tên nhân viên, số ca đi làm trong các ngày trong tháng, tổng số ca đi làm thực tế (là tổng số ca mà nhân viên đi lam đúng giơ và nhân viên đi làm muộn) , tổng số ca đi làm muộn và lương cơ bản của nhân viên trong tháng.

+ Mã lệnh:

*Hàm tìm hiển thị dữ liệu+:*

private void reloaddatagrid()

{

if (Year.Text.Length == 0)

{

WorkersDataGrid.Columns.Clear();

}

else

{

WorkersDataGrid.Columns.Clear();

string fid = Convert.ToString(facultycbb.SelectedItem);

int year = Convert.ToInt32(Year.Text);

int month = Convert.ToInt32(Monthcbb.SelectedItem);

// Add DataGrid columns dynamically

DataGridTextColumn column1 = new DataGridTextColumn();

column1.Header = "ID";

column1.Binding = new Binding("ID\_Worker");

WorkersDataGrid.Columns.Add(column1);

DataGridTextColumn column2 = new DataGridTextColumn();

column2.Header = "Name";

column2.Binding = new Binding("WorkerName");

WorkersDataGrid.Columns.Add(column2);

int DayOfMonth = 31;

switch (month)

{

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12:

DayOfMonth = 31;

break;

case 2:

if (year % 4 == 0 && year % 100 != 0 && year % 400 != 0)

{

DayOfMonth = 29;

}

else DayOfMonth = 28;

break;

case 4:

case 6:

case 9:

case 11:

DayOfMonth = 30;

break;

}

for (int i = 1; i <= DayOfMonth; i++)

{

DataGridTextColumn column = new DataGridTextColumn();

column.Header = i.ToString() + "/" + Monthcbb.SelectedValue;

column.Binding = new Binding($"Shifts[{i - 1}]");

WorkersDataGrid.Columns.Add(column);

}

DataGridTextColumn column3 = new DataGridTextColumn();

column3.Header = "Sum";

column3.Binding = new Binding("Sum");

WorkersDataGrid.Columns.Add(column3);

DataGridTextColumn column4 = new DataGridTextColumn();

column4.Header = "SumLate";

column4.Binding = new Binding("SumLate");

WorkersDataGrid.Columns.Add(column4);

DataGridTextColumn column5 = new DataGridTextColumn();

column4.Header = "Salary";

column4.Binding = new Binding("SalaryMonth");

WorkersDataGrid.Columns.Add(column5);

// Get Data In Database

List<Worker> source = new List<Worker>();//items in datagrid

List<WorkerList> listWorker = new List<WorkerList>();

try

{

string querry1 = "SELECT \* FROM WorkerList WHERE fid = '" + fid + "' ";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

{

dtc.GetConnection().Open();

}

SQLcmd = new SqlCommand(querry1, dtc.GetConnection());

SqlDataReader reader1 = SQLcmd.ExecuteReader();

while (reader1.Read())

{

WorkerList addWorker = new WorkerList();

addWorker.Id = reader1.GetInt32(0);

addWorker.Fullname = reader1.GetString(1);

DateTime date = reader1.GetDateTime(2);

addWorker.Birth = date.Date;

addWorker.Salary = reader1.GetInt32(3);

addWorker.Images = reader1.GetString(4);

addWorker.Fid = reader1.GetString(5);

listWorker.Add(addWorker);

}

reader1.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

for (int i = 0; i < listWorker.Count(); i++)

{

try

{

List<int> shifts = new List<int>();

for (int j = 0; j < DayOfMonth; j++)

{

string querryAttendance = "SELECT COUNT(shift\_worked) FROM Attendance " +

"WHERE id\_faculty = @fid " +

"AND YEAR(d\_m) = @year " +

"AND MONTH(d\_m) = @month " +

"AND DAY(d\_m) = @day " +

"AND id\_worker = @id ";

SQLcmd = new SqlCommand(querryAttendance, dtc.GetConnection());

SQLcmd.Parameters.AddWithValue("@fid", fid);

SQLcmd.Parameters.AddWithValue("@year", year);

SQLcmd.Parameters.AddWithValue("@month", month);

SQLcmd.Parameters.AddWithValue("@day", j + 1);

SQLcmd.Parameters.AddWithValue("@id", listWorker[i].Id);

int shiftAttendance = Convert.ToInt32(SQLcmd.ExecuteScalar());

string querryLate = "SELECT COUNT(shift\_worked) FROM LateList " +

"WHERE id\_faculty = @fid " +

"AND YEAR(d\_m) = @year " +

"AND MONTH(d\_m) = @month " +

"AND DAY(d\_m) = @day " +

"AND id\_worker = @id ";

SQLcmd = new SqlCommand(querryLate, dtc.GetConnection());

SQLcmd.Parameters.AddWithValue("@fid", fid);

SQLcmd.Parameters.AddWithValue("@year", year);

SQLcmd.Parameters.AddWithValue("@month", month);

SQLcmd.Parameters.AddWithValue("@day", j + 1);

SQLcmd.Parameters.AddWithValue("@id", listWorker[i].Id);

int shiftLate = Convert.ToInt32(SQLcmd.ExecuteScalar());

shifts.Add(shiftAttendance + shiftLate);

}

string querrySum = "SELECT COUNT(shift\_worked) FROM Attendance " +

"WHERE id\_faculty = @fid " +

"AND YEAR(d\_m) = @year " +

"AND MONTH(d\_m) = @month " +

"AND id\_worker = @id ";

SQLcmd = new SqlCommand(querrySum, dtc.GetConnection());

SQLcmd.Parameters.AddWithValue("@fid", fid);

SQLcmd.Parameters.AddWithValue("@year", year);

SQLcmd.Parameters.AddWithValue("@month", month);

SQLcmd.Parameters.AddWithValue("@id", listWorker[i].Id);

int sum = Convert.ToInt32(SQLcmd.ExecuteScalar());

string querrySumLate = "SELECT COUNT(shift\_worked) FROM LateList " +

"WHERE id\_faculty = @fid " +

"AND YEAR(d\_m) = @year " +

"AND MONTH(d\_m) = @month " +

"AND id\_worker = @id ";

SQLcmd = new SqlCommand(querrySumLate, dtc.GetConnection());

SQLcmd.Parameters.AddWithValue("@fid", fid);

SQLcmd.Parameters.AddWithValue("@year", year);

SQLcmd.Parameters.AddWithValue("@month", month);

SQLcmd.Parameters.AddWithValue("@id", listWorker[i].Id);

int sumLate = Convert.ToInt32(SQLcmd.ExecuteScalar());

int salaryMonth = (sum + sumLate) \* listWorker[i].Salary;

salaryMonth = salaryMonth - (sumLate \* 25000);

Worker worker = new Worker(listWorker[i].Id, listWorker[i].Fullname, shifts, sum + sumLate, sumLate, salaryMonth);

source.Add(worker);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

WorkersDataGrid.ItemsSource = source;

}

}

Đoạn mã trên được dùng để lấy và hiển thị dữ liệu bảng chấm công của nhân viên trong phòng ban. Dữ liệu đầu vào là tháng và năm. Sau khi lấy được số ngày trong tháng năm đó, add thêm các cột cần thiết vào bảng DataGrid với phương thức Columns.Add(). Lấy danh sách nhân viên trong bảng WorkerList. Với mỗi nhân viên, sử dụng vòng for với i chạy từ 0 đến số ngày trong tháng, trong đó ở mỗi ngày sử dụng SqlCommand để thực hiện câu lệnh đếm số ca điểm danh trong ngày sql bằng ExecuteScalar() trả về 1 giá trị đầu ra. Ở đây sử dụng 2 lệnh SQL để đếm số ca đi làm trong cả 2 bảng Attendance và LateList. Đối với cột còn lại là tổng số ca đi làm, tổng số ca đi muộn thì tương tự đếm số ca đi làm trong bảng Attendance và LateList với điều kiện dữ liệu trùng với tháng và năm mà actor đã nhập. Với cột lương thì lương cơ bản của nhân viên theo tháng được tính bằng công thức:

Lương cơ bản tháng = Tổng số ca đi làm \* hệ số lương theo ca – (số ca đi muộn \*25000).

Với những trường hợp đi muộn sẽ bị trừ 25000 đ với mỗi buổi, còn những buổi không đi làm sẽ không được tính lương.

Bắt lỗi nếu có lỗi trong khi truy vấn dữ liệu bị lỗi và hiển thị lên màn hình.

**Màn hình Late\_Manage:**

+ Giao diện: Hiển thị các button với các chức năng tương ứng để actor tương tác, hiển thị các ô nhập dữ liệu, với đầu vào gồm tên nhân viên, tháng, năm hoặc là chọn ngày, chương trình sẽ hiển thị danh sách nhân viên đi muộn thỏa mãn điều kiện.

+ Mã lệnh:

*Hàm tìm kiếm theo nhân viên, tháng và năm:*

private void Search\_Button\_Click(object sender, RoutedEventArgs e)

{

if (Search\_TextBox.Text.Trim() != "" &&

Month\_ComboBox.SelectedItem != null &&

Year\_TextBox.Text.Trim() != "")

{

string name = Search\_TextBox.Text.Trim();

int month = Convert.ToInt32(Month\_ComboBox.SelectedValue);

int year = Int32.Parse(Year\_TextBox.Text.Trim());

try

{

Late\_DataGrid.Items.Clear();

string querry = "SELECT LateList.id, LateList.id\_worker, " +

"fullname, d\_m, shift\_worked, detail " +

"FROM LateList INNER JOIN WorkerList ON " +

"LateList.id\_worker = WorkerList.id " +

"AND id\_faculty = @faculty " +

"WHERE fullname = @fullname " +

"AND MONTH(d\_m) = @month " +

"AND YEAR(d\_m) = @year ";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

{

dtc.GetConnection().Open();

}

cmd = new SqlCommand(querry, dtc.GetConnection());

cmd.Parameters.AddWithValue("@faculty", this.faculty);

cmd.Parameters.AddWithValue("@fullname", name);

cmd.Parameters.AddWithValue("@month", month);

cmd.Parameters.AddWithValue("@year", year);

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

int id = reader.GetInt32(0);

int id\_worker = reader.GetInt32(1);

string name\_worker = reader.GetString(2);

DateTime date = reader.GetDateTime(3);

int shift = reader.GetInt32(4);

string detail = reader.GetString(5);

Late\_DataGrid.Items.Add(new { Id = id, IdWorker = id\_worker, NameWorker = name\_worker, DateTime = date.ToString(), Shift = shift, Detail = detail });

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error");

}

}

else MessageBox.Show("Please enter full date in text box", "Message");

}

Đoạn mã trên được dùng để tìm kiếm danh sách nhân viên đi muộn với đầu vào là tên nhân viên, tháng và năm. Sử dụng SqlCommand để thực thi câu lệnh SQL với điều kiện trùng với dữ liệu đầu vào. Sau đó sử dụng SqlDataReader để đọc dữ liệu đầu ra và hiển thị lên DataGrid với phương thức Items.Add(). Bắt lỗi nếu có lỗi trong khi truy vấn dữ liệu bị lỗi và hiển thị lên màn hình, và nếu khi chưa nhập đủ thông tin đầu vào thì báo lỗi nhập đủ thông tin.

*Hàm xóa dữ liệu nhân viên đi muộn:*

private void Delete\_Button\_Click(object sender, RoutedEventArgs e)

{

if (Late\_DataGrid.SelectedItems.Count == 1)

{

MessageBoxResult result = MessageBox.Show("Delete this data from database ?", "Warning", MessageBoxButton.OKCancel);

if (result == MessageBoxResult.OK)

{

dynamic temp = Late\_DataGrid.SelectedItem;

if (temp != null)

{

dynamic Idselected = temp.Id;

string querry = "DELETE FROM LateList WHERE id = '" + Idselected + "'";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

{

dtc.GetConnection().Open();

}

cmd = new SqlCommand(querry, dtc.GetConnection());

cmd.ExecuteNonQuery();

MessageBox.Show("Successful", "Message");

}

}

}

else

{

MessageBox.Show("Please choose one to edit", "Message");

}

}

Đoạn mã trên được dùng để xóa dữ liệu nhân viên đi muộn. Với đầu vào là dữ liệu được chọn ở trong DataGrid, sử dụng dynamic để lấy được thuộc tính Id của thành phần đã chọn. Sử dụng SqlCommand để thực thi câu lệnh SQL xóa dữ liệu trong bảng LateList với điều kiện trùng id vừa lấy. Đưa ra thông báo xóa thành công. Bắt lỗi nếu có lỗi trong khi truy vấn dữ liệu bị lỗi và hiển thị lên màn hình, và nếu khi chưa chọn dữ liệu trong DataGrid thì sẽ đưa ra yêu cầu chọn.

**Màn hình Edit\_Late:**

+ Giao diện: Hiển thị các button với các chức năng tương ứng để actor tương tác, hiển thị các ô nhập dữ liệu, với đầu vào gồm id của dữ liệu trong LateList, tên nhân viên, ngày đi muộn, ca đi muộn và lý do đi muộn. Chương trình sẽ tìm kiếm id đó rồi cập nhật lại dữ liệu giống với dữ liệu đầu vào

+ Mã lệnh:

*Hàm lấy dữ liệu:*

public void setValueLate(int id)

{

try

{

string querry = "SELECT LateList.id\_worker, d\_m, shift\_worked, detail " +

"FROM LateList INNER JOIN WorkerList " +

"ON LateList.id\_worker = WorkerList.id " +

"AND LateList.id = '" + id + "' ";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

{

dtc.GetConnection().Open();

}

cmd = new SqlCommand(querry, dtc.GetConnection());

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

NameWorker\_ComboBox.SelectedValue = reader.GetInt32(0);

Date\_DatePicker.SelectedDate = reader.GetDateTime(1).Date;

Shift\_ComboBox.SelectedValue = reader.GetInt32(2);

Detail\_TextBox.Text = reader.GetString(3);

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

Đoạn mã trên được dùng để lấy dữ liệu đã có sẵn trên CSDL với id LateList được truyền vào từ màn hình LateManage khi bấm nút Edit. Thực thi câu lệnh SQL với dữ liệu đầu ra là id nhân viên, ngày đi muộn, ca và lý do đi muộn trong bảng LateList. Trên giao diện sử dụng ComboBox để chọn nhân viên, ComboBox sử dụng cách lấy dữ liệu tương tự, rồi set Content của thành phần là tên nhân viên, Tag của thành phần là id của nhân viên

string querry = "SELECT fullname,id FROM WorkerList WHERE fid = '" + fid + "' ";

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

{

dtc.GetConnection().Open();

}

cmd = new SqlCommand(querry, dtc.GetConnection());

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

ComboBoxItem item = new ComboBoxItem();

item.Content = reader.GetString(0);

item.Tag = reader.GetInt32(1);

NameWorker\_ComboBox.Items.Add(item);

}reader.Close();

*Hàm cập nhật dữ liệu vào trong CSDL:*

private void Save\_Button\_Click(object sender, RoutedEventArgs e)

{

if (checkInput())

{

int id\_worker = Convert.ToInt32(NameWorker\_ComboBox.SelectedValue);

DateTime date = Convert.ToDateTime(Date\_DatePicker.SelectedDate.Value);

int shift = Convert.ToInt32(Shift\_ComboBox.SelectedValue);

string detail = Detail\_TextBox.Text.Trim();

if (!checkAttendanced(id\_worker, date, shift) && !checkExist(id\_worker, date, shift))

{

try

{

if (dtc.GetConnection().State == System.Data.ConnectionState.Closed)

{

dtc.GetConnection().Close();

}

string querry = "UPDATE LateList SET " +

"id\_worker = @id\_worker, " +

"id\_faculty = @id\_faculty, " +

"d\_m = @d\_m, " +

"shift\_worked = @shift\_worked, " +

"detail = @detail " +

"WHERE id = @id";

cmd = new SqlCommand(querry, dtc.GetConnection());

cmd.Parameters.AddWithValue("@id\_worker", id\_worker);

cmd.Parameters.AddWithValue("@id\_faculty", this.faculty);

cmd.Parameters.AddWithValue("@d\_m", date);

cmd.Parameters.AddWithValue("@shift\_worked", shift);

cmd.Parameters.AddWithValue("@detail", detail);

cmd.Parameters.AddWithValue("@id", this.idLate);

cmd.ExecuteNonQuery();

MessageBox.Show("Successful !", "Message");

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Error");

}

}

}

}

Sau khi kiểm tra điều kiện đầu vào thỏa mãn, kiểm tra nếu dữ liệu đó nhân viên chưa điểm danh (xác nhận là đi muộn), kiểm tra dữ liệu sau khi sửa chưa tồn tại trên CSDL. Sử dụng SqlCommand để thực hiện câu lệnh cập nhật dữ liệu với đầu vào là id nhân viên, id phòng ban, ngày đi muộn, ca đi muộn, lý do đi muộn; cập nhật dữ liệu với id là id được truyền từ LateManage. Sau đó thông báo thành công. Bắt lỗi nếu trong quá trình chạy có lỗi rồi hiển thị lỗi lên màn hình.