**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Capstone Project Document

The Roll System using Mobile Device

|  |  |
| --- | --- |
| **Group 1** | |
| **Group member** | Pham Huy Hoang – Team Leader – 60740  Nguyen Thanh Binh – Team Member - 60321  Nguyen Quoc Huy – Team Member - 60551  Do Minh Dat – Team Member - 60545 |
| **Supervisor** | Mr. Kieu Trong Khanh |
| **Ext. Supervisor** | N/A |
| **Capstone Project code** | RSM |

-Ho Chi Minh City, 09/2013-

*This page is intentionally left blank*

***ACKNOWLEDGEMENTS***

We wish to thank various people for their contribution to this project: Our teachers for their advices and participant in the final review, our friend for the valuable technical support.

Special thanks should be given to Mr.Kiều Trọng Khánh, our research supervisor for his professional guidance and the useful, constructive recommendations throughout the course of this project.

# Table of Contents

[Table of Contents 4](#_Toc369798746)

[List of Tables 7](#_Toc369798747)

[List of Figures 9](#_Toc369798748)

[Definitions, Acronyms, and Abbreviations 11](#_Toc369798749)

[A. Introduction 12](#_Toc369798750)

[1. Project Information 12](#_Toc369798751)

[2. Introduction 12](#_Toc369798752)

[3. Current Roll Call System 12](#_Toc369798753)

[4. Problem Definition 12](#_Toc369798754)

[5. Proposed Solution 13](#_Toc369798755)

[5.1 Feature functions 13](#_Toc369798756)

[5.2 Advantage and disadvantage 13](#_Toc369798757)

[6. Functional Requirement 13](#_Toc369798758)

[6.1 Create Account 14](#_Toc369798759)

[6.2 Instructor Management 14](#_Toc369798760)

[6.3 Class Management 14](#_Toc369798761)

[6.4 Student Management 14](#_Toc369798762)

[6.5 Course Management 14](#_Toc369798763)

[6.6 Roll Call Management 14](#_Toc369798764)

[6.7 Attendance Checking 14](#_Toc369798765)

[6.8 Stranger Alert 14](#_Toc369798766)

[6.9 Attendance Report 14](#_Toc369798767)

[7. Role and Responsibility 14](#_Toc369798768)

[B. Software Project Management Plan 16](#_Toc369798769)

[1. Problem Definition 16](#_Toc369798770)

[1.1 Name of this Capstone Project 16](#_Toc369798771)

[1.2 Problem Abstract 16](#_Toc369798772)

[1.3 Project Overview 16](#_Toc369798773)

[2. Project organization 18](#_Toc369798774)

[2.1 Software Process Model 18](#_Toc369798775)

[2.2 Roles and responsibilities 19](#_Toc369798776)

[2.3 Tools and Techniques 20](#_Toc369798777)

[3. Project Management Plan 21](#_Toc369798778)

[3.1 Iteration 21](#_Toc369798779)

[3.2 Iteration Detail 22](#_Toc369798780)

[3.3 All Meeting Minutes 26](#_Toc369798781)

[4. Coding Convention 26](#_Toc369798782)

[C. Software Requirement Specification 27](#_Toc369798783)

[1. User Requirement Specification 27](#_Toc369798784)

[1.1 Guest Requirement 27](#_Toc369798785)

[1.2 Instructor Requirement 27](#_Toc369798786)

[1.3 Staff Requirement 27](#_Toc369798787)

[1.4 Student Requirement 27](#_Toc369798788)

[1.5 Admin Requirement 27](#_Toc369798789)

[1.6 System Requirement 27](#_Toc369798790)

[2. System Requirement Specification 27](#_Toc369798791)

[2.1 External Interface Requirement 27](#_Toc369798792)

[2.2 System Overview Use Case 28](#_Toc369798793)

[2.3 List of Use Case 29](#_Toc369798794)

[3. Software System Attribute 100](#_Toc369798795)

[3.1 Usability 100](#_Toc369798796)

[3.2 Reliability 100](#_Toc369798797)

[3.3 Availability 100](#_Toc369798798)

[3.4 Security 100](#_Toc369798799)

[3.5 Maintainability 100](#_Toc369798800)

[3.6 Portability 100](#_Toc369798801)

[3.7 Performance 100](#_Toc369798802)

[4. ERD 101](#_Toc369798803)

[D. Software Desgin Description 102](#_Toc369798804)

[1. Design Overview 102](#_Toc369798805)

[2. System Architectural Design 103](#_Toc369798806)

[3. Component Diagram 104](#_Toc369798807)

[4. Detailed Description of Components 105](#_Toc369798808)

[4.1 Class Diagram 105](#_Toc369798809)

[4.2 Class Diagram Explanation 105](#_Toc369798810)

[4.3 Sequence Diagram 113](#_Toc369798811)

[5. Database Design 125](#_Toc369798812)

[5.1 Logical Diagram 125](#_Toc369798813)

[5.2 Data Dictionary 126](#_Toc369798814)

[6. Algorithms 132](#_Toc369798815)

[6.1 Face Detection 132](#_Toc369798816)

[6.2 Face Recognition 136](#_Toc369798817)

# List of Tables

[Table 1: Roles and Responsibility 15](#_Toc369798818)

[Table 2: Hardware Requirement for Server 18](#_Toc369798819)

[Table 3: Hardware Requirement for Mobile App 18](#_Toc369798820)

[Table 4: Roles and Responsibility Details 20](#_Toc369798821)

[Table 5: Iteration 22](#_Toc369798822)

[Table 6: Phase 1: Preliminary Investigation or Analysis 22](#_Toc369798823)

[Table 7: Phase 2: Face Detect & Recognize 23](#_Toc369798824)

[Table 8: Phase 3: Student Management 24](#_Toc369798825)

[Table 9: Phase 4: Roll Call Management 24](#_Toc369798826)

[Table 10: Phase 5: Web Service Implement 25](#_Toc369798827)

[Table 11: Phase 6: Attendance Checking 25](#_Toc369798828)

[Table 12: Phase 7: Attendance Report 26](#_Toc369798829)

[Table 13: <Admin> Add Image for Single Student 31](#_Toc369798830)

[Table 14: <Admin> Add Image for Multi Students 34](#_Toc369798831)

[Table 15: <Admin> Remove Student Images 36](#_Toc369798832)

[Table 16: <Admin> Create Account 38](#_Toc369798833)

[Table 17: <Admin> Active Account 39](#_Toc369798834)

[Table 18: <Admin> Inactive Account 41](#_Toc369798835)

[Table 19: <Admin> Config System 42](#_Toc369798836)

[Table 20: <System> Face Detection 44](#_Toc369798837)

[Table 21: <System> Recognize Student 45](#_Toc369798838)

[Table 22: <System> Auto Free Storage Space 46](#_Toc369798839)

[Table 23: <System> Auto Active Roll Call 47](#_Toc369798840)

[Table 24: <System> Auto Inactive Roll Call 49](#_Toc369798841)

[Table 25: <Staff>Add Roll Call 51](#_Toc369798842)

[Table 26: <Staff>Edit Roll Call 54](#_Toc369798843)

[Table 27: <Staff> Add Subject 55](#_Toc369798844)

[Table 28: <Staff> Edit Subject 57](#_Toc369798845)

[Table 29: <Staff> Active Subject 59](#_Toc369798846)

[Table 30: <Staff> Inactive Subject 60](#_Toc369798847)

[Table 31: <Staff> Add Student 62](#_Toc369798848)

[Table 32: <Staff> Edit Student 65](#_Toc369798849)

[Table 33: <Staff> Import Student List 66](#_Toc369798850)

[Table 34: <Staff>Add Instructor 68](#_Toc369798851)

[Table 35: <Staff> Edit Instructor 70](#_Toc369798852)

[Table 36: <Staff> Add Class 72](#_Toc369798853)

[Table 37<Staff> Edit Class 73](#_Toc369798854)

[Table 38: <Staff> Report Attendance by Class 75](#_Toc369798855)

[Table 39: <Staff> Report Attendance by Student 76](#_Toc369798856)

[Table 40: <Staff> Report Attendance by Block 78](#_Toc369798857)

[Table 41: <Staff> Report Attendance by Semester 79](#_Toc369798858)

[Table 42: <Instructor> Check Attendance by Taking Picture 83](#_Toc369798859)

[Table 43: <Instructor> Check Attendance Manually 86](#_Toc369798860)

[Table 44: <Instructor> View Roll Call List 88](#_Toc369798861)

[Table 45: <Instructor> Report Attendance by Class 90](#_Toc369798862)

[Table 46: <Student> CheckPresent Rate 92](#_Toc369798863)

[Table 47: <Guest> Login 96](#_Toc369798864)

[Table 48: < Authorized User> Change Password 98](#_Toc369798865)

[Table 49: <User> Logout 100](#_Toc369798866)

[Table 50: Entity Data Dictionary 126](#_Toc369798867)

[Table 51: Attribute Data Dictionary 131](#_Toc369798868)

# List of Figures

[Figure 1: Agile Development Model 19](#_Toc369798869)

[Figure 2: System Overview Use Case 28](#_Toc369798870)

[Figure 3:<Admin> Overview Use Case 29](#_Toc369798871)

[Figure 4: <Admin> Add Image for Single Student 29](#_Toc369798872)

[Figure 5: <Admin> Add Image for Multi Student 32](#_Toc369798873)

[Figure 6: <Admin> Remove Student Image 34](#_Toc369798874)

[Figure 7: <Admin> Create Account 36](#_Toc369798875)

[Figure 8: <Admin> Active Account 38](#_Toc369798876)

[Figure 9: <Admin> Inactive Account 39](#_Toc369798877)

[Figure 10: <Admin> Config System 41](#_Toc369798878)

[Figure 11: <System> Overview Use Case 42](#_Toc369798879)

[Figure 12: <System> Face Detection 43](#_Toc369798880)

[Figure 13: <System> Recognize Student 44](#_Toc369798881)

[Figure 14: <System> Auto Free Storage Space 46](#_Toc369798882)

[Figure 15: <System> Auto Active Roll Call 47](#_Toc369798883)

[Figure 16: <System> Auto Inactive Roll Call 48](#_Toc369798884)

[Figure 17: <Staff> Overview Use Case 49](#_Toc369798885)

[Figure 18: <Staff>Add Roll Call 49](#_Toc369798886)

[Figure 19: <Staff>Edit Roll Call 52](#_Toc369798887)

[Figure 20: <Staff> Add Subject 54](#_Toc369798888)

[Figure 21: <Staff> Edit Subject 56](#_Toc369798889)

[Figure 22: <Staff> Active Subject 57](#_Toc369798890)

[Figure 23: <Staff> Inactive Subject 59](#_Toc369798891)

[Figure 24: <Staff> Add Student 61](#_Toc369798892)

[Figure 25: <Staff> Edit Student 63](#_Toc369798893)

[Figure 26: <Staff> Import Student List 65](#_Toc369798894)

[Figure 27: <Staff>Add Instructor 67](#_Toc369798895)

[Figure 28: <Staff>Edit Instructor 68](#_Toc369798896)

[Figure 29: <Staff> Add Class 70](#_Toc369798897)

[Figure 30: <Staff> Edit Class 72](#_Toc369798898)

[Figure 31: <Staff> Report Attendance by Class 74](#_Toc369798899)

[Figure 32: <Staff> Report Attendance by Student 75](#_Toc369798900)

[Figure 33: <Staff> Report Attendance by Block 77](#_Toc369798901)

[Figure 34: <Staff> Report Attendance by Semester 78](#_Toc369798902)

[Figure 35: <Staff> Export Report 80](#_Toc369798903)

[Figure 36: <Instructor> Overview Use Case 81](#_Toc369798904)

[Figure 37: <Instructor>Capture and Show result attendance 81](#_Toc369798905)

[Figure 38: <Instructor> Check Attendance Manually 84](#_Toc369798906)

[Figure 39: <Instructor> View Roll Call List 87](#_Toc369798907)

[Figure 40: <Instructor> Report Attendance by Class 89](#_Toc369798908)

[Figure 41: <Student> Overview Use Case 90](#_Toc369798909)

[Figure 42: <Student>CheckPresent Rate 91](#_Toc369798910)

[Figure 43: <Guest> Overview Use Case 95](#_Toc369798911)

[Figure 44: <Guest> Login 95](#_Toc369798912)

[Figure 45: < Authorized User> Overview Use Case 97](#_Toc369798913)

[Figure 46: < Authorized User> Change Password 97](#_Toc369798914)

[Figure 47: < Authorized User> Logout 99](#_Toc369798915)

[Figure 48: ERD - Conceptual Diagram 101](#_Toc369798916)

[Figure 49: MVC Architecture 103](#_Toc369798917)

[Figure 50: Component Diagram 104](#_Toc369798918)

[Figure 51: Class Diagram 105](#_Toc369798919)

[Figure 52: Add Image for Singe Student Sequence Diagram 113](#_Toc369798920)

[Figure 53: Add Image for Many Students Sequence Diagram 113](#_Toc369798921)

[Figure 54: Delete Student Image Sequence Diagram 114](#_Toc369798922)

[Figure 55: Create Account Sequence Diagram 114](#_Toc369798923)

[Figure 56: Configure System Sequence Diagram 115](#_Toc369798924)

[Figure 57: Face Detection Sequence Diagram 115](#_Toc369798925)

[Figure 58: Face Recognition Sequence Diagram 116](#_Toc369798926)

[Figure 59: Auto Free Storage Space Sequence Diagram 116](#_Toc369798927)

[Figure 60: Auto Active Roll Call Sequence Diagram 117](#_Toc369798928)

[Figure 61: Add Roll Call Sequence Diagram 117](#_Toc369798929)

[Figure 62: Edit Roll Call Sequence Diagram 118](#_Toc369798930)

[Figure 63: Import Student List Sequence Diagram 118](#_Toc369798931)

[Figure 64: Export Report Sequence Diagram 119](#_Toc369798932)

[Figure 65: Add Student Sequence Diagram 119](#_Toc369798933)

[Figure 66: Add Class Sequence Diagram 120](#_Toc369798934)

[Figure 67:Edit Class Sequence Diagram 120](#_Toc369798935)

[Figure 68: Add Subject Sequence Diagram 121](#_Toc369798936)

[Figure 69: Edit Subject Sequence Diagram 121](#_Toc369798937)

[Figure 70: Take Attendance Auto Sequence Diagram 122](#_Toc369798938)

[Figure 71: Take Attendance Manual Sequence Diagram 122](#_Toc369798939)

[Figure 72: View Roll Call Detail Sequence Diagram 123](#_Toc369798940)

[Figure 73: Report Attendance by Class Sequence Diagram 123](#_Toc369798941)

[Figure 74: Check Present Rate Sequence Diagram 124](#_Toc369798942)

[Figure 75: Logical Diagram 125](#_Toc369798943)

[Figure 76: Example of Haar Future 132](#_Toc369798944)

[Figure 77: Apply Haar future to sub-window 133](#_Toc369798945)

[Figure 78: Classifier Cascade 133](#_Toc369798946)

[Figure 79: Viola-Jones method flowchart 135](#_Toc369798947)

[Figure 80: Face Recognition Process 136](#_Toc369798948)

[Figure 81: Face Recognition Algorithm Flowchart 138](#_Toc369798949)

# Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| RSM | Roll System Mobile |
| Face Detection | **Face detection** is a computer technology that determines the locations and sizes of human faces in arbitrary (digital) images. It detects facial features and ignores anything else, such as buildings, trees and bodies.  <http://en.wikipedia.org/wiki/Face_detection> |
| Face Recognition | Face recognition is the task of identifying an already detected object as a **KNOWN or UNKNOWN face**, and in more advanced cases, **telling EXACTLY WHO'S face it is**! |

# Introduction

## Project Information

* Project name: **The roll system using mobile device**
* Project Code: **RSM**
* Product Type: **Website, Phone Application**
* Start Date: **September 9th, 2013**
* End Date:

## Introduction

Roll system was known with HPLite32, SimplePass of HP fingerpint system; roll system with ID card using by almost corporation or company around the world; the system face identify by Uniqul – Finland publish on 7/15/2013 or LogonSmart by Asus. Today, the trend machines replace humans in the hard work or the work does not require high intelligence became popular. So roll system face recognition becomes ever more necessary. The number of students in a university as well as the number of employees in large corporations in Vietnam is increasing dramatically, which means that the system should have professional roll with accuracy high.

## Current Roll Call System

Below are some current roll call systems:

* By manual: This system is widely used in school, university. The instructor will call the name of each student, check the absent, then submit the result to log system.
* Using ID card: This system is usually used in corporations. Each employee has a card. The card will be read by a card reader to check the attendance of the employee.
* Using fingerprint: This system is currently used in FPT University. 15 minutes before and after a studying session, the student must show his fingerprint to a machine to take attendance.
* Using camera + face recognition: This system is just in experimental stage. Each classroom has a camera. At the beginning of the studying session, the instructor uses the computer, connect to the camera and take picture of classroom. The system will recognize the student in the picture, write to roll call log system.

## Problem Definition

Below are the advantage and disadvantage of current roll call systems:

* By manual

+ Advantage: Simple to implement, cheap.

+ Disadvantage: The roll call take 3-5 minutes, take effort of instructor, wrong roll call.

* Using ID card

+ Advantage: High accuracy, not take much effort to check attendance.

+ Disadvantage: High cost (ID Card, Card Reader).

* Using fingerprint

+ Advantage: Quickly. No effort must be made from instructor.

+ Disadvantage: Cost of fingerprint reading machine. Risk of machine error. The student checks the attendance but not go to class.

* Using camera + face recognition

+ Advantage: Quickly. The entire class picture can be stored as log.

+ Disadvantage: High cost (Camera cost). Face recognition not too accuracy.

## Proposed Solution

The system is intended for used in only school or university, where the pupils/students sit in a classroom (A small classroom, from 20-30 students). The system must to manage the course, teacher, check attendance…. In detail, the system will enable following function:

### **Feature functions**

* The admin can manage information about course, class, instructor, students.
* The system will provide a method for admin to upload the student’s images, select the students in the images to make training data.
* The system will make a roll call list (Contains: Course, Class, Time, Instructor, Student List) for each class, based on the input information.
* The system will provide a method to assign instructor and student to course. The system must check the availability of the instructor before assigning.
* The system must support a method to change instructor of a course (When instructor is sick or busy), change or add new study session for a course. The system will send a mail to notify the instructor about schedule changing.
* The system can output report about the absent rate of a course, or a student.
* The instructor/examiner can view info about: What course they are teaching. Roll Call and student lists of these classes.
* The instructor will use the mobile application. At the beginning of a studying session, a picture of entire class will be taken for checking attendance.
* Base on the student list, the system will recognize the students from the picture. The mobile app will notify attendance result to instructor.
* The instructor can, re-check attendance manually.
* The instructor can view his teaching calendar, this calendar can sync to Google Calendar
* Based on the response from instructor, the system will gather more data to become more accuracy.
* The student can view info about what course they’re studying.
* The student can view their own attendance of the course they participated in.

### Advantage and disadvantage

The advantage and disadvantage of the proposed solution:

* Advantage: Small cost or no cost (If the instructor has phone). Face Recognition accuracy is about 70~80%, will reach 100% with confirm from instructor. The attendance check is quickly (< 30 seconds). The taken picture can be stored as log.
* Disadvantage: Complex to implement. The accuracy can vary depend on: Noise, brightness, number of people, image solution… Need student’s picture to be used for recognition training.

## Functional Requirement

Function requirement of the system are listed as below:

### Create Account

* The account of instructors and students will be auto-created by the system, basing on the input from admin.
* The instructors and students can change their profile and password.

### Instructor Management

* The staff can add/edit/active/inactive instructor.

### Class Management

* The staff can add/edit/active/inactive class.

### Student Management

* The admin can add/edit/active/inactive student.
* Each student must has a training set, contains 8-20 images for face recognizing. The staff can upload images, select the faces of the students to add. The staff can also remove images from training set.

### Subject Management

* The staff can add/edit/active/inactive subject.

### Roll Call Management

* The staff can add/edit/active/inactive roll call.
* The staff must input info of a roll call: The course, the instructor who teaches the course, the student list, the time of the course (Begin, End Date).
* The staff can change schedule, change instructor of a study session in roll call. The staff can also create add new study session.
* When the schedule is changed, an email will be sent to instructor.
* The instructor can see what roll call they belong to.
* The instructor can change their roll call to another instructor, in case of sickness or busy.

### Calendar Syncing

* The instructor can see his teaching calendar: What class, subject he will teach at time.
* This calendar can sync with the instructor Google Calendar.

### Attendance Checking

* At the beginning of each studying session, the instructor takes a picture of entire class for attendance checking.
* The instructor can re-check attendance manually.
* The mobile app will notify the result to the instructor.
* In case of wrong recognition, the student face will be added to training set for higher accuracy.
* The instructor can view the attendance log of the course they are teaching.
* The student can view the attendance log of the course they are studying.

### Stranger Alert

* If stranger is detected in the classroom, the system will alert to teacher. A log file will also be written.
* In case of wrong recognition, the student face will be added to training set for higher accuracy.

### Attendance Report

* The system will make attendance report, which shows the rate of absence in a course, or the rate of absence of a student.
* The report is only available to staff and instructor

## Role and Responsibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Full Name** | **Role** | **Position** | **Contact** |
| 1 | Kiều Trọng Khánh | Project Manager | Instructor | khanhkt@fpt.edu.vn |
| 2 | Phạm Huy Hoàng | Developer | Team Leader | hoangphse60740@fpt.edu.vn |
| 3 | Nguyễn Thanh Bình | Developer | Team Member | [binhnt60321@fpt.edu.vn](mailto:binhnt60321@fpt.edu.vn) |
| 4 | Nguyễn Quốc Huy | Developer | Team Member | [huynq60551@fpt.edu.vn](mailto:huynq60551@fpt.edu.vn) |
| 5 | Đỗ Minh Đạt | Developer | Team Member | datdm60545@fpt.edu.vn |

Table 1: Roles and Responsibility

# Software Project Management Plan

## Problem Definition

### Name of this Capstone Project

The roll system using mobile device (RSM)

### Problem Abstract

Roll system was known with HPLite32, SimplePass of HP fingerpint system; roll system with ID card using by almost corporation or company around the world; the system face identify by Uniqul – Finland publish on 7/15/2013 or LogonSmart by Asus. Today, the trend machines replace humans in the hard work or the work does not require high intelligence became popular. So roll system face recognition becomes ever more necessary. The number of students in a university as well as the number of employees in large corporations in Vietnam is increasing dramatically, which means that the system should have professional roll with accuracy high.

### Project Overview

#### The Current System

Below are some current roll call systems:

* By manual: This system is widely used in school, university. The instructor will call the name of each student, check the absent, then submit the result to log system.

+ Advantage: Simple to implement, cheap.

+ Disadvantage: The roll call take 3-5 minutes, take effort of instructor, wrong roll call.

* Using ID card: This system is usually used in corporations. Each employee has a card. The card will be read by a card reader to check the attendance of the employee.

+ Advantage: High accuracy, not take much effort to check attendance.

+ Disadvantage: High cost (ID Card, Card Reader). Risk of ID card lending, missing.

* Using fingerprint: This system is currently used in FPT University. 15 minutes before and after a studying session, the student must show his fingerprint to a machine to take attendance.

+ Advantage: Quickly. No effort must be made from instructor.

Disadvantage: Cost of fingerprint reading machine. Risk of machine error. The student checks the attendance but not go to class.

* Using camera + face recognition: This system is just in experimental stage. Each classroom has a camera. At the beginning of the studying session, the instructor uses the computer, connect to the camera and take picture of classroom. The system will recognize the student in the picture; write to roll call log system.

+ Advantage: Quickly. The entire class picture can be stored as log.

+ Disadvantage: High cost (Camera cost). Face recognition not to accuracy.

#### The Proposed System

The system is intended for used in only school or university, where the pupils/students sit in a classroom. The system must to manage the course, teacher, check attendance…. In detail, the system will enable following function:

##### Web

* The admin can manage information about course, class, instructor, students. The system provide a method for staff to import student, class list from Excel file.
* The system will provide a method for admin to upload the student’s images, select the students in the images to make training data.
* The system will make a roll call list (Contains: Course, Class, Time, Instructor, Student List) for each class, based on the input information.
* The system will provide a method to assign instructor and student to course. The system must check the availability of the instructor before assigning.
* The system must support a method to change instructor of a course (When instructor is sick or busy).
* The system can output report about the attendance of a course, a student, a block or semester.
* The student can view info about what course they’re studying.
* The student can view their own attendance of the course they participated in.

##### Mobile

* The instructor can view info about: What course they are teaching. Roll Call and student lists of these classes.
* The instructor will use the mobile application to take a picture.
* The mobile shows the list of student present in class, notify the instructor if absent rate is high.
* The instructor can re-check attendance manually.
* The system will alert stranger.

#### Boundaries of the System

* The system is intended for using university, with small classroom, or for the examine room (The testing site will be FPT University).
* The maximum number of a classroom is 30 people. The classroom size is about: 6m x 8m
* The system is not intended for managing these aspect:

+ Managing the teaching calendar of instructor.

+ Managing instructor qualification, salary info.

+ Managing the testing, mark of student of each class.

* The language of the system is English.
* The complete product includes:

+ The website, for admin and students. Instructor can also use the website to change profile, view roll call info.

+ Mobile Application for instructor to check attendance.

+ All the process involved document.

#### Development Environment

##### Hardware requirements

**For server**

|  |  |  |
| --- | --- | --- |
| Windows | Minimum Requirements | Recommended |
| Internet Connection | Cable, Wifi (4 Mbps) | Cable, Wifi (8 Mbps) |
| Operating System | XP, Vista, 7, 8 | XP, Vista, 7, 8 |
| Computer Processor | Intel® Core 2 Duo | Intel® Core(TM) i5 CPU , M 460 @ 2.53GHz |
| Computer Memory | 1GB RAM | 3GB or more |

Table 2: Hardware Requirement for Server

**For Mobile Application**

|  |  |  |
| --- | --- | --- |
| Mobile | Minimum Requirements | Recommended |
| Internet Connection | Wifi (2Mbps) | Wifi (4Mbps) |
| Operating System | Android 4.0 or later version | Android 4.4 |
| Hardware | Touchscreen, Camera 2.0 MP or above | Touchscreen, Camera 4.0 MP or above |
| Memory | 512 MB or more | 1 GB or more |

Table 3: Hardware Requirement for Mobile App

##### Software requirements

* Microsoft Windows 7 Service Pack 1: operating system and platform for development.
* SQL Server 2008 Express:used to create and manage the database for system.
* StarUML: used to create models and diagrams
* Skype: used for communication and meeting
* Visual Studio 2010: used to implement website and web service.
* Eclipse Juno 4.4, Android SDK 22.0.5, ADT 22.0.5 & JDK 7u25: used to implement mobile application.
* Google Code & TortoiseSVN: used for source control.

## Project organization

### Software Process Model

Project is developed under agile model.

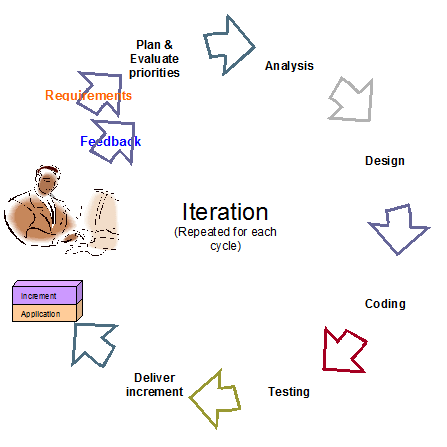


Figure 1: Agile Development Model

For more information: <http://www.indicthreads.com/1439/quick-introduction-to-agile-software-development/>

(Owner: IndicThreads.com. Online Software Developer Magazine and Conferences)

### Roles and responsibilities

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Full name** | **Role in Group** | **Responsibilities** |
| **1** | Kiều Trọng Khánh | Project manager | * Specify user requirement * Control the development process * Give out technique and business analysis support |
| **2** | Phạm Huy Hoàng | Team Leader, BA, DEV, Tester | * Managing process * Designing database * Clarifying requirements * Prepare documents * GUI Design * Create test plan * Coding * Testing |
| **3** | Nguyễn Thanh Bình | Team Member, BA, DEV, Tester | * Designing database * Clarifying requirements * Prepare documents * GUI Design * Create test plan * Coding * Testing |
| **4** | Nguyễn Quốc Huy | Team Member, BA, DEV, Tester | * Designing database * Clarifying requirements * Prepare documents * GUI Design * Create test plan * Coding * Testing |
| **5** | Đỗ Minh Đạt | Team Member, BA, DEV, Tester | * Designing database * Clarifying requirements * Prepare documents * GUI Design * Create test plan * Coding * Testing |

Table 4: Roles and Responsibility Details

### Tools and Techniques

- Front-end technologies: HTML5, CSS3, JavaScript, jQuery, AJAX.

- Back-end: Website: ASP.NET MVC3 + Entity Framework.

Web Service: WCF. Mobile App: Android - Java.

- Web Server: Microsoft IIS.

- Database Management System: MS SQL Server 2008 Express

## Project Management Plan

### Iteration

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phase**  **/Iteration** | **Description** | **Deliverables** | **Resource needed** | **Dependencies and Constrains** | **Risks** |
| **Preliminary Investigation or Analysis** | - Study similar existing systems.  -Identify and clarify requirements for the system in general. | -Introduction of proposed system.  -Main functions.  -Project Iteration Plan. | 30 man-days | N/A | Project may  not be feasible  for developing  because lack of technologies  and/or data |
| **Face Detect & Recognize** | -Studying face detect, face recognize algorithm & library.  -Find a solution, optimize for higher performance and accuracy. | -Face Detect and Recognize System (On Web Site and Web Service) | 25 man-days | N/A | Lack of experience.  The implemented algorithm is not the best.  Lack of test data |
| **Student management** | -Manage subject, student images/information. | -Student management function | 20 man-days | Depend on “**Face Detect & Recognize”** |  |
| **Roll call management** | -Create the roll call list, based on the information of student, instructor, class and course . | -Roll call management function | 30 man-days | N/A | Lack of experience.  Not have a clear understanding about business process. |
| **Web Service Implement** | -Create and deploy the web service on server. Provide RestFul service. | -Running WCF Web Service | 25 man-days | Depend on “**Roll call management”** | Lack of experience on making and deploying web service. |
| **Attendance checking** | -Instructor use mobile app to take picture, use picture for attendance checking. | -Android App with attendance checking function | 20 man-days | Depend on “**Web Service Implement”** | Lack of experience on Android development, transfer file between Android and WCF.  No Android device available for testing. |
| **Attendance report** | -Students can view their own attendance rate.  -Instructor and admin can view reports. | -Attendance report function. | 15 man-days | N/A | Lack of experience of making report. |

Table 5: Iteration

### Iteration Detail

#### Phase 1: Preliminary Investigation or Analysis

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| **1. Identifying and studying existing systems** | Find which systems currently provide similar service, their strengths and weakness. | HuyNQ, HoangPH, BinhNT |
| **2. Identifying and clarifying main functions.** | Define which main functions system should provide. | HuyNQ, HoangPH, BinhNT |
| **3. Introduction.** | Complete Introduction Report. | HoangPH |
| **4. Project Management**  **Plan.** | Prepare Project  Management Plan. | HoangPH |
| **5. Website Prototype.** | Build a prototype of proposed system (Website). | HuyNQ, HoangPH |
| **6. Mobile Prototype.** | Build a prototype of proposed system (Mobile App). | BinhNT, HoangPH |
| **7. Design ER diagram.** | Design ER diagram. | HoangPH, HuyNQ, BinhNT |

Table 6: Phase 1: Preliminary Investigation or Analysis

#### Phase 2: Face Detect & Recognize

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| **1. Identifying Requirement and Planning** | Which feature this function  should have and how to  implement. | HoangPH |
| **2. Studying Face Detection & Recognition Algorithm** | Studying algorithm, implement by using library EmguCV. | HoangPH |
| **3. Extract Face from Image** | Find the faces in images, extract them for later use | HoangPH |
| **4. Recognize Face** | From the input face, find out who the face belong to | HoangPH |
| **5. Optimize** | Optimize the implement for more performance and accuracy | HoangPH |
| **6. Implement GUI** | Create the interface for extracting and storing face | HoangPH |
| **7. Testing** | Test system behavior and  performance  Test user behavior and  performance | HoangPH, HuyNQ, BinhNT, DatDM |
| **8. Document** | Adding SRS, SDD,  Installation Guide, Manual  Guide | HoangPH, HuyNQ, BinhNT, DatDM |

Table 7: Phase 2: Face Detect & Recognize

#### Phase 3: Student Management

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| **1. Identifying Requirement and Planning** | Which feature this function  should have and how to  implement. | HoangPH, HuyNQ, BinhNT, DatDM |
| **2. Design ER Diagram** | Design ER Diagram | HoangPH, HuyNQ, BinhNT, DatDM |
| **3. Manage Student** | Allow staff to add/edit/active/inactive student  Allow staff to import student list from excel file. | HuyNQ |
| **4. Manage Student Face** | Each student has a training set, which contains 8-20 faces.  Allow admin to add, edit images in this training set. | HuyNQ, HoangPH |
| **5. Implement GUI** | Create the interface for managing student info | HuyNQ, HoangNQ |
| **6. Testing** | Test system behavior and  performance  Test user behavior and  performance | HoangPH, HuyNQ, BinhNT, DatDM |
| **7. Document** | Adding SRS, SDD,  Installation Guide, Manual  Guide | HoangPH, HuyNQ, BinhNT, DatDM |

Table 8: Phase 3: Student Management

#### Phase 4: Roll Call Management

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| **1. Identifying Requirement and Planning** | Which feature this function  should have and how to  implement. | HoangPH, HuyNQ, BinhNT, DatDM |
| **2. Manage Instructor** | Allow staff to add/edit/active/inactive instructor | DatDM |
| **3. Manage Course** | Allow admin to add/edit/active/inactive course | DatDM |
| **4. Manage Class** | Allow staff to add/edit/active/inactive course.  Assign student to class | HuyNQ |
| **5. Manage Roll Call** | Allow staff to create/edit/delete roll call.  Each roll call contain info: The instructor, the student list, the course, time, begin – end date | HuyNQ, HoangPH |
| **6. Manage Study Session** | Allow staff to add, change date, instructor of a study session in a roll call.  An email will be sent to instructor to notify the schedule changing | HoangPH, HuyNQ |
| **6. Teaching Calendar** | Based on the Roll Call List, a teaching calendar will be created for each instructor.  This calendar can be synced to Google Calendar | HoangPH, HuyNQ |
| **7. Implement GUI** | Create the interface for managing roll call | HuyNQ, BinhNT, HoangPH |
| **8. Testing** | Test system behavior and  performance  Test user behavior and  performance | HoangPH, HuyNQ, BinhNT, DatDM |
| **9. Document** | Adding SRS, SDD,  Installation Guide, Manual  Guide | HoangPH, HuyNQ, BinhNT, DatDM |

Table 9: Phase 4: Roll Call Management

#### Phase 5: Web Service Implement

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| **1. Identifying Requirement and Planning** | Which feature this function  should have and how to  implement. | HoangPH, HuyNQ, BinhNT, DatDM |
| **2. Studying WebService** | Studying the create and using of web serivce | HuyNQ |
| **3. Instructor Login** | Check the input id and password from mobile to login instructor | HuyNQ |
| **4. Get Instructor Info, Roll Call List** | Based on the instructor’s id, show to current roll call list | HuyNQ |
| **5. Face Recognize From Android Camera** | Studying about transferring image files between Android and webserivce  Transfer the result back to Mobile for showing | HuyNQ, BinhNT |
| **6. Testing** | Test system behavior and  performance  Test user behavior and  performance | HoangPH, HuyNQ, BinhNT, DatDM |
| **7. Document** | Adding SRS, SDD,  Installation Guide, Manual  Guide | HoangPH, HuyNQ, BinhNT, DatDM |

Table 10: Phase 5: Web Service Implement

#### Phase 6: Attendance Checking

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| **1. Identifying Requirement and Planning** | Which feature this function  should have and how to  implement. | HoangPH, HuyNQ, BinhNT, DatDM |
| **2. Getting Image From Mobile, sending to Web Service** | Allow instructor to take picture of class, use picture for checking attendance | BinhNT |
| **3. Confirm result, re-check attendance manually** | Allow instructor to confirm and re-check attendance | BinhNT |
| **4. Stranger Alert** | Alert when detect stranger | BinhNT |
| **5. Implement Mobile App** | Create the Android App for instructor to User | BinhNT, HuyNQ, HoangNQ |
| **6. Testing** | Test system behavior and  performance  Test user behavior and  performance | HoangPH, HuyNQ, BinhNT, DatDM |
| **7. Document** | Adding SRS, SDD,  Installation Guide, Manual  Guide | HoangPH, HuyNQ, BinhNT, DatDM |

Table 11: Phase 6: Attendance Checking

#### Phase 7: Attendance Report

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| **1. Identifying Requirement and Planning.** | Which feature this function  should have and how to  implement. | HoangPH, HuyNQ, BinhNT, DatDM |
| **2. Report attendance of a course** | System will make report about the attendance of a course | HoangPH |
| **3. Report attendance of a student** | System will make report about the attendance of a student | HoangPH |
| **4. Report attendance of a block** | System will make report about the attendance of all class in a block | HuyNQ |
| **5. Report attendance of a semester** | System will make report about the attendance of all class in a semester | HuyNQ |
| **6. Export Report** | Admin can export report to Excel file | HuyNQ |
| **7. Testing** | Test system behavior and  performance  Test user behavior and  performance | HoangPH, HuyNQ, BinhNT, DatDM |
| **8. Document** | Adding SRS, SDD,  Installation Guide, Manual  Guide | HoangPH, HuyNQ, BinhNT, DatDM |

Table 12: Phase 7: Attendance Report

### All Meeting Minutes

Refer to Meeting Minutes folder.

## Coding Convention

Java: Using to develop Android App.

Summary:

* Naming Convention.
* Indentation.
* Declaration.
* Code Examples

Follow “Code Conventions for the Java TM Programming Language, by Sun Microsystems, rev April 20, 1999”.

C#: Using to develop website and web service.

Summary:

* Naming Convention.
* Layout Convention.
* Commenting Convention.
* Language Guidelines

Using C# Code Convention From:

<http://msdn.microsoft.com/en-us/library/vstudio/ff926074.aspx>

# Software Requirement Specification

## User Requirement Specification

### Guest Requirement

Guest is a person who doesn’t have access to the system. To use system functions, guest must Login.

### Instructor Requirement

Instructor is teacher of the university. Instructor’s account is created by admin or auto created bt system. Instructor can use mobile app or website, that can do functions:

* Check Attendance
* Report Attendance Rate

### Staff Requirement

Staff is the employee of the university office. Staff’s account is created by admin. Staff can use website with these functions:

* Manage Subject: Add/ Edit/Active/ Inactive Subject
* Manage Class: Add/Edit /Active/ Inactive Class
* Manage Teacher: Add/Edit Teacher
* Manage Student: Add/Edit Student. Import Student List
* Manage Roll Call: Add/Edit Roll Call
* Report Attendance Rate: Report by Class, Block, Student. Export report

### Student Requirement

Student is the student of the university. Student’s account is created by admin. . Staff can use website with function:

* Check Present Rate

### Admin Requirement

Admin is the one who maintained and config the system. Admin can do functions:

* Manage Account: Create/Edit/Active/Inactive Account
* Manage Student Image: Add/Delete Student’s Image
* Config System

### System Requirement

System is also an actor, run in the background to keep the system working. System can do functions:

* Auto free storage space
* Face Detection
* Face Recognition: Recognition Student/Store Result
* Manage Roll Call: Auto Active/Inactive Roll Call

## System Requirement Specification

### External Interface Requirement

#### User Interface

* + - The interface of website is clear, do not annoy customer.
    - The interface of mobile app must be clear, compatibale with touch screen. The size of controls must be big enough to touch on smartphone.
    - The error, warning messages must be make clear, easy to understand. Error warning does not disturb customer.

#### Hardware Interface

* The system will use the standard hardware and data communications resources of a standard computer.

#### Software Interface

* Firefox Browser, Chromes with Resolution (1280\*800) or bigger and support JavaScript and HTML5
* Smartphone with Android 4.1 or above. Screen size (70 x 120 mm) or bigger.

#### Communication Protocol

* Website using HTTP protocol for communication between the web browser and the web server.
* Mobile app using HTTP protocol for communicating between app and web service.

### System Overview Use Case

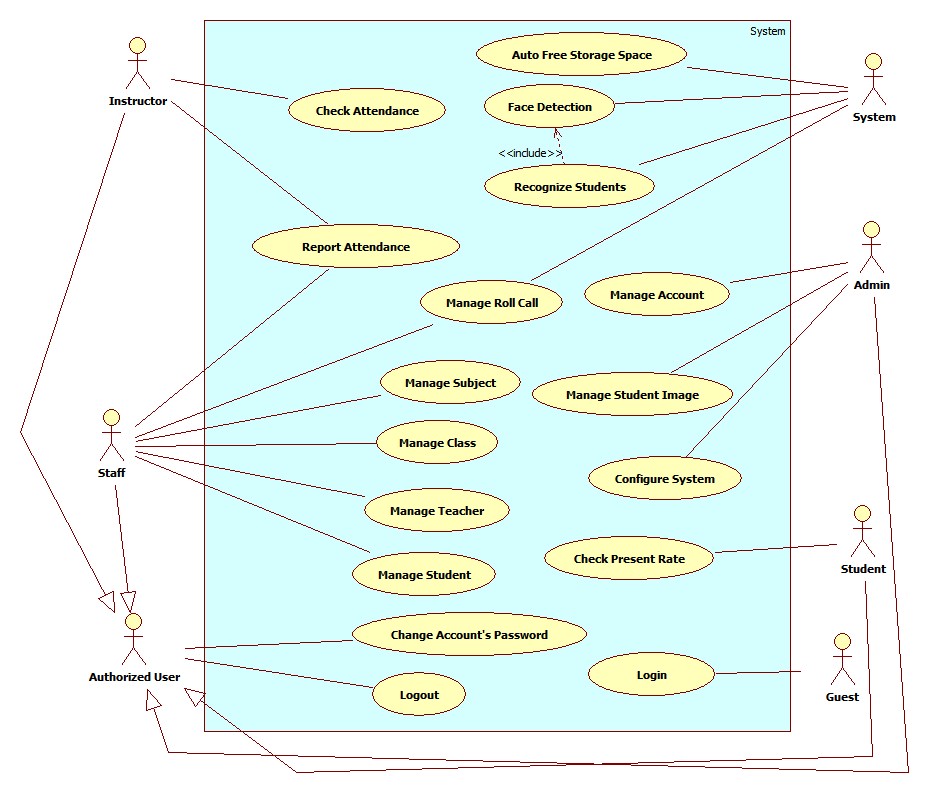


Figure 2: System Overview Use Case

### List of Use Case

#### <Admin>Overview Use Case

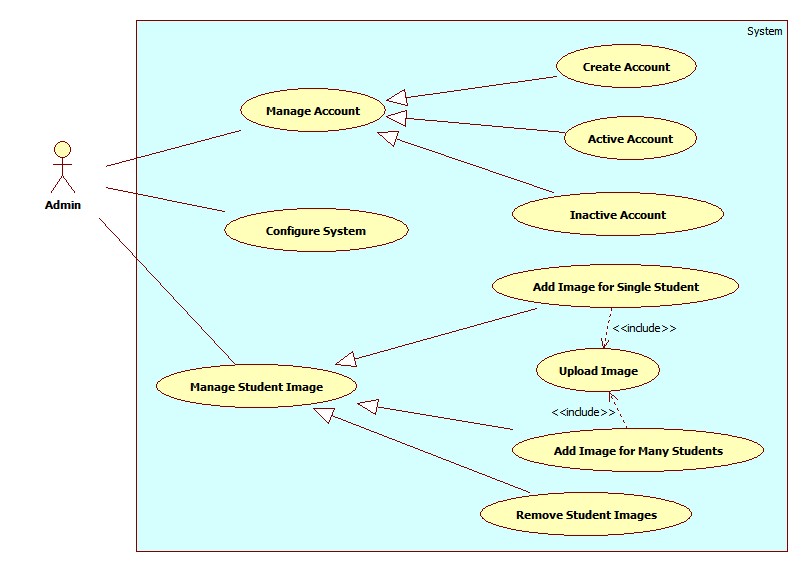


Figure 3:<Admin> Overview Use Case

##### <Admin> Add Image for Single Student

Use Case Diagram

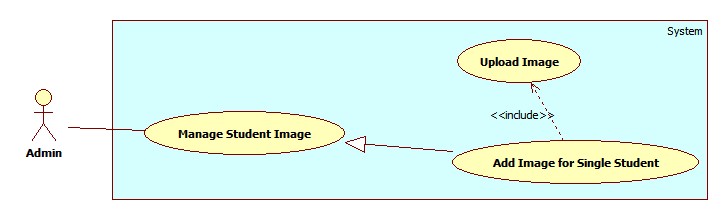


Figure 4: <Admin> Add Image for Single Student

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – AD005** | | | |
| **Use Case No.** | AD005 | **Use Case Version** | 2.0 |
| **Use Case Name** | Add Image for Single Student | | |
| **Author** | Pham Huy Hoang | | |
| **Date** | 20/09/2013 | **Priority** | Normal |
| **Actor:**Admin  **Summary:**  This user case allow admin add image data for a student.  **Goal:**  Image of a students will be stored on server, the links to images is stored in database  **Triggers:**   * For face recognition, our system must store images of students face. With more images, we can have better recognition accuracy. * To add new image for a single student, admin must: * Click menu item“Manage Student Image” -> “Student List”. * On the Student List Page, click Detail to view image data of a student. * Upload images. System will detect human faces from uploaded images. * Select the student faces from images. * Press “Save” to save the images.   **Preconditions:**   * User must logged in the system with the role is admin. * There must be at lease one student in the student list. * The admin has images contain the student’s face.   **Post Conditions:**   * **Success:** The images of a student will be stored on server and database * **Fail:**No image added   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | - Click menu item“Manage Student Image”  - Click choose “Student List” menu.  [Alternative 1] | Redirect to Student List Page, contains :   * Table with 4 columns, record of 30 lastest students: * Student Code: label * Name: label * Email: label * Image Data: “Detail” button * Class: Dropdownlist, data source: All active class in database. | | 2 | Click on “Detail” button. | Redirect to Student Detail Page, contains detail of Student Image Data, include:   * Student Current Image: A list of images * File Upload Control * Upload: button | | 2 | Select the student’s images from computer. | The selected images will be displayed as thumbnails.  [Exception 1] | | 3 | Click “Upload” button. | System will detect student faces from uploaded images, then display image full size with white **face regions** (See Use Case SY002)  [Exception 2, 3] | | 4 | Select student’s face from face region.  Click “Save” button. | The selected face region will be cropped from original picture, stored into server. Face’s owner and link will be stored in database. [Exception 4]  The pop-up is closed. The Student List Page is auto refresh. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | [Alternative 1]  - Select a class from Class drop down list.  - Press “Select” button. | A table show the records of all students in selected class.  Table with 4 columns:   * Student Code: label * Name: label * Email: label * Image Data: “Detail” button |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Select more than 20 images | Display error message: “ Too much images. Only process first 20 images” | | 2 | Not select image, click “Upload” button | Display error message: “Please choose images for uploading” | | 3 | - Upload images don’t contain human faces.  Or  -No faces detected from uploaded images | Display error message: “ No face found. Select another images” | | 4 | The selected face already in the database | Display message: “Face XXX existed in database” |   **Relationships:** Manage Student Image  **Business Rules:**   * For saving storage space, only student’s face is stored. The face is saved as jpeg, resolution 100 x 100. * The recommend number of image for a student is larger than 10 for acceptable accuracy. * The original uploaded images willbe stored on Temp folder on server. This Temp folder will be emptied at time interval (See Use Case SY001). | | | |

Table 13: <Admin> Add Image for Single Student

##### <Admin> Add Image for Multi Student

Use Case Diagram

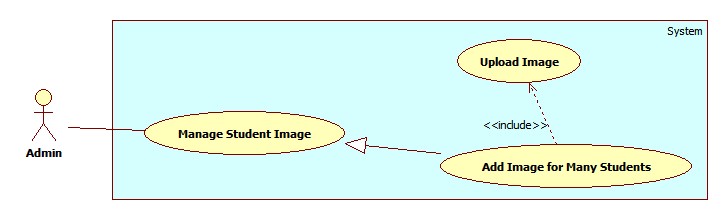


Figure 5: <Admin> Add Image for Multi Student

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – AD006** | | | |
| **Use Case No.** | AD006 | **Use Case Version** | 2.0 |
| **Use Case Name** | Add Image for Multi Student | | |
| **Author** | Pham Huy Hoang | | |
| **Date** | 20/09/2013 | **Priority** | Normal |
| **Actor:**Admin  **Summary:**  This user case allow admin to add image data for many students.  **Goal:**  Image of students will be stored on server, the links to images is stored in database  **Triggers:**   * For face recognition, our system must store images of students face. With more images, we can have better recognition accuracy. * To add new image for a single student, admin must: * Click menu item“Manage Student Image” -> “Add Image for Students”. * Upload images. System will detect student faces from uploaded images. * Select the student faces from images. * Select the each face’s owner. * Press “Save” to save the images and students.   **Preconditions:**   * User must logged in the system with the role is admin. * There must be at lease one student in the student list. * The admin has images contain the student’s face.   **Post Conditions:**   * **Success:** The images of students will be stored on server and database * **Fail:**No image added   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | - Click menu item“Manage Student Image”  - Click choose “Add Image for Students”. | Redirect to Add Image for Students Page, contains:   * Upload Students Images: text label * File Upload Control * Upload: button | | 2 | Select the student’s images from computer | The selected images will be displayed as thumbnails.  [Exception 1] | | 3 | Click “Upload” button | System will detect student faces from uploaded images, then display image full size with white **face regions** (See Use Case SY002)  [Exception 2, 3] | | 4 | Select student’s face from face region, select the face owner  [Alternative 1] | When user hover into a face region, a suggestion textbox is displayed.  (This suggestion textbox help admin to choose the face’s owner.)  The textbox data source is all students in database. A suggestion record includes:   * Student Face: image * Student ID: label * Full name: label * Class: label | | 5 | Click “Save” button. | The selected face region will be cropped from original picture, stored into server. Face’s owner and link will be stored in database. [Exception 4] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Admin doesn’t choose the face’s owner. | The suggestion textbox is cleared. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Select more than 20 images | Display error message: “ Too much images. Only process first 20 images” | | 2 | Not select image, click “Upload” button | Display error message: “Please choose images for uploading” | | 3 | Upload images don’t contain human faces.  No faces detected from uploaded images | Display error message: “ No face found. Select another images” | | 4 | The select face already in the database | Display message: “Face XXX existed in database” |   **Relationships:** Manage Student Image  **Business Rules:**   * The same as Use Case AD005. * The auto-complete textbox must show the student’s face, name, ID and class, as the same as facebook. | | | |

Table 14: <Admin> Add Image for Multi Students

##### <Admin> Remove Student Image

Use Case Diagram

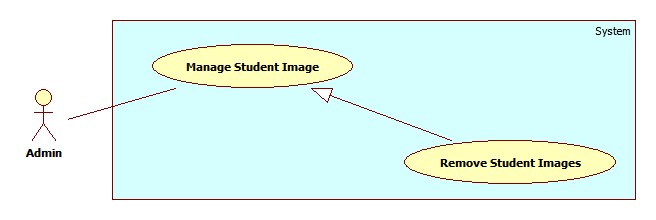


Figure 6: <Admin> Remove Student Image

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – AD007** | | | |
| **Use Case No.** | AD007 | **Use Case Version** | 2.0 |
| **Use Case Name** | Remove Student Image | | |
| **Author** | Pham Huy Hoang | | |
| **Date** | 20/09/2013 | **Priority** | Normal |
| **Actor:**Admin  **Summary:**  This user case allow admin to remove image of a student.  **Goal:**  Remove an/many images from a student Image Data  **Triggers:**   * For face recognition, our system must store images of students face. Sometimes, the admin can add the wrong face of students, which lower the accuracy. So the admin must be able to remove image of a student. * To remove image of a single student, admin must: * Click menu item“Manage Student Image” -> “Student List”. * On the Student List Page, click Detail to view image data of a student. * Click on the “X” button at the top-right corner of the image. * Click “Ok”   **Preconditions:**   * User must logged in the system with the role is admin. * There must be at lease one student in the student list. This student must has at least 1 face image.   **Post Conditions:**   * **Success:** The image of a student will be removed from server and database * **Fail:**No image removed   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click menu item“Manage Student Image” -> “Student List”.  [Alternative 1] | Redirect to Student List Page, contains :   * Table with 4 columns, record of 30 lastest students: * Student Code: label * Name: label * Email: label * Image Data: “Detail” button   Class: Drop down list, data source: All active class in database. | | 2 | Click on “Detail” button | Redirect to Student Detail Page, contains detail of Student Image Data, include:   * Student Current Image: A list of images * File Upload Control * Upload: button | | 3 | Click on the “X” button at the top-right corner of the image | Show confirms windows with message:  “Are you sure to delete this image?” | | 4 | Click “Ok” button.  [Alternative 2] | The selected image will be removed from server and database.  An message is display: “Image XXX removed”. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | [Alternative 1]  - Select a class from Class drop down list.  - Press “Select” button. | A table show the records of all students in selected class.  Table with 4 columns:   * Student Code: label * Name: label * Email: label * Image Data: “Detail” button | | 3 | [Alternative 2]  Click “Cancel” | The image is not removed. |   **Exceptions:** N/A  **Relationships:** Manage Student Image  **Business Rules:** N/A | | | |

Table 15: <Admin> Remove Student Images

##### <Admin> Create Account

Use Case Diagram

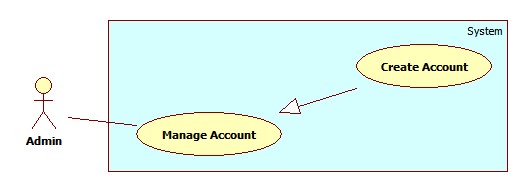


Figure 7: <Admin> Create Account

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – AD001** | | | |
| **Use Case No.** | AD001 | **Use Case Version** | 2.0 |
| **Use Case Name** | Create Account | | |
| **Author** | Pham Huy Hoang | | |
| **Date** | 20/09/2013 | **Priority** | Normal |
| **Actor:**Admin  **Summary:**  This user case allow admin to create an account for login the system  **Goal:**  A new account (For student, instructor or staff) will be added to database  **Triggers:**   * In our system, the account of a student or instructor is auto created when staff create new instructor/new student. This function is used to create staff, student or instructor account, in case of backup. * To create a new account, admin must: * From the admin Home Page, click the menu item “Account Manager”. * In the Account Manager Page, click on the Tab “Create Account”. * Input new account information, click “Create” to finish.   **Preconditions:**   * User must logged in the system with the role is admin   **Post Conditions:**   * **Success:** A new account will be created. The acocunt will be displayed on account list. * **Fail:**No acocunt created.Transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click menu item “Account Manager”. | Redirect to Account Manager Page, includes a panel with 2 tabs:   * Account List * Create Account   Current tab is Account List. This tab includes:   * Table with 5 column: * Account ID: label * Username: label * Email: label * Role: label * Select: checkbox * Action: dropdownlist, values: Inactive, Active. * Process: button | | 2 | Click Tab “Create Account”. | Display a tab panel includes:   * Username: textbox (min length, max length: 30, required). * Password: textbox (password box, min length, max length: 30, required). * Password: textbox (password box, min length, max length: 30, required). * Roles : dropdownlist, values: Admin, Staff, Instructor, Student. * Create: button * Reset: button | | 3 | Input new account information.  Click “Create” button.  [Alternative 1] | System will add the account to database.  Show the tab panel “Account List”  [Exception 1,2,3,4] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Click on “Reset” button | The form is emptied. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Not enter enough require information. | Display error message: “Please enter all require information.” | | 2 | Username/password is not in range [6,30] | Display error message: “Username/Password must in range [6,30].” | | 3 | Confirm password is not equal password | Display error message:  “Password and confirm not match.” | | 4 | Username or email existed in database | Display error message:  “Username/Email already exist.” |   **Relationships:** Manage Account  **Business Rules:**   * Each account is active when added to database. * The “confirm password” value must match “password” value. | | | |

Table 16: <Admin> Create Account

##### <Admin> Active Account

Use Case Diagram

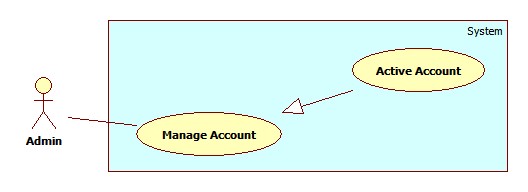


Figure 8: <Admin> Active Account

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – AD002** | | | |
| **Use Case No.** | AD002 | **Use Case Version** | 2.0 |
| **Use Case Name** | Active Account | | |
| **Author** | Pham Huy Hoang | | |
| **Date** | 20/09/2013 | **Priority** | Normal |
| **Actor:**Admin  **Summary:**  This user case allow admin to active an accout  **Goal:**  A inactive account will be activated  **Triggers:**   * In our system, when we do not want an user to login, we inactive his/her account (See Use Case – AD003). We can active the inactivated account so they can login again. * To active an account, admin must: * From the admin Home Page, click the menu item “Account Manager”. * Check the Select checkbox of the account record, select “Active” on the drop down list, and press “Process” button.   **Preconditions:**   * User must logged in the system with the role is admin. * There must be at least 1 inactive account in database.   **Post Conditions:**   * **Success:** The selected account(s) will be activated. * **Fail:**No acocunt activated.Transfer to error page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click menu item“Account Manager”. | Redirect to Account Manager Page, includes a panel with 2 tabs:   * Account List * Create Account   Current tab is Account List. This tab includes:   * Table with 5 column: * Account ID: label * Username: label * Email: label * Role: label * Select: checkbox * Action: dropdownlist, values: Inactive, Active. * Process: button | | 2 | Check the checkboxs of the account(s), select “Active” on the drop down list, and press “Process” button. | The selected account(s) will be activated  [Exception 1,2] |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | The account has already been activated by another admin | Nothing happen, the account(s) status is active | | 2 | Check no checkbox, press “Process” button | Display error message:  “Please select account to activated.” |   **Relationships:** Manage Account  **Business Rules:**   * Admin can’t active/inactive account of themselves or other admin. | | | |

Table 17: <Admin> Active Account

##### <Admin> Inactive Account

Use Case Diagram

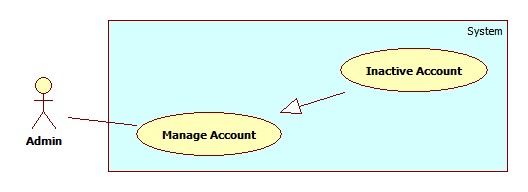


Figure 9: <Admin> Inactive Account

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – AD003** | | | |
| **Use Case No.** | AD003 | **Use Case Version** | 2.0 |
| **Use Case Name** | Inactive Account | | |
| **Author** | Pham Huy Hoang | | |
| **Date** | 20/09/2013 | **Priority** | Normal |
| **Actor:**Admin  **Summary:**  This user case allow admin to active an accout  **Goal:**  A active account will be inactivated  **Triggers:**   * In our system, when we do not want an user to login, we inactive his/her account. The account can be activated later. * To active an account, admin must: * From the admin Home Page, click the menu item “Account Manager”. * Check the Select checkbox of the account record, select “Active” on the drop down list, and press “Process” button.   **Preconditions:**   * User must logged in the system with the role is admin * There must be at least 1 active account in database.   **Post Conditions:**   * **Success:** The selected account(s) will be inactivated. * **Fail:**No acocunt inactivated.Transfer to error page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click menu item“Account Manager”. | Redirect to Account Manager Page, includes a panel with 2 tabs:   * Account List * Create Account   Current tab is Account List. This tab includes:   * Table with 5 column: * Account ID: label * Username: label * Email: label * Role: label * Select: checkbox * Action: dropdownlist, values: Inactive, Active. * Process: button | | 2 | Check the checkboxs of the account, select “Inactive” on the drop down list, and press “Process” button. | The selected account(s) will be inactivated.  [Exception 1,2] |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | The account has already been inactivated by another admin. | Nothing happen, the account(s) status is inactive | | 2 | Check no checkbox, press “Process” button. | Display error message:  “Please select account to inactivated.” |   **Relationships:** Manage Account  **Business Rules:**   * Admin can’t active/inactive account of themselves or other admin. | | | |

Table 18: <Admin> Inactive Account

##### <Admin> Config System

Use Case Diagram

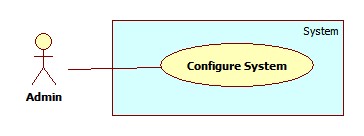


Figure 10: <Admin> Config System

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – AD004** | | | |
| **Use Case No.** | AD004 | **Use Case Version** | 2.0 |
| **Use Case Name** | Configure System | | |
| **Author** | Pham Huy Hoang | | |
| **Date** | 20/09/2013 | **Priority** | Normal |
| **Actor:**Admin  **Summary:**  This user case allow admin to configure system’s properties.  **Goal:**  Change current properties of the system  **Triggers:**   * For some reason, administrator needs to re-configure system’s properties. * To config the system, admin must: * Click menu item“Configuration”. * On the configuration page, edit properties and click on “Update” button corresponding with each properties in list.   **Preconditions:**   * User must logged in the system with the role is admin.   **Post Conditions:**   * **Success:** System’s properties is updated to new value. * **Fail:**Nothing changed.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click menu item “Configuration”. | Redirect to System Configuration Page, with list of system’s properties include:   * Name: label * Value: textbox * Update: button | | 2 | Editproperty value and click “Update” button. | Show confirms windows with message:  “Are you sure to change this property?” | | 3 | Click “Ok” to confirm.  [Alternative 1] | Update property value to new value. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Click “Cancel”. | Nothing happen, the value is unchanged. |   **Exceptions:** N/A  **Relationships:** N/A  **Business Rules:**   * Configure system’s properties will affect to all function using these properties.. | | | |

Table 19: <Admin> Config System

#### <System> Overview Use Case



Figure 11: <System> Overview Use Case

##### <System> Face Detection

Use Case Diagram

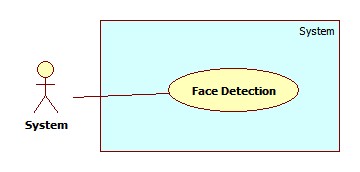


Figure 12: <System> Face Detection

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SY002** | | | |
| **Use Case No.** | SY002 | **Use Case Version** | 2.0 |
| **Use Case Name** | Face Detection | | |
| **Author** | Pham Huy Hoang | | |
| **Date** | 21/09/2013 | **Priority** | High |
| **Actor:**System.  **Summary:**  This use case is used to detect the face regions in an image after it is uploaded successful.  **Goal:**  Detect the face regions in an image  **Triggers:**   * When admin upload images for Manage Student Image (See Use Case AD005, AD006), or instructor send images for Checking Attendance (See Use Case IU001), system will detect the face regions in the upload images. * The detected region will be displayed at client, or used to recognize the students in the images.   **Preconditions:**   * The images are uploaded to server successful.   **Post Conditions:**   * **Success:** The face regions is detected. Send result to front end for displaying. * **Fail:**No face detected   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | - Admin upload image for Manage Student Image.  Or  - Instructor send image for Checking Attendance. | Detect the face in the images, create a list of face regions. A face region’s properties:   * Top position * Left position * Height * Width   The face region can be stored as Image Data for Students, or for Student Recognition (See Use Case SY003)  [Exception 1,2] |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | The upload image is corrupt | Show error message: “The image XXX is corupted”. | | 2 | Upload images don’t contain human faces.  No faces detected from uploaded images. | Display error message: “ No face found. Select another images” |   **Relationships:** N/A  **Business Rules:**   * The accuracy of face detection is larger than 80%. Picture with higher brightness and resolution will has higher accuracy. * If the uploaded images are too big, the image will be resized for faster processing. | | | |

Table 20: <System> Face Detection

##### <System> Recognize Student

Use Case Diagram

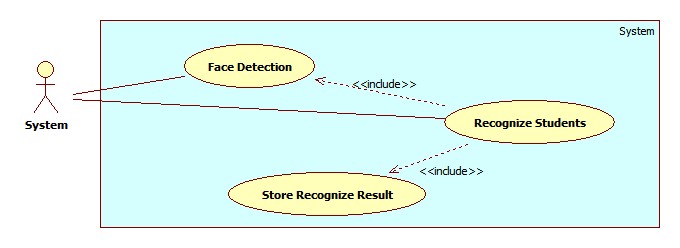


Figure 13: <System> Recognize Student

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SY003** | | | |
| **Use Case No.** | SY002 | **Use Case Version** | 2.0 |
| **Use Case Name** | Recognize Student | | |
| **Author** | Pham Huy Hoang | | |
| **Date** | 21/09/2013 | **Priority** | Normal |
| **Actor:**System.  **Summary:**  This use case is used to detect the face regions in an image (for checking attendance)  **Goal:**  Detect the face regions in an image  **Triggers:**   * When instructor send images for Checking Attendance (See Use Case IU001), system will detect the face regions in the upload images. After that, the system will recognize the student present in the images. * The recognized student list will be stored as attendance log in database.   **Preconditions:**   * The images are uploaded to server successful. * The Image Data of the students must existed in database (At least 10 images). If a student doesn’t have image in database, that student will be recognized as “Unknow”   **Post Conditions:**   * **Success:** The students are recognized from images. Recognize result will be stored. * **Fail:**No student recognized. No log stored.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Instructor send image for Checking Attendance. | The same as step 1 , Use Case SY002.   * Detect face regionsin the picture. * Create a face matching list. * Each face will be matched to find the face’s owner. * After recognizing, show the result and store the result to database.   [Exception 1,2,3] |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | The number of student detected is too small ( fewer than 5) | Show error message: “Only some students detected. Please check”. | | 2 | Find unknow people in the image | Alert: “Stranger Detected. Please check”. |   **Relationships:** Face Detection, Face Recogntion, Store Recognize Result  **Business Rules:**   * The accuracy of face recognition is 60~80%, depend on the number of image in training data. * The recommend number of image for a student is larger than10 for acceptable accuracy. * When stranger is detected, a log will be stored for later checking. | | | |

Table 21: <System> Recognize Student

##### <System> Auto Free Storage Space

Use Case Diagram

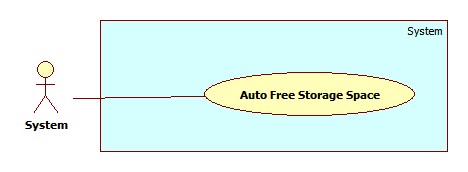


Figure 14: <System> Auto Free Storage Space

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SY001** | | | |
| **Use Case No.** | SY001 | **Use Case Version** | 2.0 |
| **Use Case Name** | Auto Free Storage Space | | |
| **Author** | Pham Huy Hoang | | |
| **Date** | 21/09/2013 | **Priority** | Normal |
| **Actor:**System.  **Summary:**  This use case is used to auto free space on server  **Goal:**  Delete the image in the Temp folder to free space on server.  **Triggers:**   * When admin upload images for Managed Student Images (See Use Case AD005), those images will be stored in Temp folder. * At 0.00am, or when the size of Temp folder is bigger than 500MB, the Temp folder will be emptied for free space.   **Preconditions:** N/A.  **Post Conditions:**   * **Success:** The Temp folder is emptied. * **Fail:**No images deleted from temp folder   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | - Time hit 0.00am  - Size of Temp folder is bigger than 500MB | Delete all the images in Temp Folder |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:** N/A  **Business Rules:** N/A | | | |

Table 22: <System> Auto Free Storage Space

##### <System> Auto Active Roll Call

Use Case Diagram

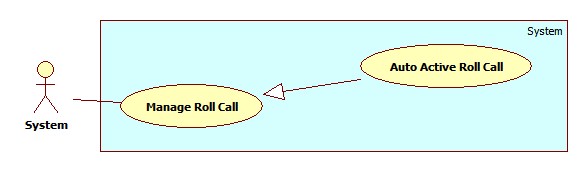


Figure 15: <System> Auto Active Roll Call

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SY004** | | | |
| **Use Case No.** | SY004 | **Use Case Version** | 2.0 |
| **Use Case Name** | Auto Active Roll Call | | |
| **Author** | Pham Huy Hoang | | |
| **Date** | 21/09/2013 | **Priority** | Normal |
| **Actor:**System.  **Summary:**  This use case is used to auto active a roll call  **Goal:**  The roll call on roll call list will be activated when begin date is current date  **Triggers:**   * System will auto active a roll call when begin date is current date. * When a roll call is activated, its status in Roll Call List will be “Active”. * The instructor can only take attendance of an active roll call list.   **Preconditions:** N/A.  **Post Conditions:**   * **Success:** System will auto active a roll call. * **Fail:**No roll call activated   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Roll call begin date is current date.  The system auto activate a roll call. | The roll call status become “Active”.  The roll call is show on instructor’s phone. The instructor can take attendance of the roll call. |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:** Manage Roll Call  **Business Rules:**   * A roll call has 3 status: Created, Active, Inactive. * Staff manage the info of a roll call, but staff can’t active/inactive rollcall manually. * When a roll call is Active, the staff can only Change the Instructor, or Add/Remove Student. | | | |

Table 23: <System> Auto Active Roll Call

##### <System> Auto Inactive Roll Call

Use Case Diagram

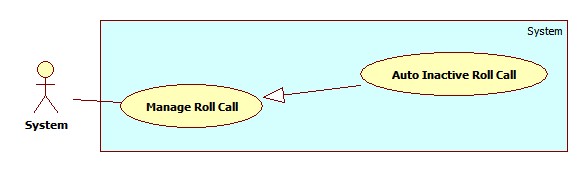


Figure 16: <System> Auto Inactive Roll Call

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SY005** | | | |
| **Use Case No.** | SY005 | **Use Case Version** | 2.0 |
| **Use Case Name** | Auto Active Roll Call | | |
| **Author** | Pham Huy Hoang | | |
| **Date** | 21/09/2013 | **Priority** | Normal |
| **Actor:**System.  **Summary:**  This use case is used to auto inactive a roll call  **Goal:**  The roll call on roll call list will be inactivated when end date expired.  **Triggers:**   * System will auto inactive a roll call when end date expired. * When a roll call is inactivated, its status in Roll Call List will be “Inactive”. * The instructor can’t take attendance of an active roll call list. The staff can’t edit info of an inactive roll call.   **Preconditions:** N/A.  **Post Conditions:**   * **Success:** System will auto inactive a roll call. * **Fail:**No roll call activated   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Roll call end date expired. The system auto inactivate a roll call. | The roll call status become “Inactive”.  The roll call not show instructor’s phone. |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:** Manage Roll Call  **Business Rules:**   * A roll call has 3 status: Created, Active, Inactive. * Staff manage the info of a roll call, but staff can’t active/inactive rollcall manually. * When a roll call is Inactive, the staff can’t edit the roll call info. The instructor can’t take attendance of an inactive roll call. | | | |

Table 24: <System> Auto Inactive Roll Call

#### <Staff >Overview Use Case

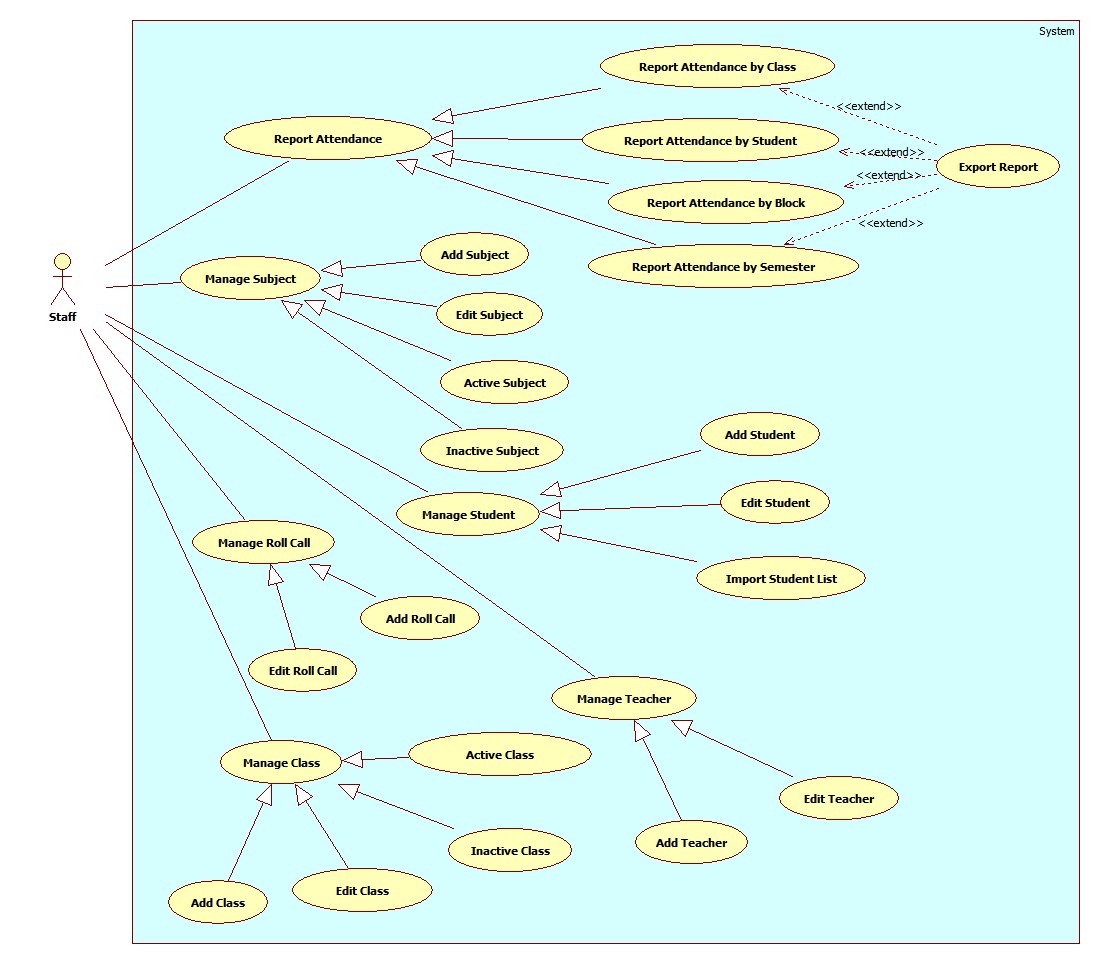


Figure 17: <Staff> Overview Use Case

##### <Staff>Add Roll Call

Use Case Diagram

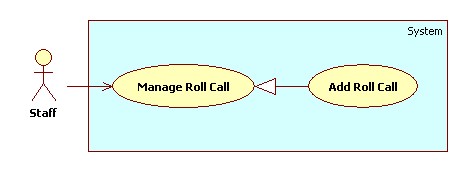


Figure 18: <Staff>Add Roll Call

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU001** | | | |
| **Use Case No.** | SU001 | **Use Case Version** | 2.0 |
| **Use Case Name** | Add Roll Call. | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  Staff can create a new Roll Call.  **Goal:**  A new roll call includes: instructor, subject, class, date, time will be added to database.  **Triggers:**   * Staff can create new roll call for every block/semester * Staff must do these step to create new roll call:   + On Home page, click on link “Manage Roll Call” in menu sidebar and Manage Roll Call page will be showed.   + Then, choose “New Roll Call” tab.   + Then, choose the instructor, subject, class, student, date, time… for class.   + Click on button “Add”.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: A new roll call will be created. The roll call will be displayed on roll call list. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Manage Roll Call” in menu bar. | Redirect to Manage roll call page, include 2 tabs:   * Roll Call List * New Roll Call   Current Tab is Roll Call List, include table roll call list, for each roll call:   * No. : label * Instructor: label * Subject: label * Class: label * Date: label * Time: label * Show Student List: Button * Detail: Button * Status: label | | 2 | Click “New Roll Call” tab. | Display “New Roll Call” tab with :   * Instructor: Drop down list (value: all active instructor in database) * Subject: Drop down list (value: all active subject in database) * Class: Drop down list (value: all active class in database). * Button “Show Student List” * Date picker “Start date” and “End date” * Time: Drop down list(value: 7:00, 8:45, 10:30, 12:30, 14:15, 16:00) * Button “Add” | | 3 | Input roll call information, click “Add” button. | Add new roll call to database. Redirect Roll call list tab.  [Exception 1, 2, 3, 4] |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | The teacher already has different class at selected time. | Show error message “The teacher has a class at XX:XX, choose different time”. | | 2 | Class and subject have different major. | Show error message “The class and subject should have same major”. | | 3 | The subject, which has 2 slots, must not have time start at 10:30 or 16:00. | Show error message“The subject, which has 2 slot, should have time start at 7:00/8:45/12:30 or 14:15 ” | | 4 | Select “End date” is smaller than “Start Date”. | Show error message “The end date must be bigger than start date” |   **Relationships:** Manage Roll Call.  **Business Rules:**   * Start date must be later than current date; end date must be later than start date. * The class and subject belong to the same major. * When staff select a class for roll call, all students in that class is assigned to the roll call. | | | |

Table 25: <Staff>Add Roll Call

##### <Staff>Edit Roll Call

Use Case Diagram

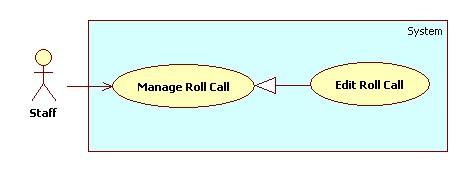


Figure 19: <Staff>Edit Roll Call

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU002** | | | |
| **Use Case No.** | SU002 | **Use Case Version** | 2.0 |
| **Use Case Name** | Edit Roll Call. | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  Staff can use this case to edit a roll call.  **Goal:**  Update new information roll call includes: instructor, subject, class, date, time…  **Triggers:**   * Staff can update new information for roll call to database. * Staff must do these step to edit roll call: * In Home Page, click on link “Manage roll call”. * Click on button “Detail” in row, which staff wants to edit. * In “Roll call detail” pop-up window, edit information. * Click on button “Update”.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: A roll call will be updated. The system will transfer to roll call list page. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Manage Roll call” in menu bar. | Redirect to Manage roll call page, include 2 tabs:   * Roll Call List * New Roll Call   Current Tab is Roll Call List, include table roll call list, for each roll call:   * No. : label * Instructor: label * Subject: label * Class: label * Date: label * Time: label * Show Student List: Button * Detail: Button * Status: label | | 2 | Click button “Detail” in row, which staff wants to edit. | The system will show “Roll call detail” pop-up:   * Instructor: Drop down list (value of instructor list) * Subject: Drop down list (value of subject list) * Class: Drop down list (value of Class list) * Button “Show” * Date picker “Start date” and “End date” * Time: Drop down list(value: 7:00, 8:45, 11:00, 12:30, 14:15, 16:00) * Button “Update” * Button “Cancel” | | 3 | Edit Subject information, click “Update” button. | The system will update roll call to database. The system transfer to roll call list page.  [Exception 1, 2, 3] |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | The teacher teaches different class at time. | Display “The teacher should teach classes in different time”. | | 2 | Class and subject have different major. | Display “The class and subject should have same major”. | | 3 | The subject, which has 2 slots, must not have time start at 11:00 or 16:00. | Display “The subject, which has 2 slot, should have time start at 7:00/8:45/12:30 or 14:15 ” | | 4 | Select “End date” is smaller than “Start Date”. | Show error message “The end date must be bigger than start date” |   **Relationships:** Manage Roll Call.  **Business Rules:**   * In pop-up window, the value of instructor, subject, class, student list, date, time will be auto set by subject was chosen. * If the roll call status is “Created”, the staff can edit all roll call info. * If the roll call status is “Active”, the staff can only change the teacher of roll call. * If the roll call status is “Inactive”, the staff can edit anything. | | | |

Table 26: <Staff>Edit Roll Call

##### <Staff> Add Subject

Use Case Diagram

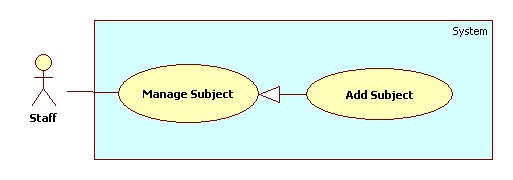


Figure 20: <Staff> Add Subject

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU003** | | | |
| **Use Case No.** | SU003 | **Use Case Version** | 2.0 |
| **Use Case Name** | Add Subject. | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  Staff can create a new Subject.  **Goal:**  A new Subject include: major, name, slot will be added to database.  **Triggers:**   * Staff can create new subject with all information: name, major , slot … * Staff must do these step to create new subject:   + On Home page, click on link “Manage subject” in menu sidebar and Manage subject page will be showed.   + Then, choose “New subject” tab.   + Then, choose the major, input name, block for subject.   + Click on button “Add”.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: A new Subject will be created. The Subject will be displayed on Subject list. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Manage Subject” in menu bar. | Redirect to Manage Subject Page, include 2 tabs:   * Subject List * New Subject   Currnt Tab is Subject List. its include table subject ,for each row:   * No. : label * Major: label * Name: label * Slot (/day): label * Detail: button * Active/Inactive: button | | 2 | Click “New Subject” tab. | Display “New Subject” tab with :   * Major: Multiple chosen (value: Software Engineering, Financial Banking, Business Analysis) * Name: textbox (min length:1, max length: 30, required). * Slot (/day): drop down list (value: 1, 2). | | 3 | Input Subject information, click “Add” button. | Create new Subject to database. Redirect Subject list tab.  [Exception 1] |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Name’s length is not in range [1,30] | Display “Name’s length must be from 1 to 30 words”. |   **Relationships:** Manage Subject.  **Business Rules:**N/A | | | |

Table 27: <Staff> Add Subject

##### <Staff> Edit Subject

Use Case Diagram

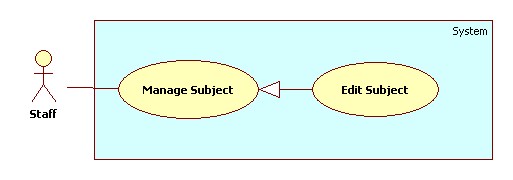


Figure 21: <Staff> Edit Subject

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU004** | | | |
| **Use Case No.** | SU004 | **Use Case Version** | 2.0 |
| **Use Case Name** | Edit Subject. | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  Staff can use this case to edit a Subject.  **Goal:**  Updatenew information Subject includes: major, name, slot…  **Triggers:**   * User can update new information for subject to database. * Staff must do these step to Edit subject: * In Home Page, click on link “Manage subject”. * Click on button “Detail” in row, which staff wants to edit. * In “Subject detail” pop-up window, edit information. * Click on button “Update”.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: A Subject will be updated. The system will transfer to Subject list page. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Manage Subject” in menu bar. | Redirect to Manage Subject Page, include 2 tabs:   * Subject List * New Subject   Currnt Tab is Subject List. its include table subject ,for each row:   * No. : label * Major: label * Name: label * Slot (/day): label * Detail: button * Active/Inactive: button | | 2 | Click button “Detail” in row, which staff wants to edit. | The system will show “Subject detail” pop-up:   * Major: Drop down list (value: Software Engineering, Financial Banking, Business Analysis ) * Name: textbox (min length: 1 max length: 30, required). * Slot (/day): drop down list (value: 1, 2). | | 3 | Edit Subject information, click “Update” button. | The system will update Subject to database. The system transfer to Subject list page.  [Exception 1] |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Name’s length is not in range [1,30] | Display “Name is invalid”. |   **Relationships:** Manage Subject.  **Business Rules:**   * In pop-up win down, the value of major, name, slot will be auto set by subject was chosen. | | | |

Table 28: <Staff> Edit Subject

##### <Staff> Active Subject

Use Case Diagram

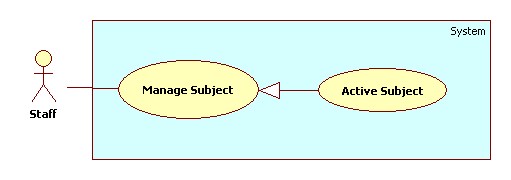


Figure 22: <Staff> Active Subject

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU005** | | | |
| **Use Case No.** | SU005 | **Use Case Version** | 2.0 |
| **Use Case Name** | Active Subject. | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  Staff can use this case to active subject.  **Goal:**  Active a subject for new block.  **Triggers:**   * Staff can active a subject have exit. * Staff must do these step to Active subject: * In Home Page, click on link “Manage subject”. * Click on button “Inactive” in row, which staff wants to edit. * The system will show a message box confirm. * Click on button “Yes”.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: A Subject will be active. The system will still in current manage subject page. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Manage Subject” in menu bar. | Redirect to Manage Subject Page, include 2 tabs:   * Subject List * New Subject   Currnt Tab is Subject List. its include table subject ,for each row:   * No. : label * Major: label * Name: label * Slot (/day): label * Detail: button * Active/Inactive: button | | 2 | Click button “Inactive” in row, which staff wants to edit. | The system will show confirm message box:   * Text: “Do you want to active this object?” * Button “Yes” * Button “No” | | 3 | Click on button “Yes”  [Alternative 1] | The system will change status subject to active and still in current page. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | In confirm message box, staff click on button “No” | The system will still in current page with no value is changed. |   **Exceptions:** N/A  **Relationships:** Manage Subject.  **Business Rules:**   * The button active has color blue and button inactive has color orange. | | | |

Table 29: <Staff> Active Subject

##### <Staff> Inactive Subject

Use Case Diagram

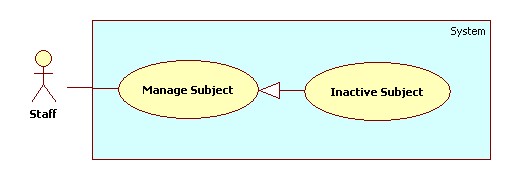


Figure 23: <Staff> Inactive Subject

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU006** | | | |
| **Use Case No.** | SU006 | **Use Case Version** | 2.0 |
| **Use Case Name** | Inactive Subject. | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  Staff can use this case to inactive subject.  **Goal:**  Inactive a subject.  **Triggers:**   * Staff can inactive a subject have exit. * Staff must do these step to Inactive subject: * In Home Page, click on link “Manage subject”. * Click on button “Active” in row, which staff wants to edit. * The system will show a message box confirm. * Click on button “Yes”.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: A Subject will be inactive. The system will still in current manage subject page. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Manage Subject” in menu bar. | Redirect to Manage Subject Page, include 2 tabs:   * Subject List * New Subject   Currnt Tab is Subject List. its include table subject ,for each row:   * No. : label * Major: label * Name: label * Slot (/day): label * Detail: button * Active/Inactive: button | | 2 | Click button “Active” in row, which staff wants to edit. | The system will show confirm message box:   * Text: “Do you want to inactive this object?” * Button “Yes”. * Button “No”. | | 3 | Click on button “Yes”  [Alternative 1] | The system will change status subject to inactive and still in current page. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | In confirm message box, staff click on button “No” | The system will still in current page with no value is changed. |   **Exceptions:** N/A  **Relationships:** Manage Subject.  **Business Rules:**   * The button active has color blue and button inactive has color orange. | | | |

Table 30: <Staff> Inactive Subject

##### <Staff>Add Student

Use Case Diagram

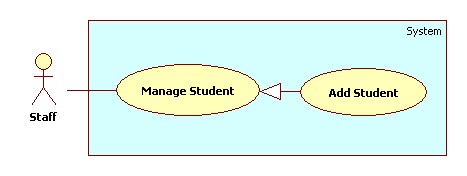


Figure 24: <Staff> Add Student

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU007** | | | |
| **Use Case No.** | SU007 | **Use Case Version** | 2.0 |
| **Use Case Name** | Add Student. | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  Staff can create a new student.  **Goal:**  A new student include: name, email, student id, birthday, citizen id, address will be added to database.  **Triggers:**   * Staff can create new student with all information: name, email, student id, birthday, citizen id, address... * Staff must do these step to create new student:   + On Home page, click on link “Manage student” in menu sidebar and Manage student page will be showed.   + Then, choose “New student” tab.   + Then, choose the class, input name, id, email, citizen id, address for student.   + Click on button “Add”.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: A new student will be created. The student will be displayed on student list. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Manage Student” in menu bar. | Redirect to Manage Student Page, include 2 tabs:   * Student List * New Student * Import Student List   Current Tab is Student List, its include table student ,for each row:   * No: label * Name: label * Current Classes: label * Email: label * Student ID: label * Detail: button * Active/Inactive: button | | 2 | Click “New Student” tab. | Display “New Student” tab with :   * Name: textbox (min length: 1 max length: 30, required). * Student ID: textbox (min length: 1 max length: 30, required). * Birthday: date picker (value: current date.) * Address: textbox (min length: 1 max length: 50, required). * Email: textbox (match regular expression:   ^([\w\.])+@([\w])+\.(\w){2,6}(\.([\w]){2,4})\*$  , required).   * Citizen ID: (match regular expression:^\d{8}$   , required).   * Button “Add” * Button “Cancel” | | 3 | Input Student information, click “Add” button. | Create new student to database. Redirect student list tab.  [Exception 1,2,3,4] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Cancel” button. | All control in “New Student” tab is cleared. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Name’s length is not in range [1,30] | Display “Name is invalid”. | | 2 | Address’s length is not in range [1,50] | Display “Address is invalid”. | | 3 | Student ID’s length is not in range [1,30] | Display “Student ID is invalid”. | | 4 | Email is not matched regular expression. | Display “Invalid email address”. |   **Relationships:** Manage Student.  **Business Rules:** N/A | | | |

Table 31: <Staff> Add Student

##### <Staff>Edit Student

Use Case Diagram

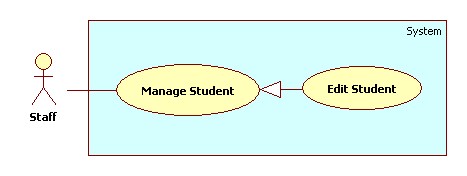


Figure 25: <Staff> Edit Student

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU008** | | | |
| **Use Case No.** | SU008 | **Use Case Version** | 2.0 |
| **Use Case Name** | Edit Student. | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  Staff can use this case to edit a student.  **Goal:**  Update new information student includes: name, birthday, citizen ID, email…  **Triggers:**   * Staff can update new information for student to database. * Staff must do these step to edit student: * In Home Page, click on link “Manage Student”. * Click on button “Detail” in row, which staff wants to edit. * In “Student detail” pop-up window, edit information. * Click on button “Update”.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: A student will be updated. The system will transfer to student list page. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Manage Student” in menu bar. | Redirect to Manage Student Page, include 2 tabs:   * Student List * New Student * Import Student List   Current Tab is Student List, its include table student ,for each row:   * No: label * Name: label * Current Classes: label * Email: label * Student ID: label * Detail: button * Active/Inactive: button | | 2 | Click button “Detail” in row, which staff wants to edit. | The system will show “Student Detail” pop-up:   * Name: textbox (min length: 1 max length: 30, required). * Student ID: textbox (min length: 1 max length: 30, required, disabled). * Birthday: date picker (value: current date.) * Address: textbox (min length: 1 max length: 50, required). * Email: textbox (match regular expression:   ^([\w\.])+@([\w])+\.(\w){2,6}(\.([\w]){2,4})\*$  , required).   * Citizen ID: (match regular expression:^\d{8}$   , required).   * Update: button * Cancel: button | | 3 | Edit student information, click “Update” button.  [Alternative 1] | The system will update student to database. The system transfer to student list page.  [Exception 1,2,3,4] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Cancel” button. | The pop-up is closed. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Name’s length is not in range [1,30] | Display “Name is invalid”. | | 2 | Address’s length is not in range [1,50] | Display “Address is invalid”. | | 3 | Student ID’s length is not in range [1,30] | Display “Student ID is invalid”. | | 4 | Email is not matched regular expression. | Display “Invalid email address”. |   **Relationships:** Manage Student.  **Business Rules:**   * In pop-up window, the value of name, current class, address, citizen id, and birthday will be auto set by student was chosen. | | | |

Table 32: <Staff> Edit Student

##### <Staff>Import Student List

Use Case Diagram

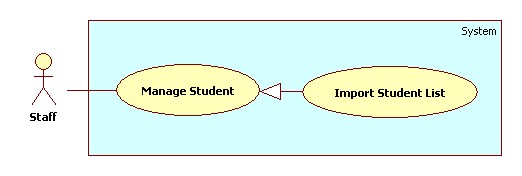


Figure 26: <Staff> Import Student List

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU009** | | | |
| **Use Case No.** | SU009 | **Use Case Version** | 2.0 |
| **Use Case Name** | Import student list | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  Staff can use this case to import student list.  **Goal:**  Create multiple student by import from excel file.  **Triggers:**   * User can import an excel file student list. * Staff must do these step to import student list: * In Home Page, click on link “Manage Student”. * Click on tab “Import Student List” * Click on button “Import” * Then, choose the excel file in browser dialog.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: Student list will be added to database. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Manage Student” in menu bar. | Redirect to Manage Student Page, include 2 tabs:   * Student List * New Student * Import Student List   Current Tab is Student List, its include table student ,for each row:   * No: label * Name: label * Current Classes: label * Email: label * Student ID: label * Detail: button * Active/Inactive: button | Click “Manage Student” in menu bar. | Redirect to Manage Student Page, include 2 tabs:   * Student List * New Student   Student List is tab default, its include table student ,for each row:   * “No.” * “Name”. * “Current Classes”. * “Email”. * “Student ID”. * Button “Detail” * Button “Active/Inactive”. | | 2 | Click “Import Student List” tab | Display “Import Student List” tab with :   * Import file: File upload control. * Button “Import” | | 3 | Click on textbox Import file, choose file. Click “Ok” | The system will get path file and show on text box (only allow excel file.) | | 4 | Click on button “Import” | Import student list to database and show a message box:   * Text:“Import is successful!   There are [number of student in file] student and [number of class in file] class were imported to database”   * Button “Ok”.   [Exception 1] |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Import file get error. | Show error message, the sheet, line, row where error occur. |   **Relationships:** Manage Student.  **Business Rules:**   * The input excel file must follow a student list excel template. | | | |

Table 33: <Staff> Import Student List

##### <Staff>Add Instructor

Use Case Diagram

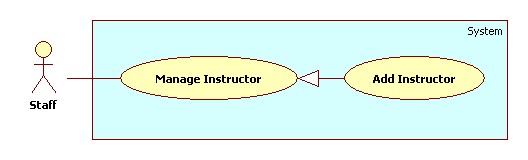


Figure 27: <Staff>Add Instructor

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU010** | | | |
| **Use Case No.** | SU010 | **Use Case Version** | 2.0 |
| **Use Case Name** | Add instructor. | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  Staff can create a new instructor.  **Goal:**  A new instructor include: name, current subject, current class, and phone number will be added to database.  **Triggers:**   * Staff can create new instructor with all information: name, current subject , current class, phone number … * Staff must do these step to create new instructor:   + On Home page, click on link “Manage Instructor” in menu sidebar and Manage Instructor page will be showed.   + Then, choose “New Instructor” tab.   + Then, choose the Classes, subject, input name, phone number for instructor.   + Click on button “Add”.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: A new instructor will be created. The instructor will be displayed on instructor list. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Manage Instructor” in menu bar. | Redirect to Manage instructor page, include 2 tabs:   * Instructor List * New Instructor   Current Tab is Instructor List, include table instructor list, for each instructor:   * No: label * Name: label * Teaching Subject: label * Teaching Class: label * Detail: button * Active/Inactive: button | | 2 | Click “New Instructor” tab. | Display “New Instructor” tab with :   * Instructor name: Text box (min length:1 max length:30, required) * Email: textbox (match regular expression:   ^([\w\.])+@([\w])+\.(\w){2,6}(\.([\w]){2,4})\*$  , required).   * Phone number: text box (min length: 8, max length: 11, match regular expression: ^\s\*\d+\s\*$ ) | | 3 | Input instructor information, click “Add” button. | Create new instructor to database. Redirect instructor list tab.  [Exception 1, 2, 3] |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Name’s length is not in range [1,30] | Display “Name is invalid”. | | 2 | Email not matched regular expresion | Display “Email is invalid”. | | 3 | Phone number is not matched regular expresion. | Display “Phone number is invalid”. |   **Relationships:** Manage instructor.  **Business Rules:** N/A | | | |

Table 34: <Staff>Add Instructor

##### <Staff>Edit Instructor

Use Case Diagram

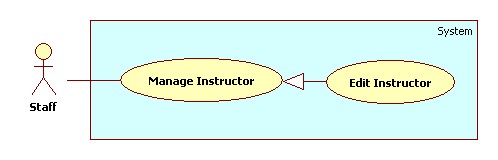


Figure 28: <Staff>Edit Instructor

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU011** | | | |
| **Use Case No.** | SU011 | **Use Case Version** | 2.0 |
| **Use Case Name** | Edit instructor. | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  Staff can use this case to edit an instructor.  **Goal:**  Update new information instructor includes: name, current subjects, current classes, phone…  **Triggers:**   * Staff can update new information for instructor to database. * Staff must do these step to edit instructor: * In Home Page, click on link “Manage Instructor”. * Click on button “Detail” in row, which staff wants to edit. * In “Instructor detail” pop-up window, edit information. * Click on button “Update”.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: An instructor will be updated. The system will transfer to instructor list page. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Manage Instructor” in menu bar. | Redirect to Manage instructor page, include 2 tabs:   * Instructor List * New Instructor   Current Tab is Instructor List, include table instructor list, for each instructor:   * No: label * Name: label * Teaching Subject: label * Teaching Class: label * Detail: button * Active/Inactive: button | | 2 | Click button “Detail” in row, which staff wants to edit. | The system will show “Instructor detail” pop-up:   * Instructor name: Text box (min length:1 max length:30, required) * Email: textbox (match regular expression:   ^([\w\.])+@([\w])+\.(\w){2,6}(\.([\w]){2,4})\*$  , required).   * Phone number: text box (min length: 8, max length: 11, match regular expression: ^\s\*\d+\s\*$ ) * Update: button * Cancel: button | | 3 | Edit instructor information, click “Update” button.  [Alternative 1] | The system will update instructor to database. The system transfer to instructor list page.  [Exception 1, 2] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Cancel” button. | The pop-up is closed. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Name’s length is not in range [1,30] | Display “Name is invalid”. | | 2 | Phone number is not matched regular exception. | Display “Phone number is invalid”. |   **Relationships:** Manage Instructor.  **Business Rules:**   * In pop-up window, the value of name, current class, current subject, phone number will be auto set by instructor was chosen. | | | |

Table 35: <Staff> Edit Instructor

##### <Staff>Add Class

Use Case Diagram

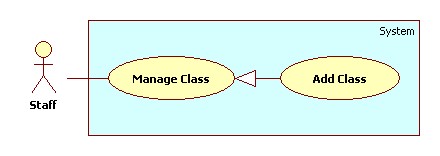


Figure 29: <Staff> Add Class

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU012** | | | |
| **Use Case No.** | SU012 | **Use Case Version** | 2.0 |
| **Use Case Name** | Add Class. | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  Staff can create a new Class.  **Goal:**  A new class includes: major, name will be added to database.  **Triggers:**   * Staff can create new class with for every block/semester/year… * Staff must do these step to create new class:   + On Home page, click on link “Manage Class” in menu sidebar and Manage subject page will be showed.   + Then, choose “New Class” tab.   + Then, choose the major, input name for class.   + Click on button “Add”.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: A new class will be created. The class will be displayed on Class list. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Manage Class” in menu bar. | Redirect to Manage Class Page, include 2 tabs:   * Class List * New Class   Current Tab is Class List, include table class list, for each class:   * No: label * Major: label * Name: label * ShowStudent List: button * Detail: button * Active/Inactive: button | | 2 | Click “New Class” tab. | Display “New Class” tab with :   * Major: Drop down list (value: Software Engineering, Financial Banking, Business Analysis ) * Name: textbox (min length: 1 max length: 30, required). * Add: button * Cancel: button | | 3 | Input class information, click “Add” button. | Create new class to database. Redirect Class list tab.  [Exception 1, 2] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Cancel” button. | The input form is reseted. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Name’s length is not in range [1,30] | Display “Name is invalid”. | | 2 | The name class has existed. | Display “The class has existed”. |   **Relationships:** Manage Class.  **Business Rules:** N/A | | | |

Table 36: <Staff> Add Class

##### <Staff>Edit Class

Use Case Diagram

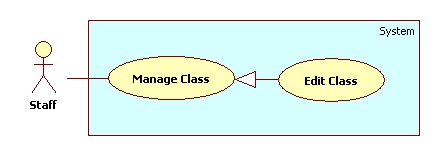


Figure 30: <Staff> Edit Class

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU013** | | | |
| **Use Case No.** | SU013 | **Use Case Version** | 2.0 |
| **Use Case Name** | Edit Class. | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  Staff can use this case to edit a class.  **Goal:**  Update new information class includes: major, name …  **Triggers:**   * User can update new information for class to database. * Staff must do these step to Edit class: * In Home Page, click on link “Manage Class”. * Click on button “Detail” in row, which staff wants to edit. * In “Class detail” pop-up window, edit information. * Click on button “Update”.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: A class will be updated. The system will transfer to class list page. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Manage Class” in menu bar. | Redirect to Manage Class Page, include 2 tabs:   * Class List * New Class   Current Tab is Class List, include table class list, for each class:   * No: label * Major: label * Name: label * ShowStudent List: button * Detail: button * Active/Inactive: button | | 2 | Click button “Detail” in row, which staff wants to edit. | The system will show “Class detail” pop-up:   * Major: Drop down list (value: Software Engineering, Financial Banking, Business Analysis ) * Name: textbox (min length: 1 max length: 30, required). * Table: Student List of class. * Update: button * Cancel: button | | 3 | Edit Class information, click “Update” button.  [Alternative 1] | The system will update class to database. The system transfer to class list page. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Cancel” button. | The pop-up is closed. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | Name’s length is not in range [1,30] | Display “Name is invalid”. |   **Relationships:** Manage Class.  **Business Rules:**   * In pop-up win down, the value of major, name will be auto set by class was chosen. | | | |

Table 37<Staff> Edit Class

##### <Staff> Report Attendance by Class

Use Case Diagram

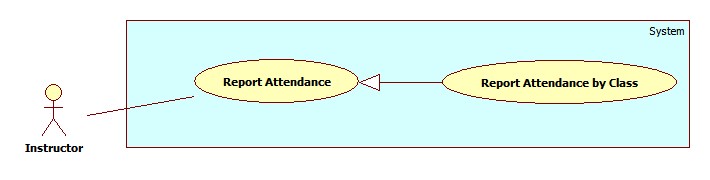


Figure 31: <Staff> Report Attendance by Class

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU014** | | | |
| **Use Case No.** | SU014 | **Use Case Version** | 2.0 |
| **Use Case Name** | Report Attendance by Class | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  This use case allows staff report attendance of all students by class.  **Goal:**  Staff can report attendance of class.  **Triggers:**   * Staff can report information of class: subject, student, attendance… * Staff must do these step to view report result:   + On Home page, click on link “Report” in menu bar and Report page will be showed.   + Then, in select box “Report by” choose type: Class.   + Then, in drop down list “Name” chose the class name.   + Click on button “Statistic”.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: A report attendance by class will be showed on screen. * **Fail:**The system will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Report” link in menu bar. | Redirect to Report Page, include 4 part:   * Drop down list “Report by” (value: Class, Student, Block, and Semester). * Drop down list “Name” (value: auto set based on drop down list Report by) * Button “Statistic” * Button “Export”. | | 2 | Choose “Class” in drop down list Report by. | System will auto set value based on type Class (just active class) for drop down list “Name”. | | 3 | Chose the class in drop down list “Name”. Click button “Statistic” | System will show result attendance by class on screen.  System still in current “Report” page. |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:** Manage Report Attendance.  **Business Rules:**   * System will show attendance result in chosen conditions. * Default value when manager clicks on tab “Report” is: Class. | | | |

Table 38: <Staff> Report Attendance by Class

##### <Staff> Report Attendance by Student

Use Case Diagram

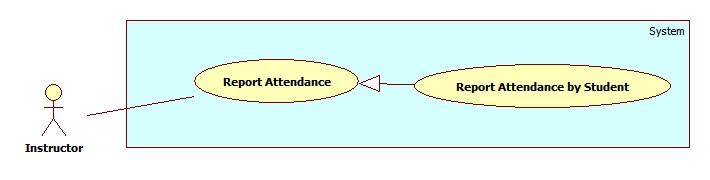


Figure 32: <Staff> Report Attendance by Student

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU015** | | | |
| **Use Case No.** | SU015 | **Use Case Version** | 2.0 |
| **Use Case Name** | Report Attendance by Student | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  This use case allows staff report attendance of student.  **Goal:**  Staff can report attendance of student.  **Triggers:**   * Staff can report information of student: student name, class, subject, attendance… * Staff must do these step to view report result:   + On Home page, click on link “Report” in menu bar and Report page will be showed.   + Then, in drop down list “Report by” choose type: Student.   + Then, in drop down list “Name” choose the student name.   + Click on button “Statistic”.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: A report attendance by student will be showed on screen. * **Fail:** System will not return any value. Transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Report” link in menu bar. | Redirect to Report Page, include 4 part:   * Drop down list “Report by” (value: Class, Student, Block, and Semester). * Drop down list “Name” (value: auto set based on drop down list Report by) * Button “Statistic” * Button “Export”. | | 2 | Choose “Student” in drop down list Report by. | System will auto set value based on type Student (just active student) for drop down list “Name”. | | 3 | Chose the student in drop down list “Name”. Click button “Statistic”. | System will show result attendance by student on screen.  System still in current “Report” page. |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:** Manage Report Attendance.  **Business Rules:**   * System will show attendance result in chosen conditions. * Default value when manager clicks on tab “Report” is: Class. | | | |

Table 39: <Staff> Report Attendance by Student

##### <Staff> Report Attendance by Block

Use Case Diagram

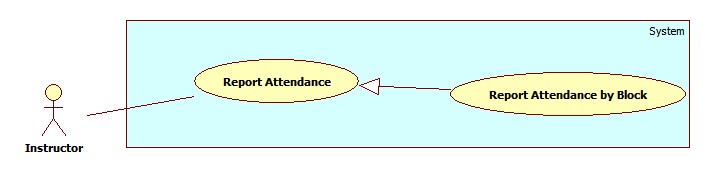


Figure 33: <Staff> Report Attendance by Block

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU016** | | | |
| **Use Case No.** | SU016 | **Use Case Version** | 2.0 |
| **Use Case Name** | Report Attendance by Block | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  This use case allows staff report attendance all students by block.  **Goal:**  Staff can report attendance of student.  **Triggers:**   * Staff can report information of student: Block, class, subject, attendance… * Staff must do these step to view report result:   + On Home page, click on link “Report” in menu bar and Report page will be showed.   + Then, in drop down list “Report by” choose type: Block.   + Then, in drop down list “Name” choose the block.   + Click on button “Statistic”.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: A report attendance by block will be showed on screen. * **Fail:** System will not return any value. Transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Report” link in menu bar. | Redirect to Report Page, include 4 part:   * Drop down list “Report by” (value: Class, Student, Block, and Semester). * Drop down list “Name” (value: auto set based on drop down list Report by) * Button “Statistic” * Button “Export”. | | 2 | Choose “Block” in drop down list Report by. | System will auto set value based on type Block (just active block) for drop down list “Name”. | | 3 | Chose the Block in drop down list “Name”. Click button “Statistic”. | System will show result attendance by block on screen.  System still in current “Report” page. |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:** Manage Report Attendance.  **Business Rules:**   * System will show attendance result in chosen conditions. * Default value when manager clicks on tab “Report” is: Class. | | | |

Table 40: <Staff> Report Attendance by Block

##### <Staff> Report Attendance by Semester

Use Case Diagram

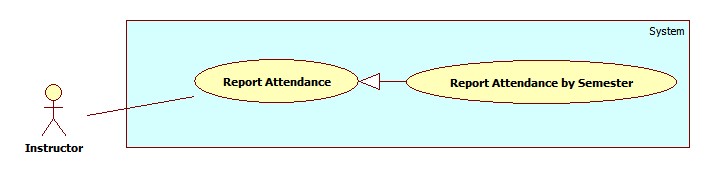


Figure 34: <Staff> Report Attendance by Semester

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU017** | | | |
| **Use Case No.** | SU017 | **Use Case Version** | 2.0 |
| **Use Case Name** | Report by Semester. | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  This use case allows staff report attendance by semester.  **Goal:**  Staff can report attendance by semester.  **Triggers:**   * Staff can report information of student: Block, class, subject … * Staff must do these step to view report result:   + On Home page, click on link “Report” in menu bar and Report page will be showed.   + Then, in drop down list “Report by” choose type: Semester.   + Then, in drop down list “Name” choose the semester.   + Click on button “Statistic”.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: A report attendance by semester will be showed on screen. * **Fail:** System will not return any value. Transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Report” link in menu bar. | Redirect to Report Page, include 4 part:   * Drop down list “Report by” (value: Class, Student, Block, and Semester). * Drop down list “Name” (value: auto set based on drop down list Report by) * Button “Statistic” * Button “Export”. | | 2 | Choose “Semester” in drop down list Report by. | System will auto set value based on type Semester (just active semester) for drop down list “Name”. | | 3 | Chose the semester in drop down list “Name”. Click button “Statistic”. | System will show result attendance by semester on screen.  System still in current “Report” page. |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:** Manage Report Attendance.  **Business Rules:**   * System will show attendance result in chosen conditions. * Default value when manager clicks on tab “Report” is: Class. | | | |

Table 41: <Staff> Report Attendance by Semester

##### <Staff> Export Report

Use Case Diagram

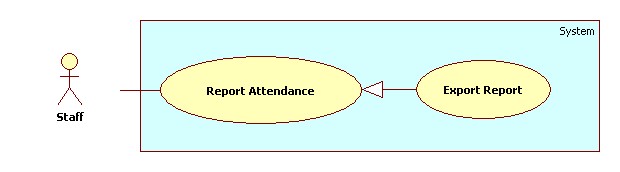
**

Figure 35: <Staff> Export Report

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – SU018** | | | |
| **Use Case No.** | SU018 | **Use Case Version** | 2.0 |
| **Use Case Name** | Export report. | | |
| **Author** | Nguyen Quoc Huy. | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Staff.  **Summary:**  Staff uses this case to Export report.  **Goal:**  Allow staff to export report.  **Triggers:**   * Staff can export report. * On report page, click on button “Export” to export report.   **Preconditions:**   * User must logged in the system with the role is staff.   **Post Conditions:**   * **Success**: Staff can export report. The system still in current page. * **Fail:** N/A.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click on “Export” to request export report on Report page | System will export report to file excel |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:** Manage Report Attendance.  **Business Rules:**   * Staff export report to file excel and attendance: class, block, student, semester … | | | |

#### <Instructor>Overview Use Case

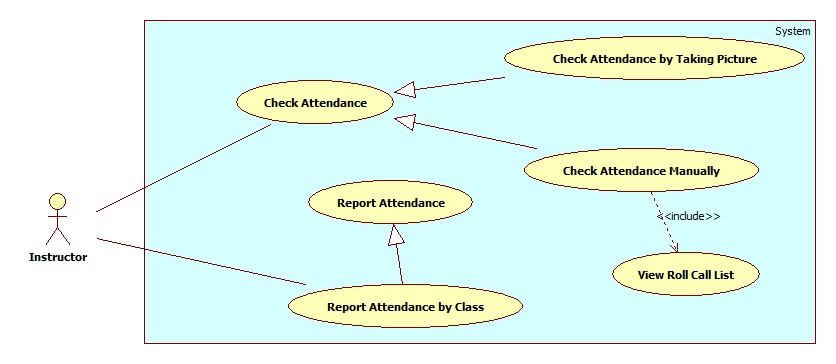


Figure 36: <Instructor> Overview Use Case

##### <Instructor>Capture and Show result attendance

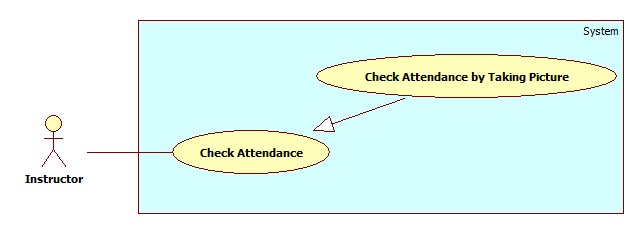


Figure 37: <Instructor>Capture and Show result attendance

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – IU001** | | | |
| **Use Case No.** | IU001 | **Use Case Version** | 2.0 |
| **Use Case Name** | Capture and Show result attendance | | |
| **Author** | Nguyen Thanh Binh | | |
| **Date** | 23/09/2013 | **Priority** | High |
| **Actor:** Instructor.  **Summary:**   * Instructor uses this case to check attendance.   **Goal:**   * Allow instructor capture image, send image to web service and check attendance.   **Triggers:**   * Instructor can use mobile to take picture. This picture will be used for attendance checking ... To check attendance using picture, instructor must do: * Instructor login application and click button “Take Attendance” in Course Activity. System transfer to Camera activity. * On Camera activity, Instructor capture image, then send image to web service for detected face of member. * Instructor receive result from Web service. Instructor view result and check attendance for members.   **Preconditions:**   * User must logged in the system with the role is Instructor * Instructor is assign to class   **Post Conditions:**   * **Success**: Instructor can check attendance for student. The attendance log will be stored. * **Fail:** System will alert error.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Instructor open mobile app.  Instructor type username, password, press “Login” to login into application. | System will show the Course activity with information includes:   * Subject: textview * Class name: textview * Class Time: textview * Class date: textview   **Main menu:**   * Take Attendance: Button | | 2 | Instructor click button “Take Attendance” to capture picture.  [Alternative 1] | System transfer to Camera activity with:   * Camera screen view: Frame layout * Capture: Button | | 3 | Instructor look at screen and click “Capture” button to capture image and send image to web service. | System will save image on real time when instructor click button “Capture”.  Image will show on screen with two button: Retake & Ok.  [Exception 1,2] | | 4 | Instructor click “OK” button to return to Roll Call activity.  [Alternative 2] | System will notify result to Instructor.  Result contains the students detected from image.  [Exception 3, 4, 5] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Instructor click one Roll Call to check attendance manual with use case “IU002”. | System will transfer to Roll Call activity. | | 2 | Instructor click Retake “button” to retake an image. | System will transfer to Camera activity without saving image. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Error with camera (No camera, can’t connect to camera) | System will show popup “Camera is not available”. | | 2 | Can’t send image to service | System will show a popup “Failed to connect server, please check you internet connection and try it again”. Instructor click “OK” button to retake an image. | | 3 | No respond from service | System will show popup “Request timeout. Please try again”. | | 4 | Only some student detected (<30% number of class students) | System will vibrate, sound and notify “Only X student detect. You can take more picture, or check attendance manually (See Use Case IU002)”. | | 5 | Stranger detected | System will vibrate, sound and notify “Stranger Detected”. |   **Relationships:** N/A  **Business Rules:**   * Image : min 1500px \* 1000px max 3552px \* 2000px or size smaller than 3mb; resolution 72dpi (recommend) * Cable: 3mb ~ 6mb, Wifi: 3mb ~ 6mb * The average time for attendance taking is 10 seconds. * In success case, the app will notify normally, without sound or vibration. In exception case, the notification is with sound and vibration. | | | |

Table 42: <Instructor> Check Attendance by Taking Picture

##### <Instructor> Check Attendance Manually

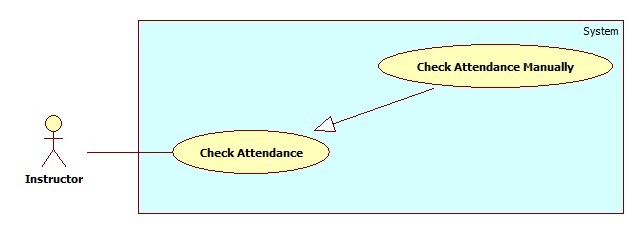


Figure 38: <Instructor> Check Attendance Manually

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – IU002** | | | |
| **Use Case No.** | IU002 | **Use Case Version** | 2.0 |
| **Use Case Name** | Check Attendance Manually | | |
| **Author** | Nguyen Thanh Binh | | |
| **Date** | 23/09/2013 | **Priority** | High |
| **Actor:** Instructor.  **Summary:**   * Instructor uses this case to check attendance manually.   **Goal:**   * Allow instructor check attendance again for some student have special reasons.   **Triggers:**   * Sometimes, the auto attendance checking is not accuracy, the instructor must check attendance manually. * To check attendance again manually, instructor must do: * Instructor login application and choose one roll call in “Course” activity. System transfer to “Roll Call” activity. * Instructor review all members in List student and click button “Log Attendance” transfer to Log activity. * Instructor review click on Log that want to review and check attendance again. The system will show the popup have information about log. * Instructor tick choose the member to check attendance.   **Preconditions:**   * User must logged in the system with the role is Instructor. * Instructor is assign to class.   **Post Conditions:**   * **Success**: Instructor can check attendance for member on this day or previous day. * **Fail:** System will alert error.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Instructor open mobile app.  Instructor type username, password, press “Login” to login into application. | System will show the Course activity with information includes:   * Subject: textview * Class name: textview * Class Time: textview * Class date: textview   **Main menu:**   * Take Attendance: Button | | 2 | Instructor click one “Roll Call” row to view detail of that Roll Call.  [Alternative 1] | System will transfer to Roll Call activity with:   * Class Name: textview * Subject: textview * Date & Time: textview * Student List: (Name (textview), Code (textview)) | | 3 | Instructor click “Log Attendance” button with correlative date. | System will show Log detail page on correlative with :   * Date: textview * Present: textview | | 4 | Instructor click one “Log” row to view detail of that Log. | System will show popup with:   * Date: textview * Student Name: textview * Checkbox: checkbox | | 5 | Instructor check the student to check attendance.  [Alternative 2]  Click button “Ok” send request to check attendance for present member.  [Alternative 3] | System will send request check attendance for present student to server. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Instructor click button “Take Attendance” to check attendance with Use case “IU001”. | System will transfer to Camera activity. | | 2 | Instructor uncheck student. | System will set absent for correlative student. | | 3 | Instructor click “Cancel” button | System will close the popup without saving result. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Can’t connection to service | System will show a popup “Failed to connect server, please check you internet connection and try it again”. Instructor click “OK” button to transfer Roll Call page. | | 2 | No respond from service | System will show popup “Request timeout. Please try again”. |   **Relationships:** N/A  **Business Rules:**   * Cable: 3mb ~ 6mb, Wifi: 3mb ~ 6mb * Instructor can check present member again for student have special reason. | | | |

Table 43: <Instructor> Check Attendance Manually

##### <Instructor> View Roll Call Detail

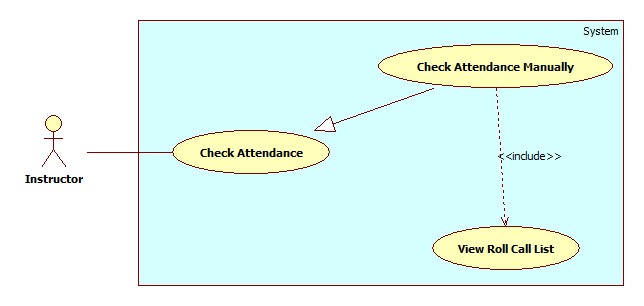


Figure 39: <Instructor> View Roll Call Detail

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – IU003** | | | |
| **Use Case No.** | IU003 | **Use Case Version** | 2.0 |
| **Use Case Name** | View Roll Call Detail | | |
| **Author** | Nguyen Thanh Binh | | |
| **Date** | 23/09/2013 | **Priority** | Normal |
| **Actor:** Instructor.  **Summary:**   * Instructor uses this case to view all roll call detail.   **Goal:**   * Allow instructor view roll call detail with selected roll call, class.   **Triggers:**   * Instructor login application and choose one roll call in “Course” activity. System transfer to “Roll Call” activity. * Instructor review all members with course, class and selected day.   **Preconditions:**   * User must logged in the system with the role is Instructor. * Instructor is assign to class.   **Post Conditions:**   * **Success**: Instructor can view list of students with selected roll call, class. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Instructor open mobile app.  Instructor type username, password, press “Login” to login into application. | System will show the Course activity with information includes:   * Subject: textview * Class name: textview * Class Time: textview * Class date: textview   **Main menu:**   * Take Attendance: Button | | 2 | Instructor click one “Roll Call” row to view detail of that Roll Call.  [Alternative 1] | System will transfer to Roll Call activity with:   * Class Name: textview * Subject: textview * Date & Time: textview * Student List: (Name (textview), Code (textview)) |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Instructor click button “Take Attendance” to check attendance with Use case “IU001”. | System will transfer to Camera activity. |   **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Can’t connection to service | System will show a popup “Failed to connect server, please check you internet connection and try it again”. Instructor click “OK” button to transfer Course page. |   **Relationships:** Check Attendance Manually  **Business Rules:**   * Cable: 3mb ~ 6mb, Wifi: 3mb ~ 6mb * Instructor can view all roll call detail, help to review status of student and check attendance again for students have special reason. | | | |

Table 44: <Instructor> View Roll Call List

##### <Instructor> Report Attendance by Class



Figure 40: <Instructor> Report Attendance by Class

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – IU004** | | | |
| **Use Case No.** | IU004 | **Use Case Version** | 2.0 |
| **Use Case Name** | Report Attendance by Class | | |
| **Author** | Nguyen Thanh Binh. | | |
| **Date** | 23/09/2013 | **Priority** | Normal |
| **Actor:** Instructor.  **Summary:**  This use case allows Instructor report attendance of all students by class.  **Goal:**  Instructor can report attendance of class.  **Triggers:**   * Instructor can report information of class: subject, student, attendance… * Instructor must do these step to view report result: * On Course page, click on link “Report” in menu bar and Report page will be showed. * Then, in select box “Report by” choose type: Class. * Then, in drop down list “Name” chose the class name. * Click on button “Statistic”.   **Preconditions:**   * User must logged in the system with the role is Instructor.   **Post Conditions:**   * **Success**: A report attendance by class will be showed on screen. * **Fail:** System will not return any value. Transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Click “Report” link in menu bar. | Redirect to Report Page, include 4 part:   * Drop down list “Report by” (value: Class, Student, Block, and Semester). * Drop down list “Name” (value: auto set based on drop down list Report by) * Button “Statistic” * Button “Export”. | | 2 | Choose “Class” in drop down list Report by. | System will auto set value based on type Class. | | 3 | Chose the class in drop down list “Name”. Click button “Statistic”. | System will show result attendance by class on screen.  System still in current “Report” page. |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:** Manage Report Attendance.  **Business Rules:**   * System will show attendance result in chosen conditions. * Default value when manager clicks on tab “Report” is: Class. | | | |

Table 45: <Instructor> Report Attendance by Class

#### <Student>Overview Use Case

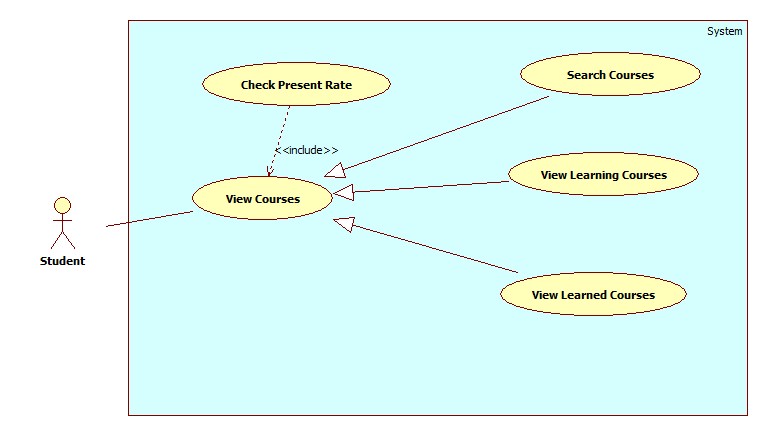


Figure 41: <Student> Overview Use Case

##### <Student>Check Present Rate by Searched Course

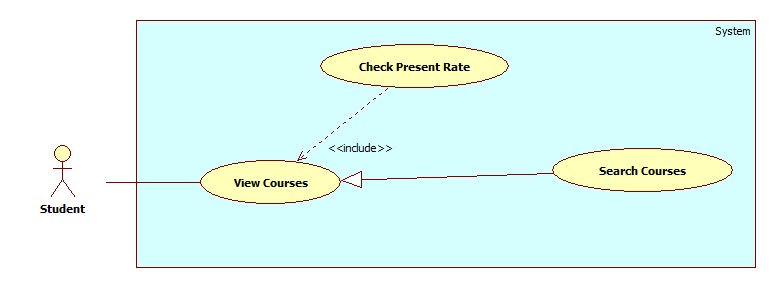
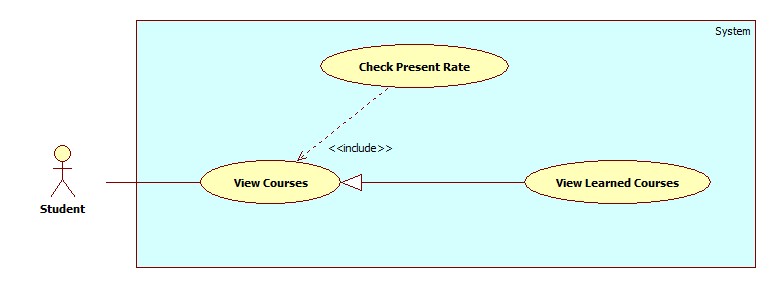


Figure 42: <Student>CheckPresent Rate

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – STU001** | | | |
| **Use Case No.** | STU001 | **Use Case Version** | 2.0 |
| **Use Case Name** | Check Present Rate by Searched Course | | |
| **Author** | Nguyen Thanh Binh | | |
| **Date** | 23/09/2013 | **Priority** | Normal |
| **Actor:** Student.  **Summary:**   * Student uses this case to search learned/learning course.   **Goal:**   * Allow student views learned/learning course info. The search condition is course name.   **Triggers:**   * Student can search course name on list course page. The searching results follow course name. * On list courses page, student choose search condition and clicks search button. System will show list search in list course table at current page. * On list search page, student choose the course. System will show information detail about that course.   **Preconditions:**   * User must logged in the system with the role is Student.   **Post Conditions:**   * **Success**: The search course will be found. System will show list search in lists course page. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | At lists course page, student choose search condition. | System will show the search condition include:  Course Name: text box | | 2 | After chose search condition, student clicks on search button (submit button) finish. | System will show list search of courses that student searched by condition in List Course table.  List Courses table includes 5 column: No (number), Course Name, Semester, Time, Present Rate. | | 3 | Ai list course table, student choose the course they want to view and click on that course name. | System will show information about that course in current page.  Information course includes: Roll call, Course Name, Semester, Time, Student Name & Code, Present, Present Rate and Detail (about exactly day present or absent) |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Search not found | System will show “No course was found!” |   **Relationships:** Check Present Rate  **Business Rules:**   * The courses will be search by course name. * Student can use search course to view their course easily. | | | |

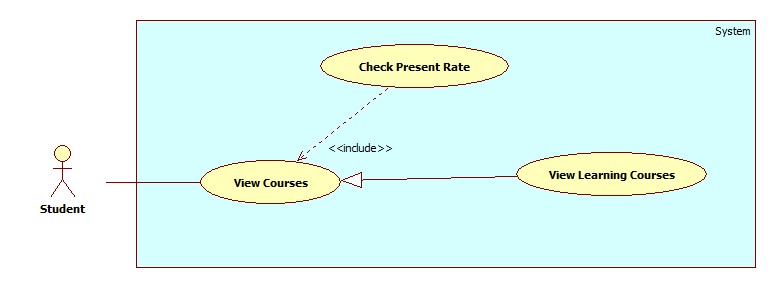
Table 46: <Student> CheckPresent Rate

##### <Student>Check Present Rate by Learned Course



|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – STU002** | | | |
| **Use Case No.** | STU002 | **Use Case Version** | 2.0 |
| **Use Case Name** | Check Present Rate by Learned Course | | |
| **Author** | Nguyen Thanh Binh | | |
| **Date** | 23/09/2013 | **Priority** | Normal |
| **Actor:** Student.  **Summary:**   * Student uses this case to search learned course.   **Goal:**   * Allow student views learned course info. The search condition is semester.   **Triggers:**   * On list learned courses page, student choose search condition. System will show list search in list course table at current page. * On list search page, student choose the course. System will show information detail about that course.   **Preconditions:**   * User must logged in the system with the role is Student.   **Post Conditions:**   * **Success**: The search course will be found. System will show list search in lists course page. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | At lists course page, student choose search condition. | System will show the search condition include:  Semester: drop down list | | 2 | After chose search condition, student view list learned course will be show on current page. | System will show list search of courses that student searched by condition in List Course table.  List Courses table includes 5 column: No (number), Course Name, Semester, Time, Present Rate. | | 3 | Ai list course table, student choose the course they want to view and click on that course name. | System will show information about that course in current page.  Information course includes: Roll call, Course Name, Semester, Time, Student Name & Code, Present, Present Rate and Detail (about exactly day present or absent) |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:** Check Present Rate  **Business Rules:**   * The learned courses will be search by semester. * Student can use search course to view their course easily. | | | |

##### <Student>Check Present Rate by Learning Course



|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – STU003** | | | |
| **Use Case No.** | STU003 | **Use Case Version** | 2.0 |
| **Use Case Name** | Check Present Rate by Learning Course | | |
| **Author** | Nguyen Thanh Binh | | |
| **Date** | 23/09/2013 | **Priority** | Normal |
| **Actor:** Student.  **Summary:**   * Student uses this case to search learning course.   **Goal:**   * Allow student views learning course info.   **Triggers:**   * On list learning courses page student choose the course. System will show information detail about that course.   **Preconditions:**   * User must logged in the system with the role is Student.   **Post Conditions:**   * **Success**: The search course will be found. System will show list search in lists course page. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Student choose view Learning Course page. | System will show list learning courses. List Courses table includes 5 column: No (number), Course Name, Semester, Time, Present Rate. | | 2 | Ai list course table, student choose the course they want to view and click on that course name. | System will show information about that course in current page.  Information course includes: Roll call, Course Name, Semester, Time, Student Name & Code, Present, Present Rate and Detail (about exactly day present or absent) |   **Alternative Scenario:** N/A  **Exceptions:** N/A  **Relationships:** Check Present Rate  **Business Rules:**   * Student can use search course to view their course easily. | | | |

#### <Guest>Overview Use Case



Figure 43: <Guest> Overview Use Case

##### <Guest> Login

Use Case Diagram



Figure 44: <Guest> Login

Use Case Specification

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – GU001** | | | |
| **Use Case No.** | GU001 | **Use Case Version** | 2.0 |
| **Use Case Name** | Login | | |
| **Author** | Pham Huy Hoang | | |
| **Date** | 19/09/2013 | **Priority** | Normal |
| **Actor:**Guest.  **Summary:**  Guest use this case to login into system.  **Goal:**  Allow authentication andauthorization of the system.  **Triggers:**   * Guest want to login into system. * To login: Guest go to the login page, guest enter username and password, then click on “Log in” button to login.   **Preconditions:** N/A.  **Post Conditions:**   * **Success:** Guest is authorized, redirect to corresponding page. * **Fail:**System will show error on current page   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Guest go to login page. | System show a login form, includes:   * Username: textbox (min length: 5, max length: 30, required). * Password: textbox (password box, min length:5, max length: 30, required). * Login: button. | | 1 | Guess enter username and password into textboxs.  Press “Log In” button. | Guest is logged into system, authorized with corresponding role. Redirect to corresponding page..  [Exception 1,2] |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | No | Actor Action | System Response | | 1 | No input in “Username” or “Password” textboxs | Show error message: “Please enter username/password” below Username/Password textbox. | | 2 | Username/Password not in range [5,30] | Show error message: “Username/Password must be from 5 to 30 characters” textbox. | | 3 | Input invalid “Username” and “Password” | Show error message: “Invalid username or password”. |   **Relationships:** N/A  **Business Rules:**   * Each student, instructor, staff, admin has an account. * Only active account can log in. * Only instructor’s account can log in into mobile app. | | | |

Table 47: <Guest> Login

#### <Authorized User> Overview Use case

##### <Authorized User> Overview Use Case

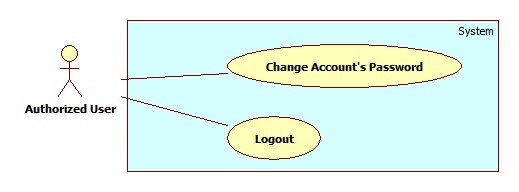


Figure 45: < Authorized User> Overview Use Case

##### <Authorized User> Change Password

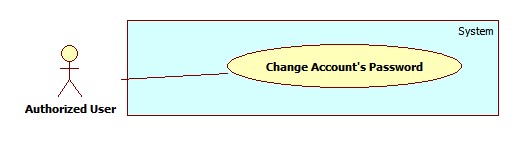


Figure 46: < Authorized User> Change Password

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UU001** | | | |
| **Use Case No.** | UU001 | **Use Case Version** | 2.0 |
| **Use Case Name** | Change Password | | |
| **Author** | Nguyen Thanh Binh | | |
| **Date** | 23/09/2013 | **Priority** | Normal |
| **Actor:**User (Instructor, Staff, Student, Admin).  **Summary:**   * User users this case to change their password.   **Goal:**   * Allow users can change their password.   **Triggers:**   * User use this use case to help improve their account security. * User login their account, click “Change Password” link. Go to “Change Password” page.   **Preconditions:**   * User must logged in the system with the role is User (Staff, Instructor, Student, Admin).   **Post Conditions:**   * **Success**: The password be changed. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | User selects the “Change Password” command in the top-right column of Home page. | System will show “Change Password” page includes:   * Username – Text box * Current Password – Text box * New Password – Text box * Confirm New Password – Text box * Submit – Button * Reset - Button | | 2 | - User enters the following information (Username, Current Password, New password, Confirm new password).  - Click “Submit” button to send request. | * System changes the user’s password. * System automatically re-authenticates the user with the new credentials. * System will show “Your password was changed successfully!” |   **Alternative Scenario:** N/A  **Exceptions:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | System verifies that the Username and Current Password entries are correct and correspond to a current user. | System will show “Wrong username/ password” and reset all. | | 2 | New password same current password | System will show “New password was used! Please choose new password” and reset all. | | 3 | New password and confirm new password does not match. | System will show “Password does not match” and reset all. |   **Relationships:**  **Business Rules:N/A** | | | |

Table 48: < Authorized User> Change Password

##### <Authorized User>Logout

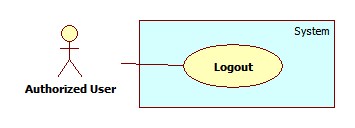


Figure 47: < Authorized User> Logout

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UU002** | | | |
| **Use Case No.** | UU002 | **Use Case Version** | 2.0 |
| **Use Case Name** | Change Password | | |
| **Author** | Nguyen Thanh Binh | | |
| **Date** | 23/09/2013 | **Priority** | Normal |
| **Actor:**User (Instructor, Staff, Student, Admin).  **Summary:**   * User use this use case to logout system.   **Goal:**   * The user selects the option to log off of the HCPT website and is then redirected back to the Login page.   **Triggers:**   * The use case starts when the user selects the option to logout. * The user selects the option to logout. * The system informs the user of logging out. * The user confirms logging out. * The system redirects user back to the Login page.   **Preconditions:**   * User must logged in the system with the role is User (Staff, Instructor, Student, Admin).   **Post Conditions:**   * **Success**: User logout system and redirect to login page. * **Fail:** System will transfer to error page or still in current page.   **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | The user selects log out. | The system will show a pop-up: Do you really want to logout”   * Yes: button * No: button | | 2 | User select “Yes”.  [Alternative 1] | The system will clear user session and the user is logged out |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | Step | Actor Action | System Response | | 1 | Userselects No. | User stay at current page. |   **Exceptions:** N/A  **Relationships:** N/A  **Business Rules:** N/A | | | |

Table 49: <User> Logout

## Software System Attribute

### Usability

#### Graphic User Interface

* All the text, label and image in staff, instructor and student page should be English.
* All the text, label and image in admin page should be English.

#### Usability

* Website admin, staff and instructor should need more than one week of training to be productive with the system.
* The student can use the system easily without training.

#### Installation

* The system must be easy to deploy. Customer can deploy successfully and learn to configure, maintain the system within one day of training.
* The mobile app must be easy to install. Compatible with almost Android Phone.
* The attached manual guide must be clear. User can read and do themselves without developer’s help.

### Reliability

* N/A

### Availability

* N/A

### Security

* Privacy: Each role of user has a specific permission to interact with system.
* System always checks authorization and authenticated before doing anything.
* Only admin can grant permission to staff.

### Maintainability

* N/A

### Portability

* N/A

### Performance

* Detect Faces From an Image: 3~4 seconds/image.
* Recognize Faces: 20~30 seconds/100 faces. With training set of 30 students, 600 images.
* The image upload speed depend on the speed of the network.

## ERD

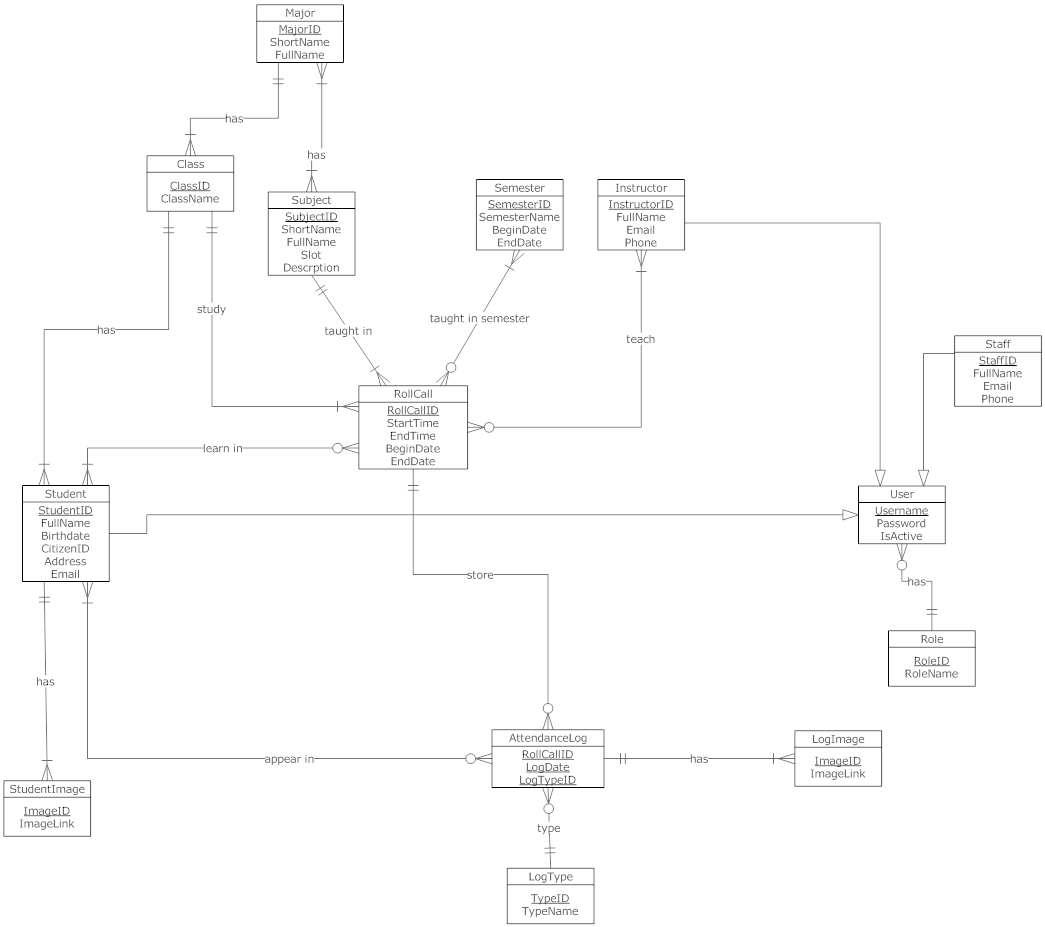


Figure 48: ERD - Conceptual Diagram

# Software Desgin Description

## Design Overview

* This document describes the technical and user interface design of The Roll System using mobile device. It includes the architectural design, the detailed design of common functions and business functions and the design of database model.
* The architectural design describes the overall architecture of the system and the architecture of each main component and subsystem.
* The detailed design describes static and dynamic structure for each component and functions. It includes class diagrams, class explanations and sequence diagrams for each use cases.
* The database design describes the relationships between entities and details of each entity.
* Document overview:
* Section 2: gives an overall description of the system architecture design.
* Section 3: gives component diagrams that describe the connection and integration of the system.
* Section 4: gives the detail design description include class diagram, class explanation, and sequence diagram to details the application functions.
* Section 5: describe an ERD with logical diagram.

## System Architectural Design



Figure 49: MVC Architecture

**(http://www.w3schools.com/aspnet/mvc\_intro.asp)**

**The Model** is the part of the application that handles the logic for the application data.  
Often model objects retrieve data (and store data) from a database.

**The View** is the parts of the application that handles the display of the data.  
Most often the views are created from the model data.

**The Controller** is the part of the application that handles user interaction.  
Typically controllers read data from a view, control user input, and send input data to the model.

## Component Diagram

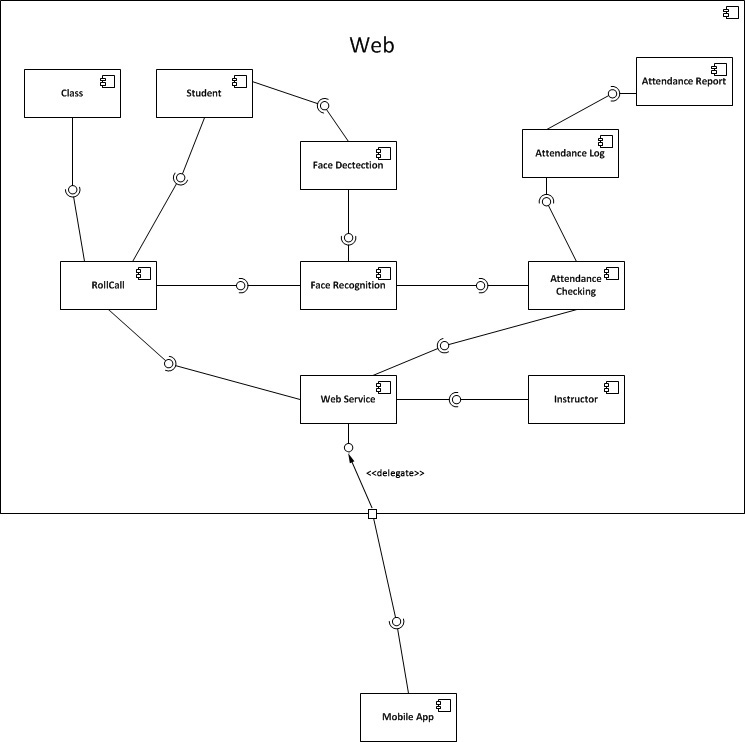


Figure 50: Component Diagram

## Detailed Description of Components

### Class Diagram

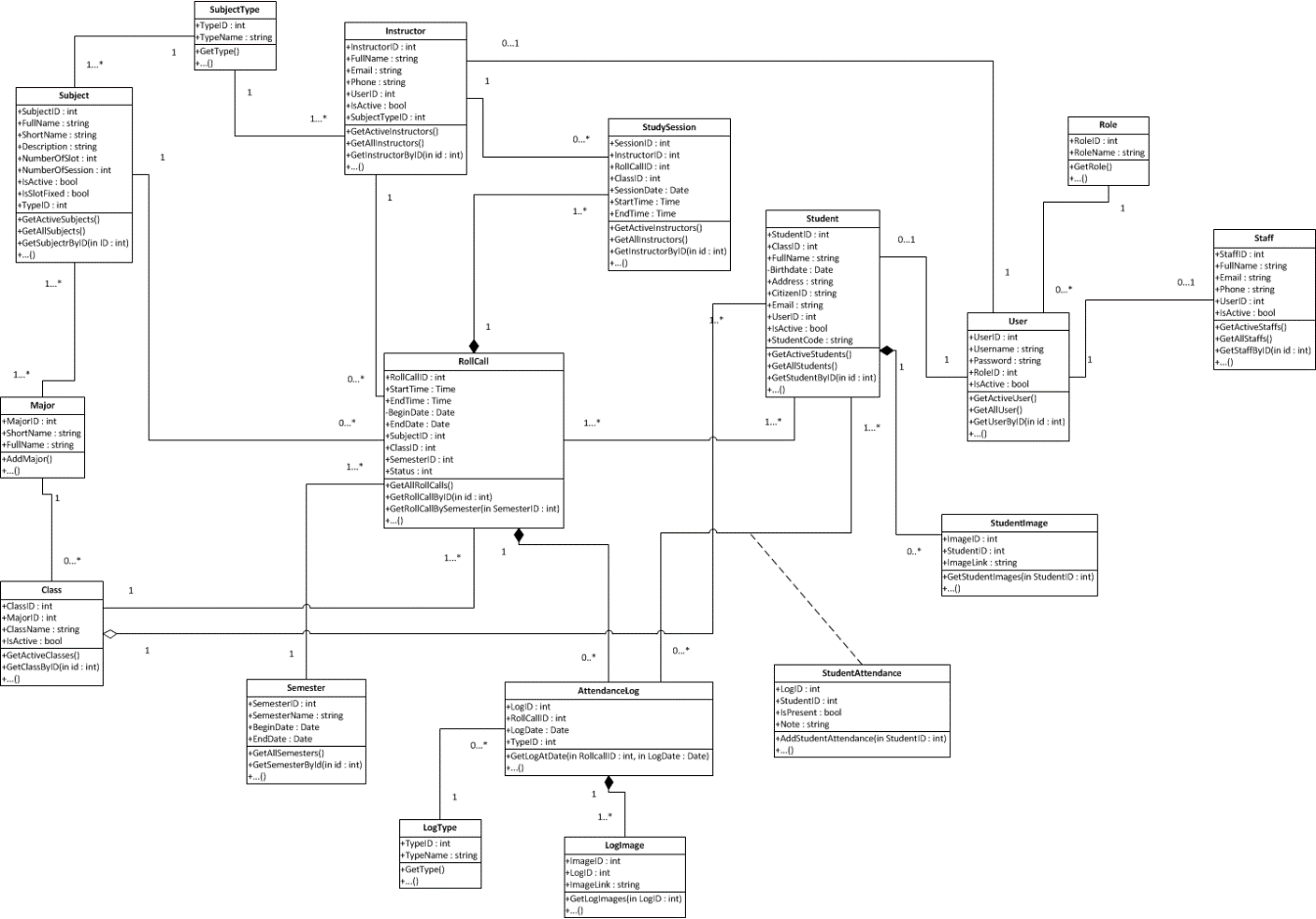


Figure 51: Class Diagram

### Class Diagram Explanation

#### Roll Call

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| RollcallID | Int | Public | Unique id of each roll call |
| StartTime | Time | Public | Start time of roll call |
| EndTime | Time | Public | End time of roll call |
| BeginDate | Date | Public | Begin date of roll call |
| EndDate | Date | Public | End date of roll call |
| SubjectID | Subject | Public | Subject of roll call |
| ClassID | Class | Public | Class of roll call |
| SemesterID | Semester | Public | Semester of roll call |
| Status | Int | Public | Status of roll call |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return Type** | **Visibility** | **Description** |
| getAllRollcalls | Void | Public | Get all roll call in system |
| getRollcallbyID | Rollcall | Public | Get roll call in system by id |
| getRollcallBySemester | Rollcall | Public | Get roll call in system by semester |

#### Subject

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| SubjectID | Int | Public | Unique id of each subject |
| FullName | String | Public | Full name of subject |
| ShortName | String | Public | Short name of subject |
| Description | String | Public | Description of subject |
| NumberOfSlot | Int | Public | Number of slot of subject |
| NumberOfSession | Int | Public | Number of slot of subject |
| IsActive | Boolean | Public | Status active of subject |
| IsSlotFixed | Booean | Public | Number of slot of special subject |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return Type** | **Visibility** | **Description** |
| GetActiveSubjects | Void | Public | Get subject are active. |
| GetAllSubjects | Void | Public | Get all subject in system |
| GetSubjectByID | Subject | Public | Get subject in system by id |

#### Major

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| MajorID | Int | Public | Unique id of each major |
| ShortName | String | Public | Short name of major |
| FullName | String | Public | Full name of major |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return Type** | **Visibility** | **Description** |
| AddMajor | Void | Public | Add new Major to database |

#### Class

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ClassID | Int | Public | Unique id of each class |
| MajorID | Major | Public | Major of class |
| ClassName | String | Public | Name of class |
| isActive | Boolean | Public | Status active of class |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return Type** | **Visibility** | **Description** |
| GetActiveClasses | Void | Public | Get all classes are active in system |
| GetClassByID | Class | Public | Get class in system by id |

#### Instructor Teaching

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ID | Int | Public | Unique id of each Instructor teaching |
| InstructorID | Instructor | Public | Instructor of instructor teaching |
| RollcallID | Roll call | Public | Roll call of instructor teaching |
| BeginDate | Date | Public | Begin date of instructor teaching |
| EndDate | Date | Public | End date of instructor teaching |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return Type** | **Visibility** | **Description** |
| AddInstructorTeachings | Void | Public | Add new instructor teaching to database |

#### Instructor

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| InstructorID | Int | Public | Unique id of each instructor |
| FullName | String | Public | Full name of instructor |
| Email | String | Public | Email of instructor |
| Phone | String | Public | Phone of instructor |
| UserID | User | Public | User name of instructor |
| IsActive | Boolean | Public | Status active of instructor |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return Type** | **Visibility** | **Description** |
| GetActiveInstructors | Void | Public | Get all instructor active in system |
| GetAllInstructors | Void | Public | Get all instructors in system |
| GetInstructorByID | Instructor | Public | Get instructor in system by id |

#### Semester

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| SemesterID | Int | Public | Unique id of each semester |
| SemesterName | String | Public | Name of semester |
| BeginDate | Date | Public | Begin date of semester |
| EndDate | Date | Public | End date of semester |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return Type** | **Visibility** | **Description** |
| GetAllSemesters | Void | Public | Get all semesters in system |
| GetSemesterByID | Semester | Public | Get semester in system by id |

#### Student

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| StudentID | Int | Public | Unique id of each student |
| ClassID | Class | Public | Class of student |
| FullName | String | Public | Full name of student |
| Birthday | Date | Public | Birthday of student |
| Address | String | Public | Address of student |
| CitizenID | String | Public | Citizen id of student |
| Email | String | Public | Email of student |
| UserID | User | Public | User id of student |
| IsActive | Boolean | Public | Status active of student |
| StudentCode | String | Public | Code of student |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return Type** | **Visibility** | **Description** |
| GetActiveStudents | Void | Public | Get all active student in system |
| GetAllStudents | Void | Public | Get all students in system |
| GetStudentByID | Student | Public | Get student in system by id |

#### Role

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| RoleID | Int | Public | Unique id of each role |
| RoleName | String | Public | Name of role |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return Type** | **Visibility** | **Description** |
| GetRole | Void | Public | Get role in system. |

#### Staff

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| StaffID | Int | Public | Unique id of each staff |
| FullName | String | Public | Full name of staff |
| Email | String | Public | Email of staff |
| Phone | String | Public | Phone number of staff |
| UserID | User | Public | UserID of staff |
| Isactive | Boolean | Public | Status active of staff |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return Type** | **Visibility** | **Description** |
| GetActiveStaffs | Void | Public | Get all staffs active in system |
| GetAllStaffs | Void | Public | Get all staff in system |
| GetStaffByID | Staff | Public | Get staff in system by id |

#### User

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| UserID | Int | Public | Unique id of each user |
| UserName | String | Public | Username of user |
| Password | String | Public | Password of user |
| RoleID | Role | Public | Role of user |
| IsActive | Boolean | Public | Status active of user |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return Type** | **Visibility** | **Description** |
| GetActiveUser | Void | Public | Get all users active in system |
| GetAllUser | Void | Public | Get all users in system |
| GetUserByID | User | Public | Get user in system by id |

#### StudentImage

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ImageID | Int | Public | Unique id of each image |
| StudentID | Student | Public | Student of image |
| ImageLink | String | Public | Link of image |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return Type** | **Visibility** | **Description** |
| GetStudentImages | Student Image | Public | Get images of student in system |

#### StudentAttendance

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| LogID | Int | Public | Log of student attendance |
| StudentID | Int | Public | Student of attendance |
| IsPresent | Boolean | Public | Status present of student |
| Note | String | Public | Note of student attendance |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return Type** | **Visibility** | **Description** |
| AddStudentAttendance | Student Attendance | Public | Add new student attendance |

#### AttendanceLog

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| LogID | Int | Private | Unique id of each log |
| RolllCallID | Int | Public | Roll call of attendance log |
| LogDate | Date | Public | Date of log |
| TypeID | Type | Public | Type of log |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return Type** | **Visibility** | **Description** |
| GetLogAtDate | Attendance log | Public | Get log in system by date |

#### LogType

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| TypeID | Int | Public | Unique id of each log type |
| TypeName | String | Public | Name of log type |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return Type** | **Visibility** | **Description** |
| GetType | Type | Public | Get type of log |

#### LogImage

Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Type** | **Visibility** | **Description** |
| ImageID | Int | Public | Unique id of each image |
| LogID | Log | Public | Log of image |
| ImageLink | String | Public | Link of image |

Method

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Return Type** | **Visibility** | **Description** |
| GetLogImages | Log image | Public |  |
|  |  |  |  |

### Sequence Diagram

#### Add Image for Singe Student

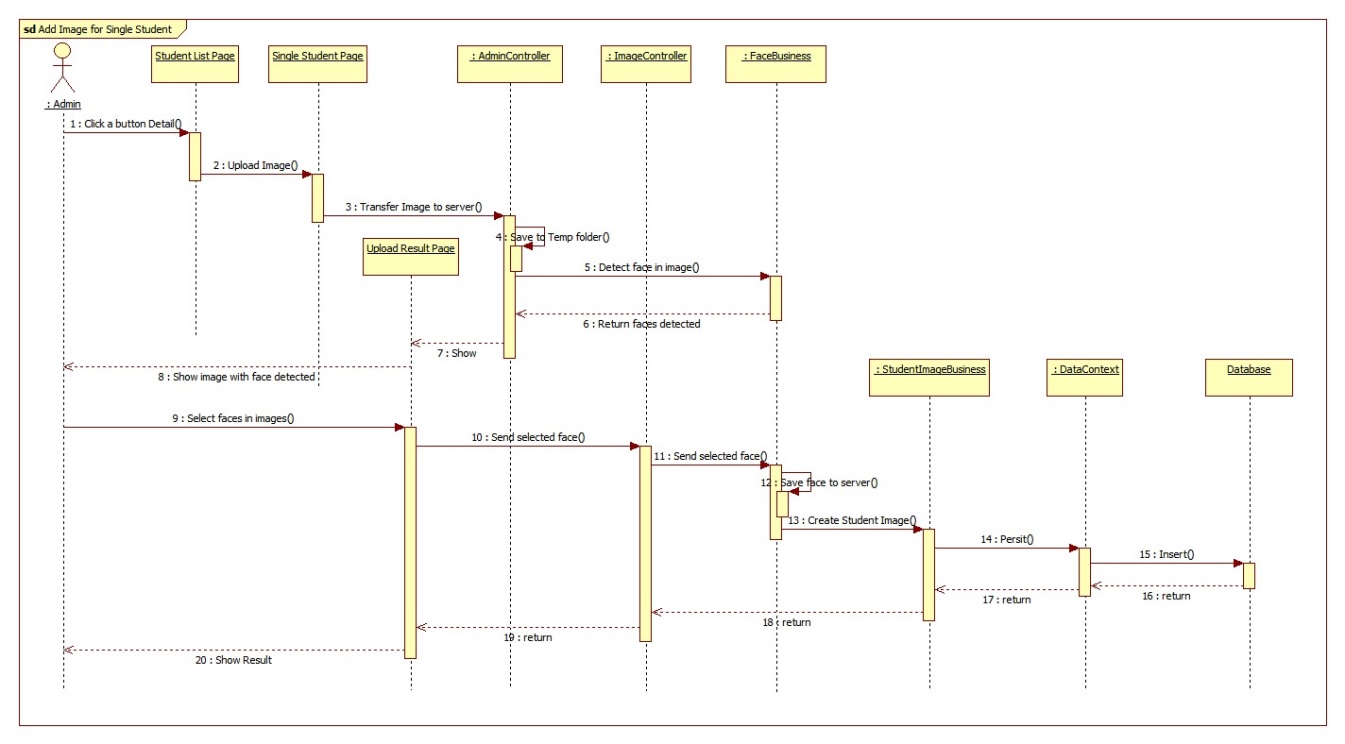


Figure 52: Add Image for Singe Student Sequence Diagram

#### Add Image for Many Students

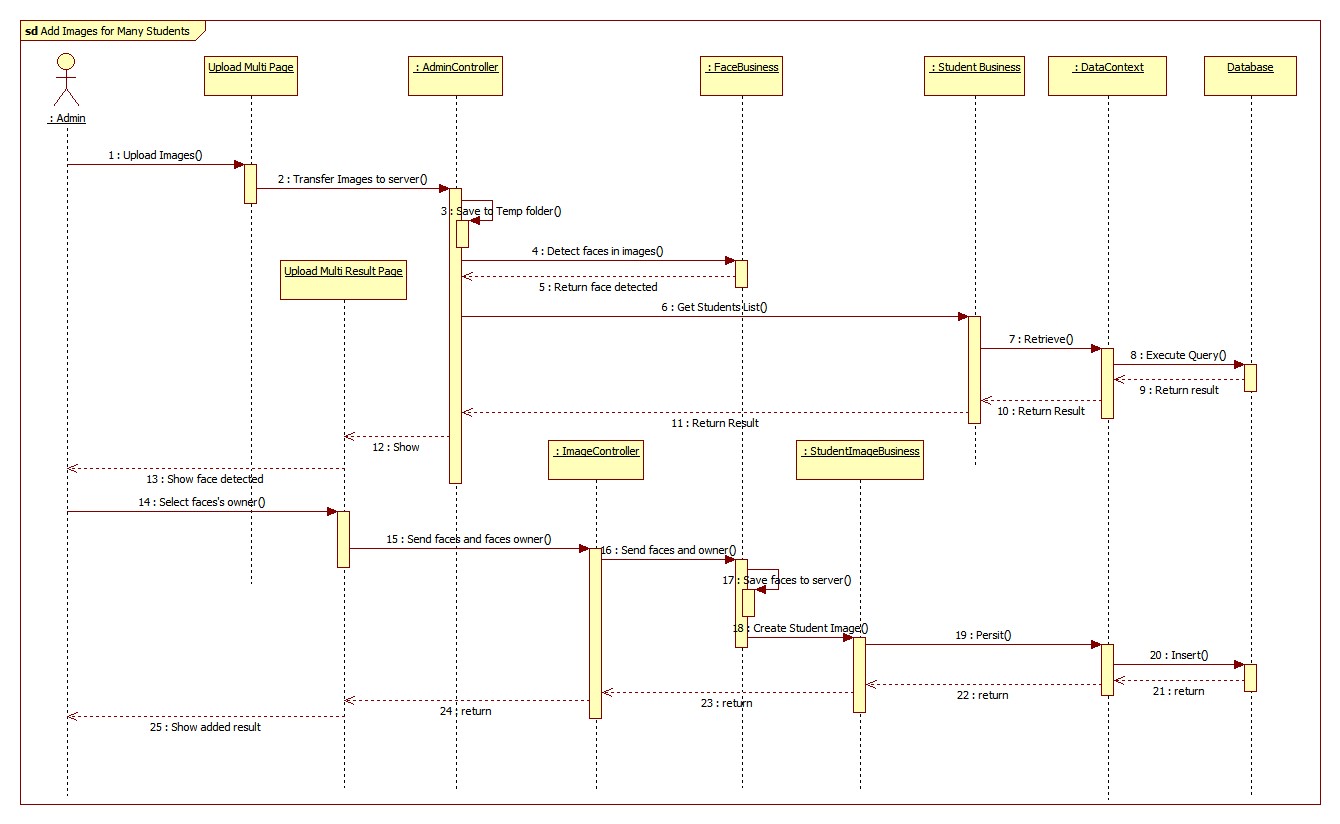


Figure 53: Add Image for Many Students Sequence Diagram

#### Delete Student Image

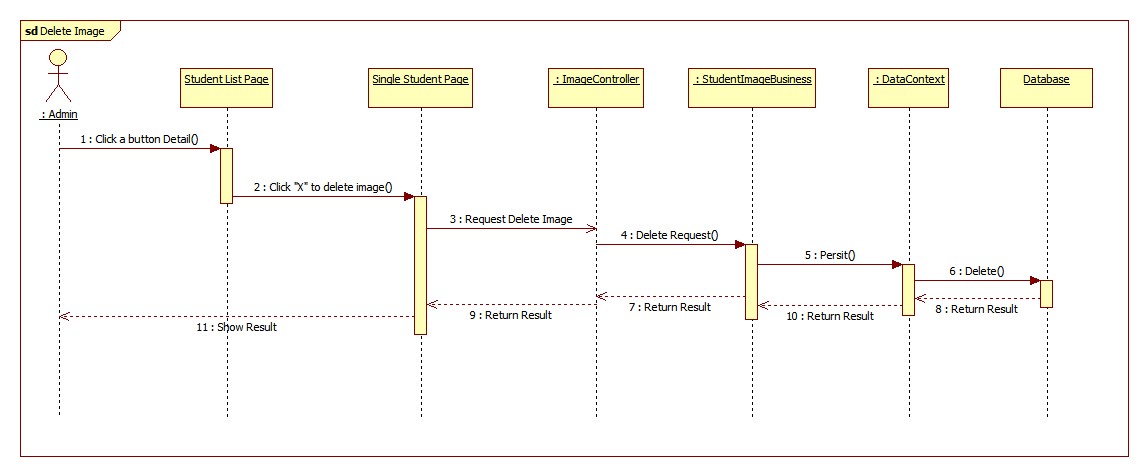


Figure 54: Delete Student Image Sequence Diagram

#### Create Account

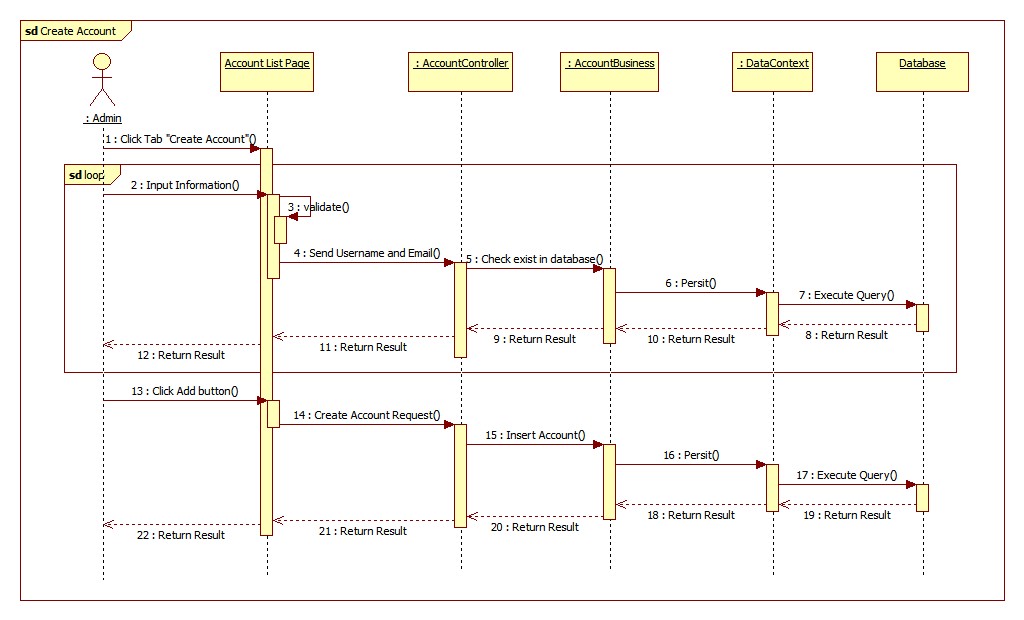


Figure 55: Create Account Sequence Diagram

#### Configure System

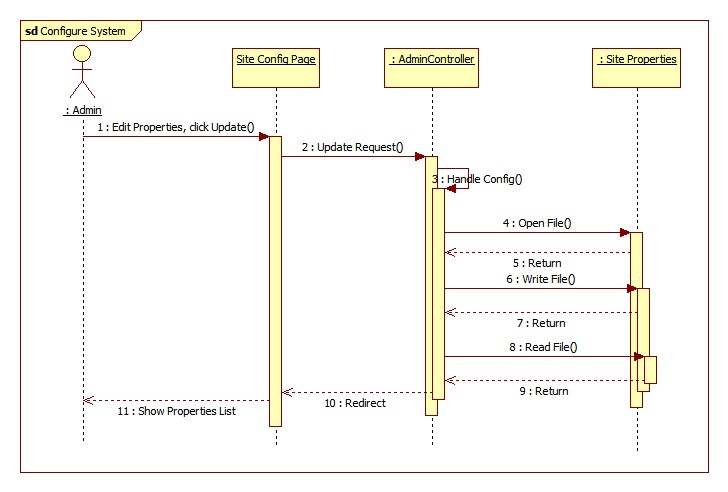


Figure 56: Configure System Sequence Diagram

#### Face Detection

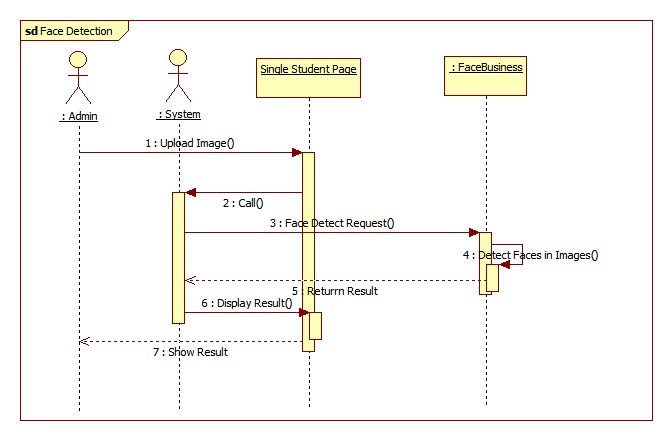


Figure 57: Face Detection Sequence Diagram

#### Face Recognition

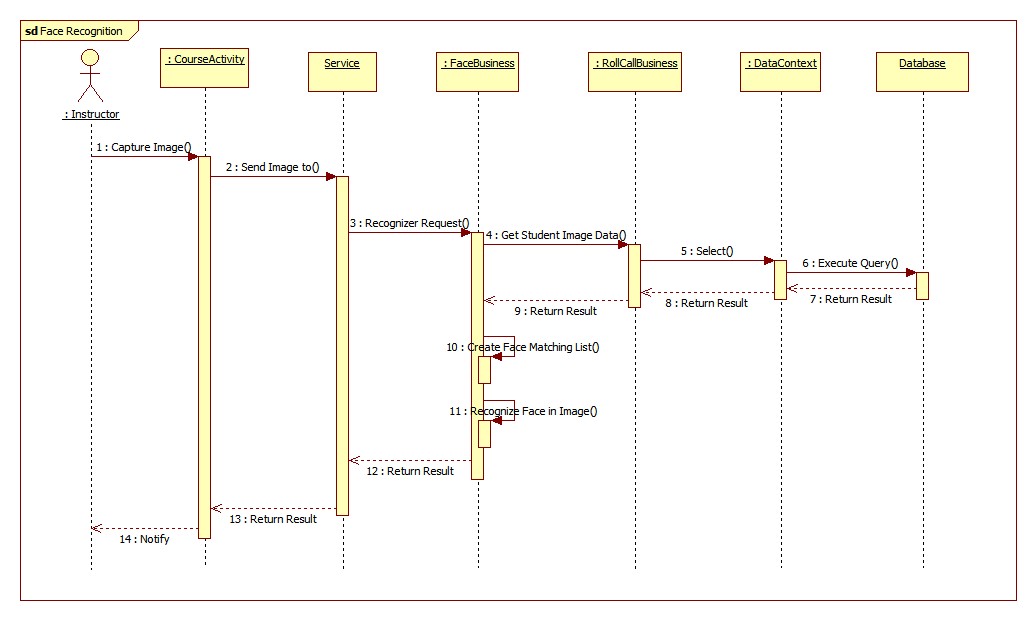


Figure 58: Face Recognition Sequence Diagram

#### Auto Free Storage Space

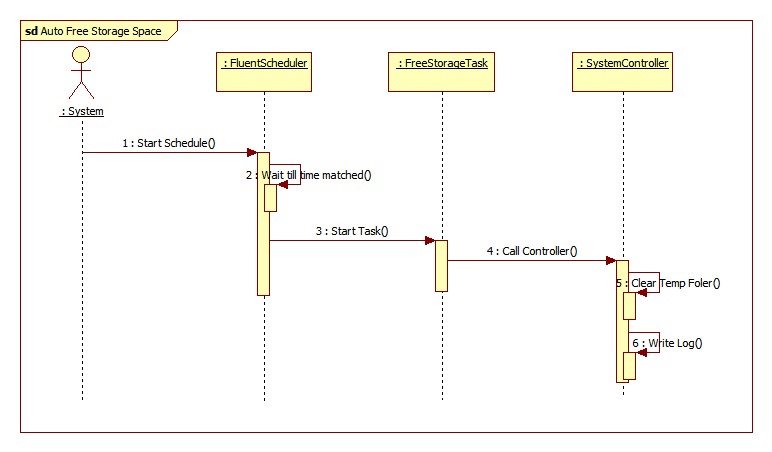


Figure 59: Auto Free Storage Space Sequence Diagram

#### Auto Active Roll Call

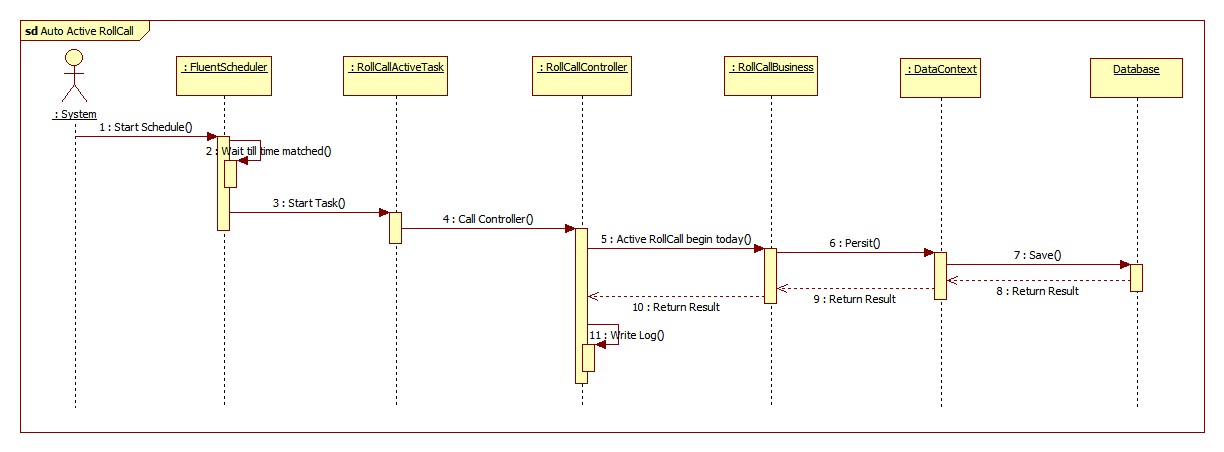


Figure 60: Auto Active Roll Call Sequence Diagram

#### Add Roll Call

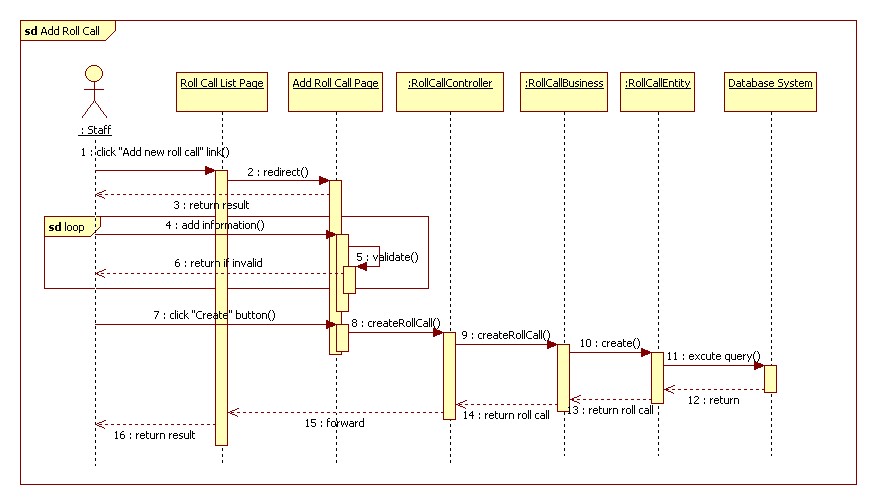
****

Figure 61: Add Roll Call Sequence Diagram

#### Edit Roll Call

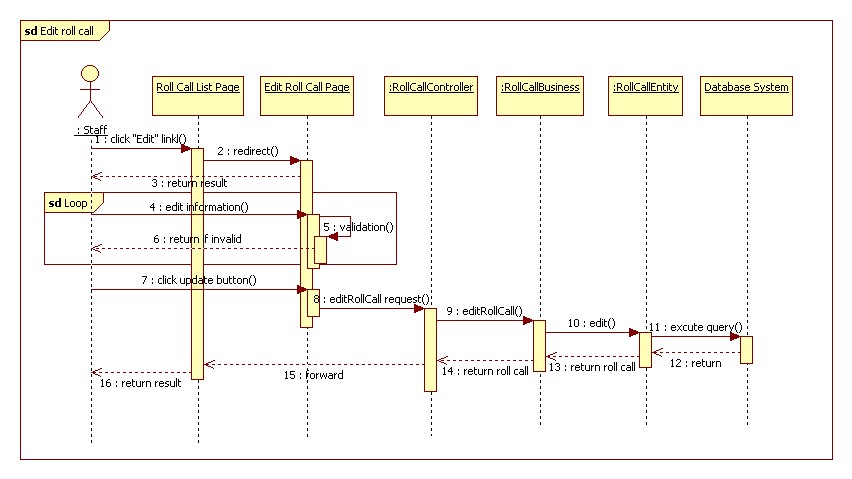
****

Figure 62: Edit Roll Call Sequence Diagram

#### Import Student List

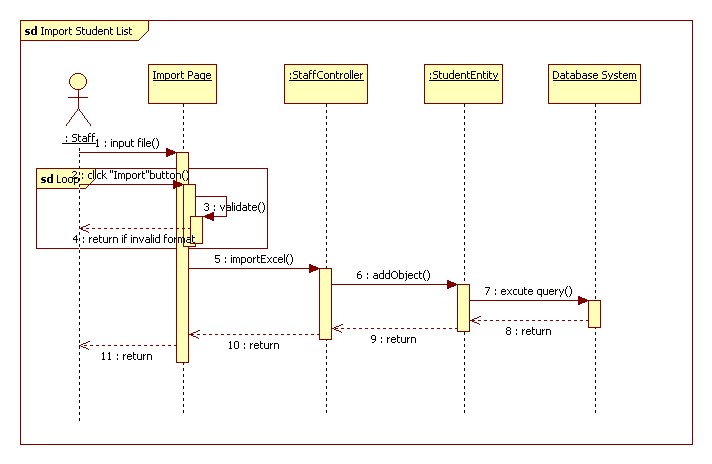
****

Figure 63: Import Student List Sequence Diagram

#### Export Report

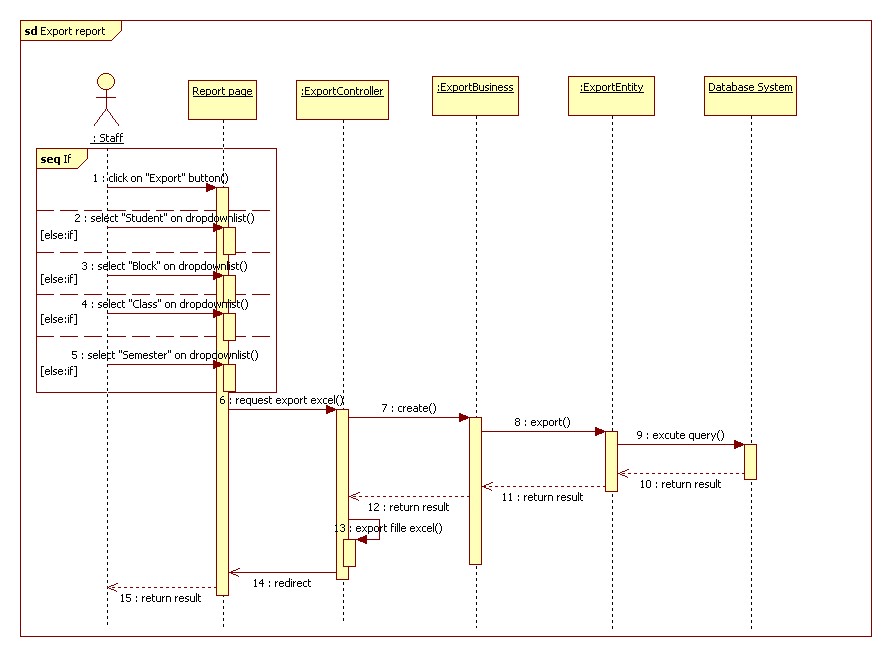
****

Figure 64: Export Report Sequence Diagram

#### Add Student

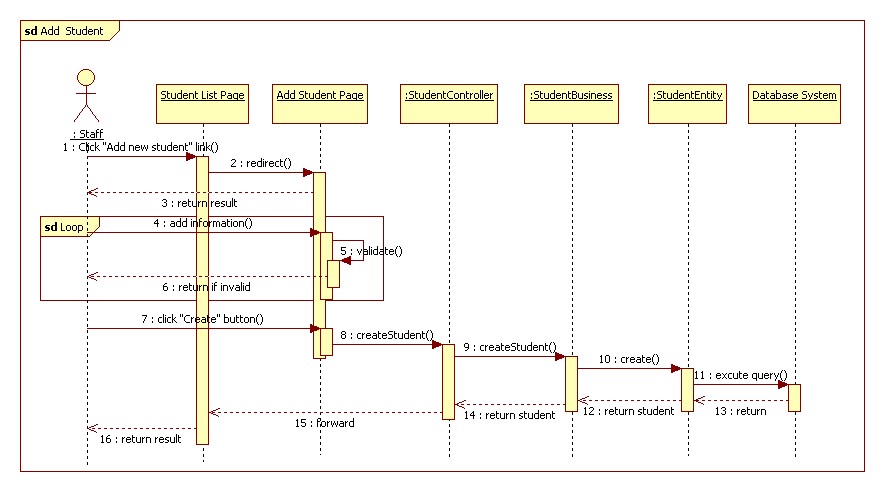
****

Figure 65: Add Student Sequence Diagram

#### Add Class

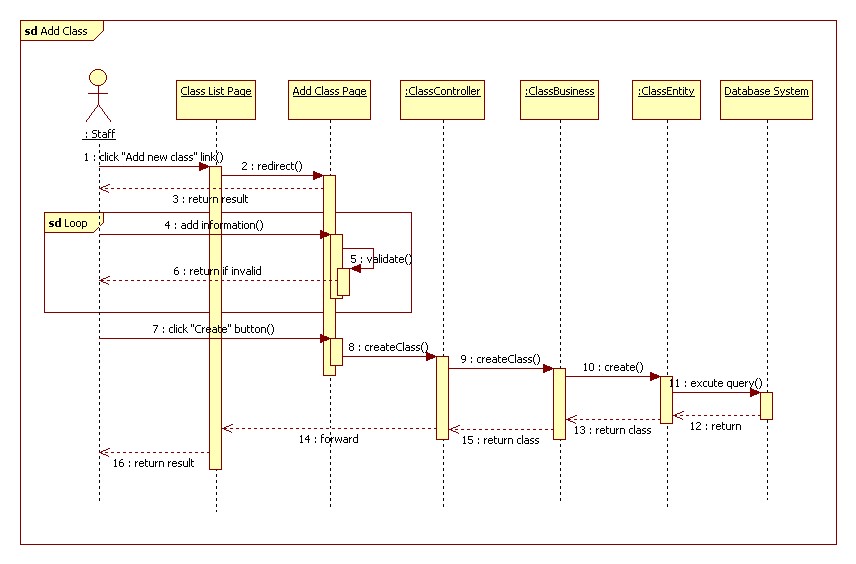
****

Figure 66: Add Class Sequence Diagram

#### Edit Class

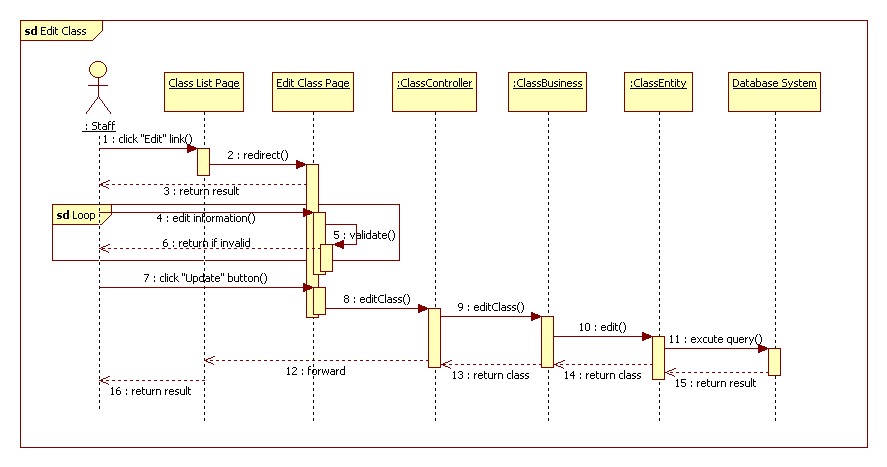
****

Figure 67:Edit Class Sequence Diagram

#### Add Subject

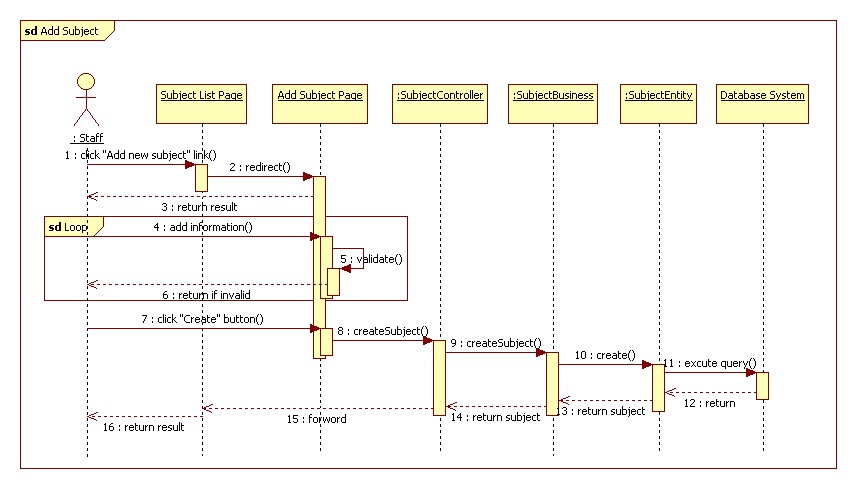
****

Figure 68: Add Subject Sequence Diagram

#### Edit Subject

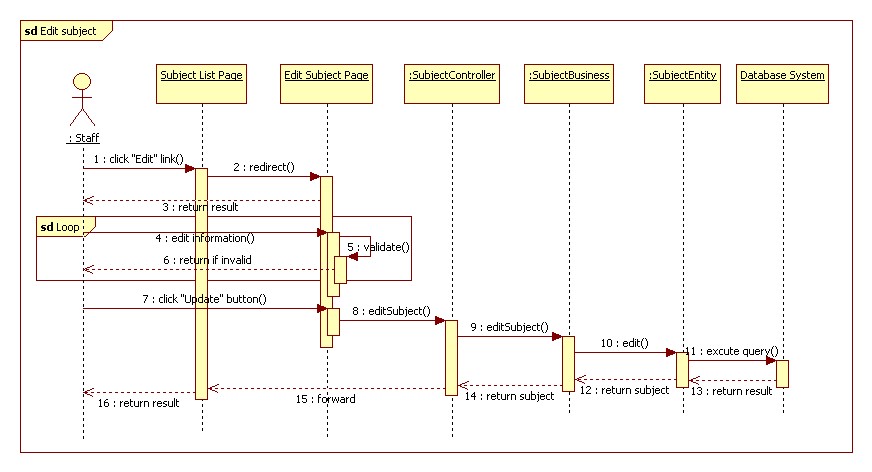
****

Figure 69: Edit Subject Sequence Diagram

#### Take Attendance Auto

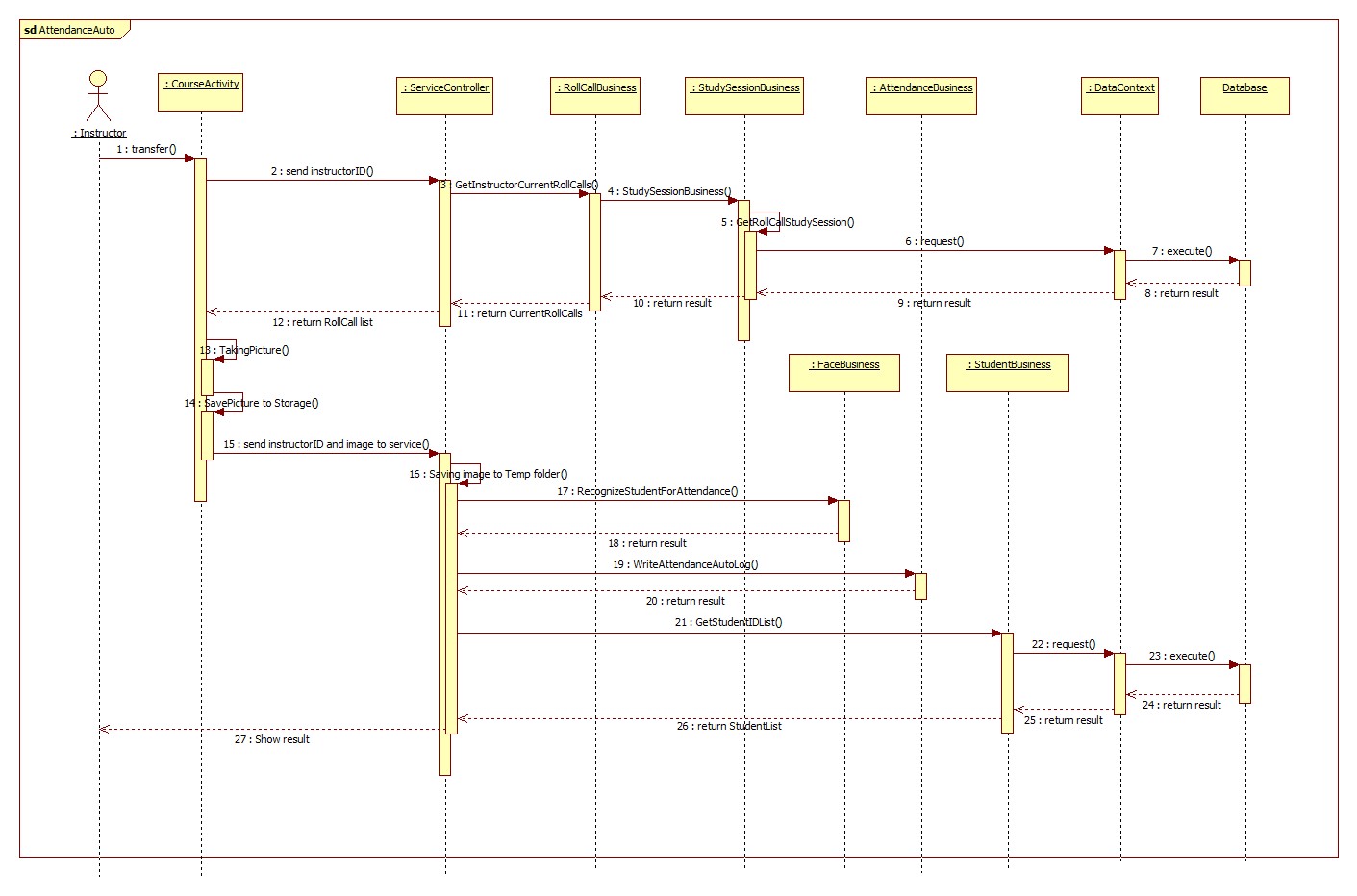


Figure 70: Take Attendance Auto Sequence Diagram

#### Take Attendance Manual

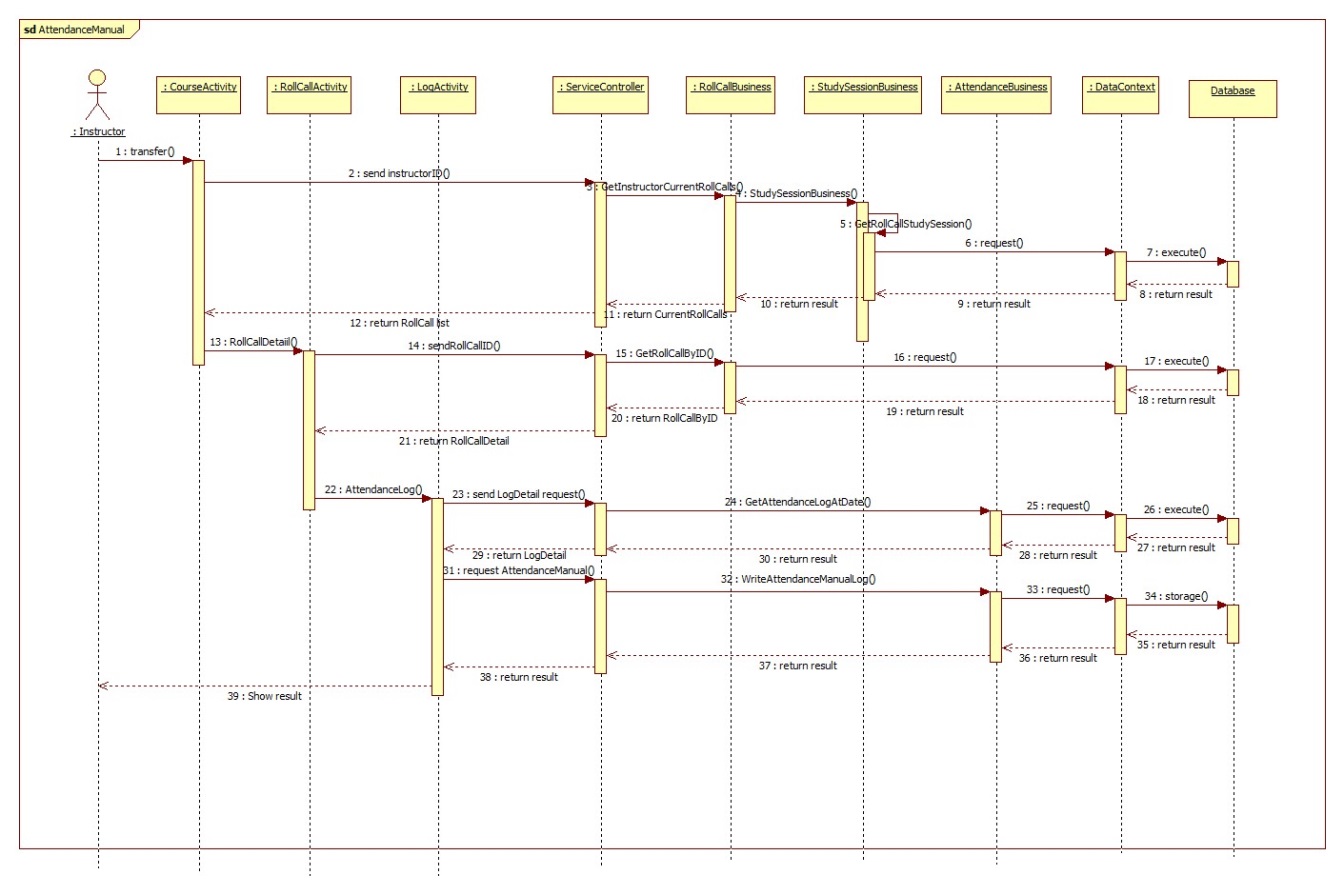


Figure 71: Take Attendance Manual Sequence Diagram

#### View Roll Call Detail

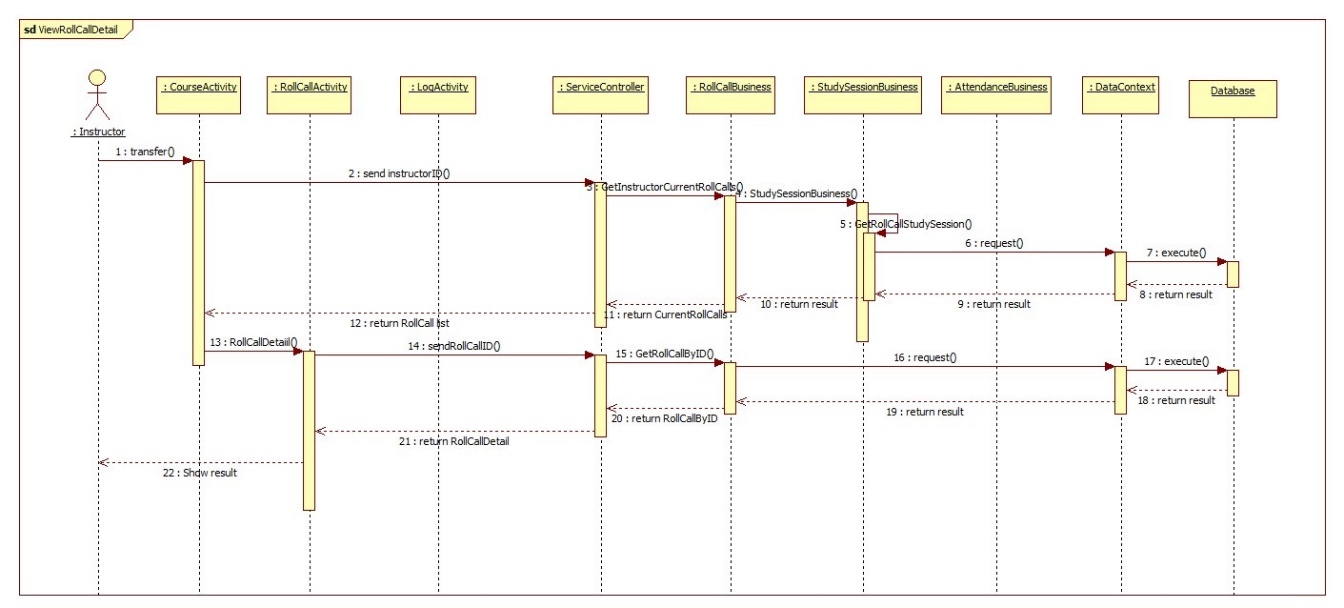


Figure 72: View Roll Call Detail Sequence Diagram

#### Report Rollcall by Class

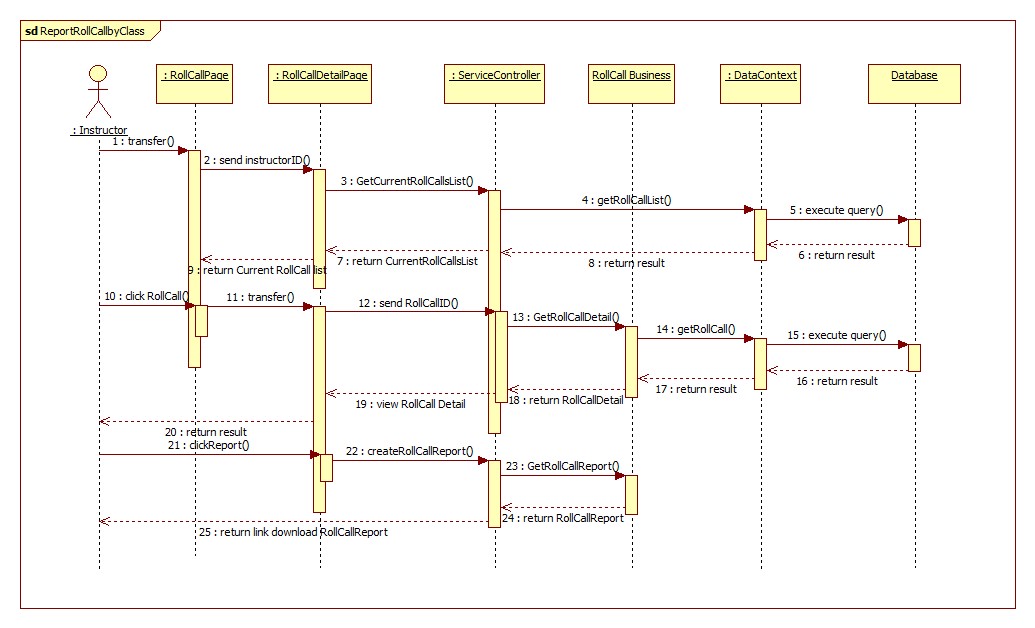


Figure 73: Report Attendance by Class Sequence Diagram

#### Check Present Rate

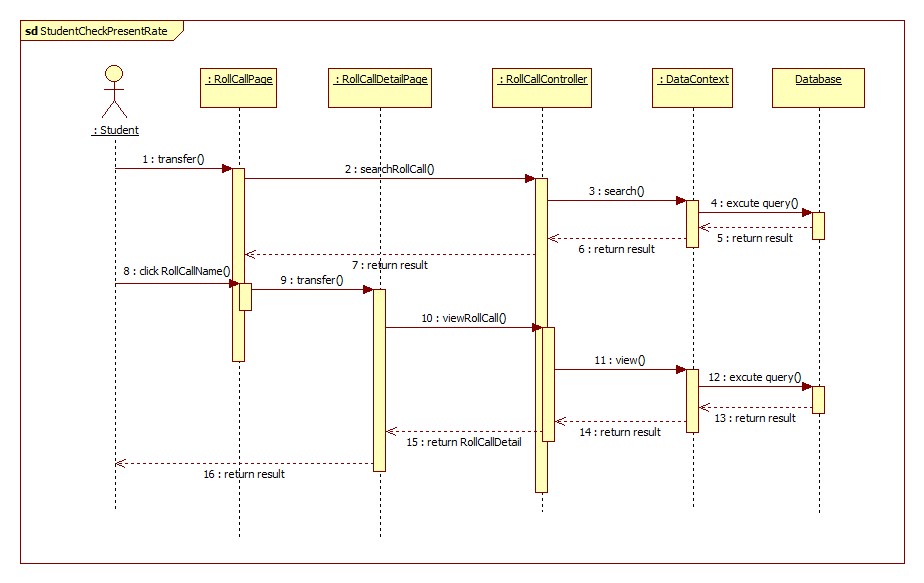


Figure 74: Check Present Rate Sequence Diagram

## User Interface Design

### Instructor Interface Design

#### Instructor Home Page



Figure 75: Instructor Home Page

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Date Type** | **Length** |
| 6 | Roll Call Info | Info of a roll call: Subject, class, time…. | Y | Y | Label | N/A | N/A |

**Button/ Hyperlink**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Available Citeria** | **Validation** | **Outcome** |
| 1 | Account Info | Username of user | Always | No | Account Info |
| 2 | Home Page | Click to go to home page | Always | No | Home Page |
| 3 | Instructor Home | Click to go to home page | Always | No | Home Page |
| 4 | Teaching Calendar | Click to View Teaching Calendar | Always | No | Teaching Calendar Page |
| 5 | Present Roll Call | Click to view the roll call at present | Always | No | Present Roll Call Page |

#### Instructor – Teaching Calendar

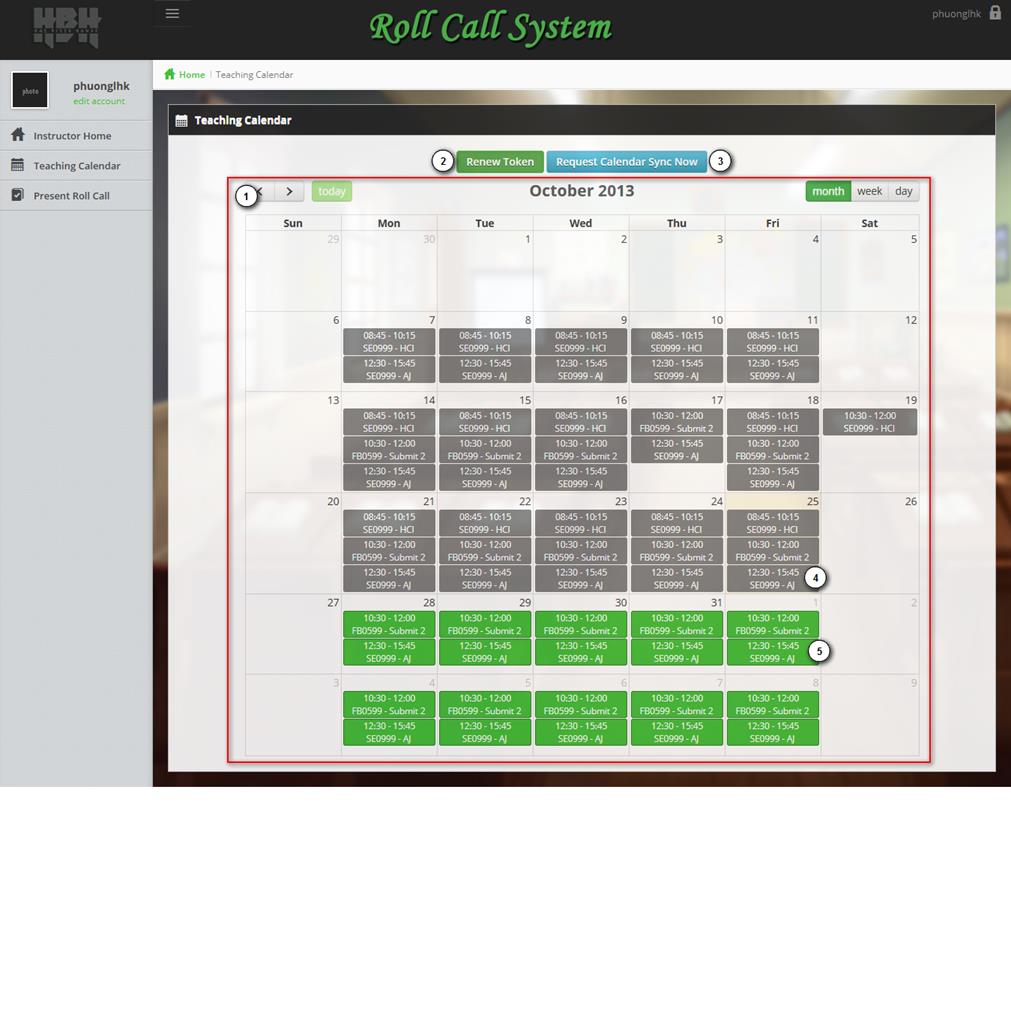


Figure 76: Instructor – Teaching Calendar

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Date Type** | **Length** |
| 1 | Traching Calendar | The teaching calendar of instructor | Y | Y | Calendar | N/A | N/A |
| 4 | Past event | Event of the days before today | Y | N | Label | N/A | N/A |
| 5 | Future Event | Event of the days after today | Y | N | Label | N/A | N/A |

**Button/ Hyperlink**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Available Citeria** | **Validation** | **Outcome** |
| 2 | Renew Token | Click to get a new token from Google Calendar | Always | No | Get Token from Google |
| 3 | Request Calendar Sync now | Click to go to sync the calendar to Google Calendar | Always | No | Calendar is synced |

#### Instructor – Present Roll Call

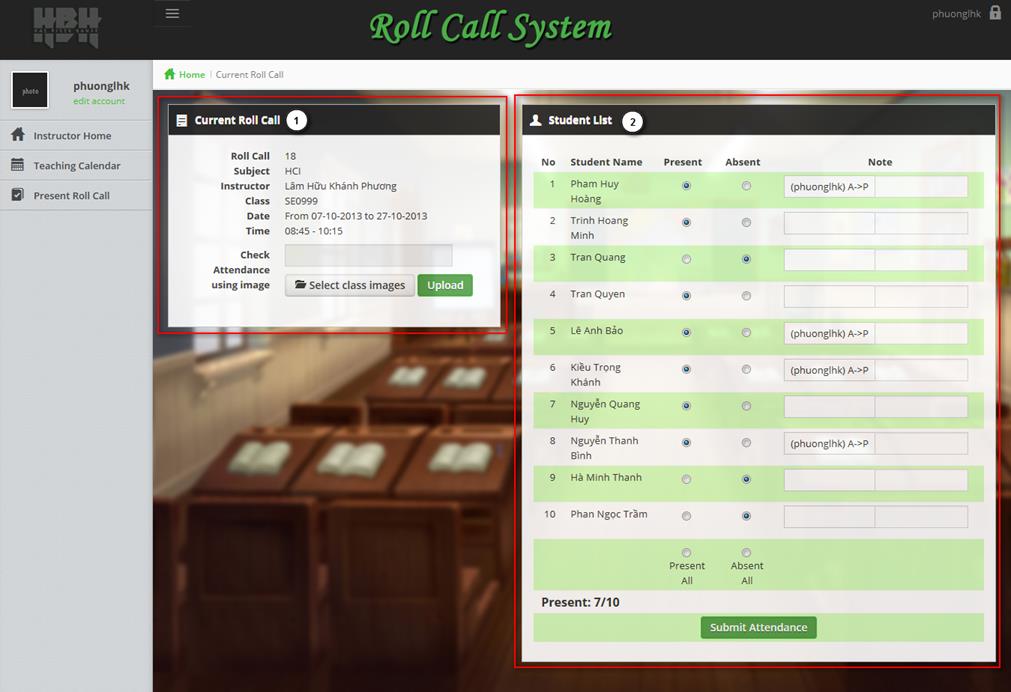


Figure 77: Instructor – Present Roll Call

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Date Type** | **Length** |
| 1 | Current Roll Call | The info of current roll call | Y | Y | N/A | N/A | N/A |
| 2 | Student List Table | The list of student and their attendance of current roll call | N | Y | Table | N/A | N/A |

**Current Roll Call includes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Read only** | **Control Type** | **Data Type** | **Length** |
| Roll Call | ID of the roll call | Y | Label | Interger | N/A |
| Subject | Subject of the roll call | Y | Label | String | N/A |
| Instructor | Instructor of the roll call | Y | Label | String | N/A |
| Class | Class of the roll call | Y | Label | String | N/A |
| Date | Begin date and end date of the roll call | Y | Label | String | N/A |
| Time | Begin time and end time and end time | Y | Label | String | N/A |
| Select Images | Upload image for check attendance | N | File Upload | N/A | N/A |
| Upload | Upload button | N | Button | N/A | N/A |

**Table Student List includes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Read only** | **Control Type** | **Data Type** | **Length** |
| No | No | Y | Label | Interger | N/A |
| Student Name | Full Name of the student | Y | Label | String | N/A |
| Present | Check if the student is present | N | Radio Button | N/A | N/A |
| Absent | Check if the student is absent | N | Radio Button | N/A | N/A |
| Note | Note of instructor | N | Textbox | String | N/A |
| Submit Attendance | Click to submit the re-check attendance | N | Button | N/A | N/A |

#### Instructor – Roll Call Detail

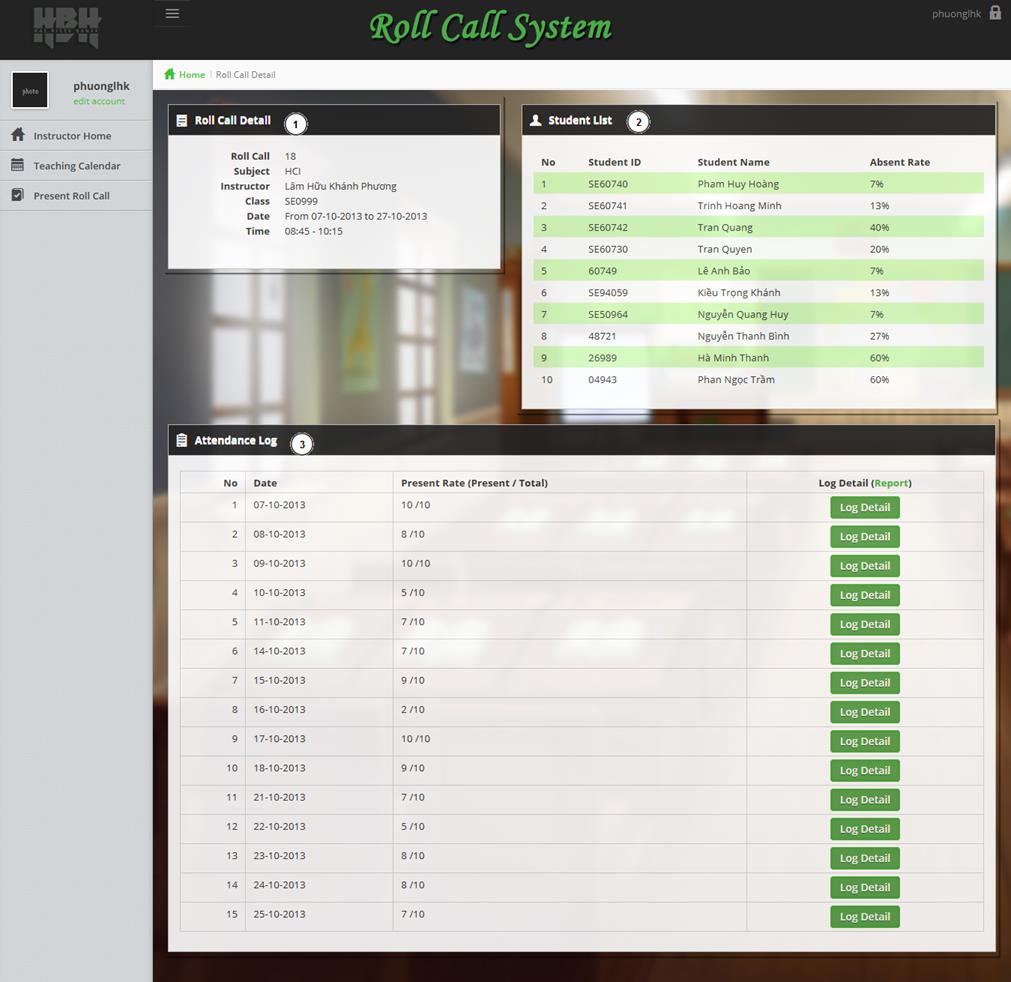


Figure 78: Instructor – Roll Call Detail

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Date Type** | **Length** |
| 1 | Current Roll Call | The info of the roll call | Y | Y | N/A | N/A | N/A |
| 2 | Student List Table | The list of student and their attendance of the roll call | Y | Y | Table | N/A | N/A |
| 3 | Attendance Log Table | Every log of the roll call | N | Y | Table | N/A | N/A |

**Table Current Roll Call includes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Read only** | **Control Type** | **Data Type** | **Length** |
| Roll Call | ID of the roll call | Y | Label | Interger | N/A |
| Subject | Subject of the roll call | Y | Label | String | N/A |
| Instructor | Instructor of the roll call | Y | Label | String | N/A |
| Class | Class of the roll call | Y | Label | String | N/A |
| Date | Begin date and end date of the roll call | Y | Label | String | N/A |
| Time | Begin time and end time and end time | Y | Label | String | N/A |

**Table Student List includes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Read only** | **Control Type** | **Data Type** | **Length** |
| No | No | Y | Label | Integer | N/A |
| Student Code | Code of the student | Y | Label | String | 7 |
| Student Name | Full Name of the student | Y | Label | String | N/A |
| Absent Rate | The absent rate of the student | Y | Label | Integer | N/A |

**Table Attendance Log includes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Read only** | **Control Type** | **Data Type** | **Length** |
| No | No | Y | Label | Integer | N/A |
| Date | The date of attendance log | Y | Label | Datetime | N/A |
| Present Rate | The present rate of attendance lig | Y | Label | String | N/A |
| Log Detail | Click to view the detail, or check attendance manually | Y | Button | N/A | N/A |

#### Instructor – Check Attendance Manual

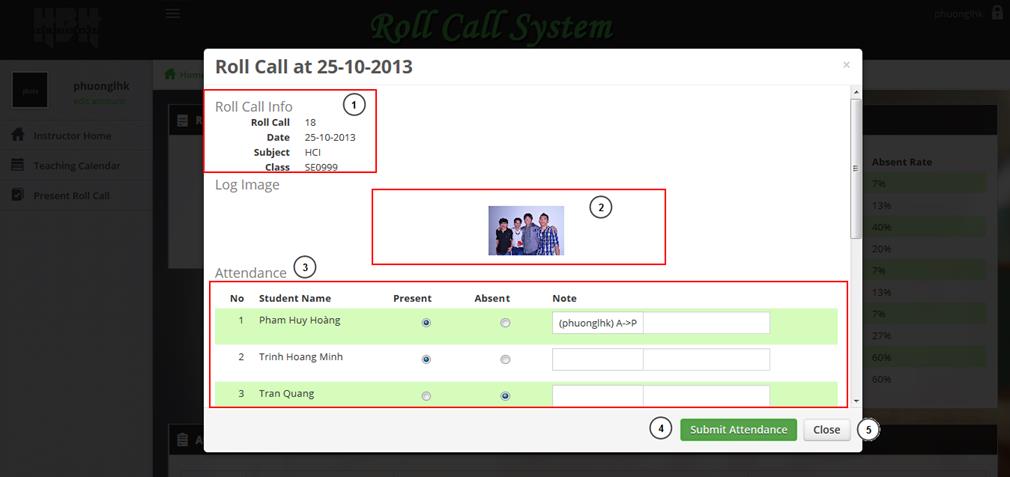


Figure 79: Instructor – Check Attendance Manual

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Date Type** | **Length** |
| 1 | Roll Call Info | The info of the roll call | Y | Y | N/A | N/A | N/A |
| 2 | Log Image | The image of the log | Y | Y | Image | N/A | N/A |
| 3 | Attendance | Attendance of student in the log | N | Y | Table | N/A | N/A |

**Button/ Hyperlink**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Available Citeria** | **Validation** | **Outcome** |
| 4 | Submit Attendance | Click to submit the attendance | Always | No | Attendance Submitted |
| 5 | Close | Close the log detail pop-up | Always | No | Pop-up is closed |

**Roll Call Info includes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Read only** | **Control Type** | **Data Type** | **Length** |
| Roll Call | ID of the roll call | Y | Label | Interger | N/A |
| Subject | Subject of the roll call | Y | Label | String | N/A |
| Class | Class of the roll call | Y | Label | String | N/A |
| Date | The date of the log | Y | Label | String | N/A |

**Table Student List includes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Read only** | **Control Type** | **Data Type** | **Length** |
| No | No | Y | Label | Interger | N/A |
| Student Name | Full Name of the student | Y | Label | String | N/A |
| Present | Check if the student is present | N | Radio Button | N/A | N/A |
| Absent | Check if the student is absent | N | Radio Button | N/A | N/A |
| Note | Note of instructor | N | Textbox | String | N/A |

### Admin Interface Design

#### Admin Home Page

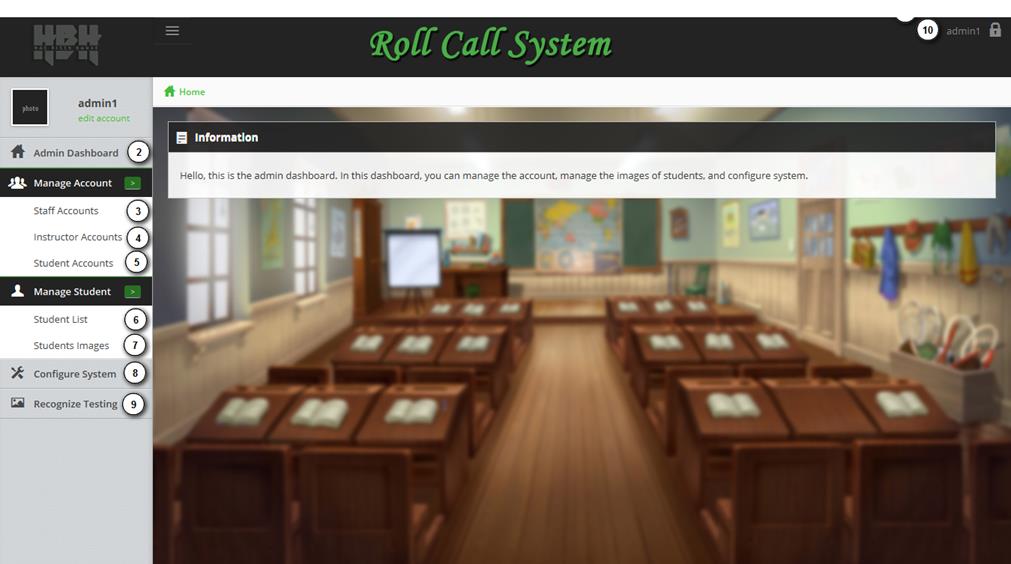


Figure 80: Admin Home Page

**Button/ Hyperlink**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Available Citeria** | **Validation** | **Outcome** |
| 1 | Account Info | Username of user | Always | No | Account Info |
| 2 | Admin Dashboard | Click to go to admin home page | Always | No | Home Page |
| 3 | Staff Accounts | Click to go to staff accounts page | Always | No | Staff Accounts Page |
| 4 | Instructor Accounts | Click to go to instructor accounts page | Always | No | Instructor Accounts Page |
| 5 | Student Accounts | Click to go to student accounts page | Always | No | Student Accounts Page |
| 6 | Student List | Click to go to student list page | Always | No | Student List Page |
| 7 | Student Images | Click to select images for students | Always | No | Image Multi Upload Page |
| 8 | Configure System | Click to config the system | Always | No | Config Page |
| 9 | Recognize Testing | Click to test the face recognize | Always | No | Recognizer Testing Page |

#### Admin – Student List

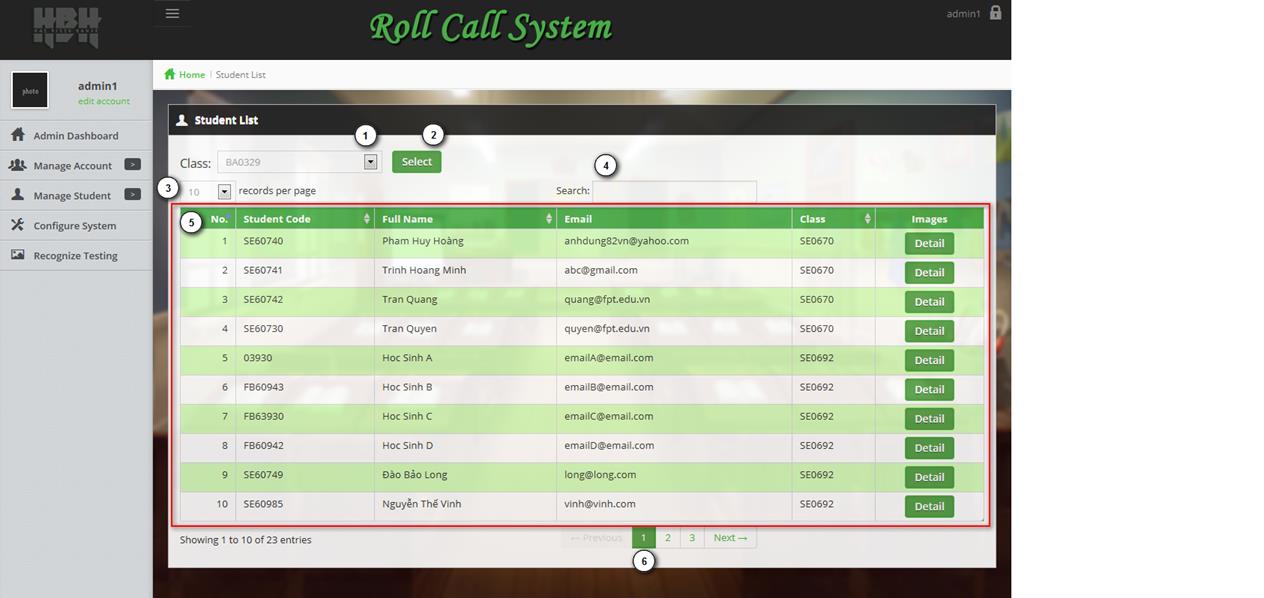


Figure 81: Admin – Student List

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Date Type** | **Length** |
| 1 | Class | Filter student by class | N | Y | Dropdowlist | String | N/A |
| 3 | Number of record | Number of record per page | N | Y | Dropdownlist | Integer | N/A |
| 4 | Search | Search textbox to find student | N | Y | Textbox | String | N/A |
| 5 | Student List | Table showing info of students | Y | Y | Table | N/A | N/A |

**Button/ Hyperlink**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Available Citeria** | **Validation** | **Outcome** |
| 2 | Select | Select the class to filter | Always | No | Table show students of class |
| 6 | Paging | Paging area of the student table | Always | No | Home Page |

**Table Student List includes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Description** | **Read only** | **Control Type** | **Data Type** | **Length** |
| No | No | Y | Label | Integer | N/A |
| Student Code | Code of the student | Y | Label | String | 7 |
| Student Name | Full Name of the student | Y | Label | String | N/A |
| Email | Email of the student | Y | Label | String | N/A |
| Class | Class of the sutdent | Y | Label | String | N/A |
| Detail | Click to view image data of a student | N | Button | N/A | N/A |

#### Admin – Single Student Info

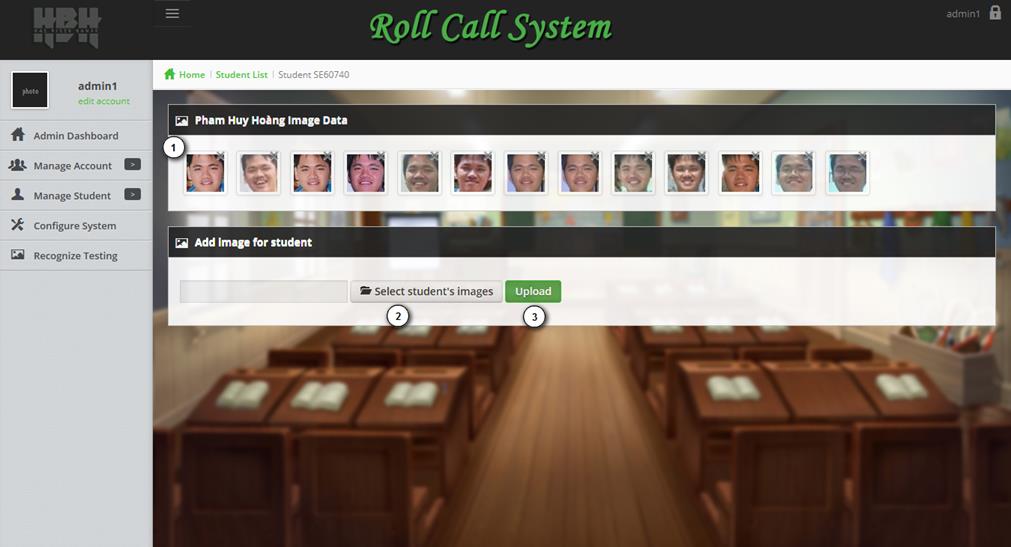


Figure 82: Admin – Single Student Info

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Date Type** | **Length** |
| 1 | Image Date | Image Data of a student. Click the X button to delete a image | N | N | N/A | N/A | N/A |
| 2 | Select Student Image | Select the image of the student | N | Y | File Upload | Image | N/A |

**Button/ Hyperlink**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Available Citeria** | **Validation** | **Outcome** |
| 3 | Upload Button | Click to upload the images | Always | No | Show Upload Result |

#### Admin – Select Images for Students

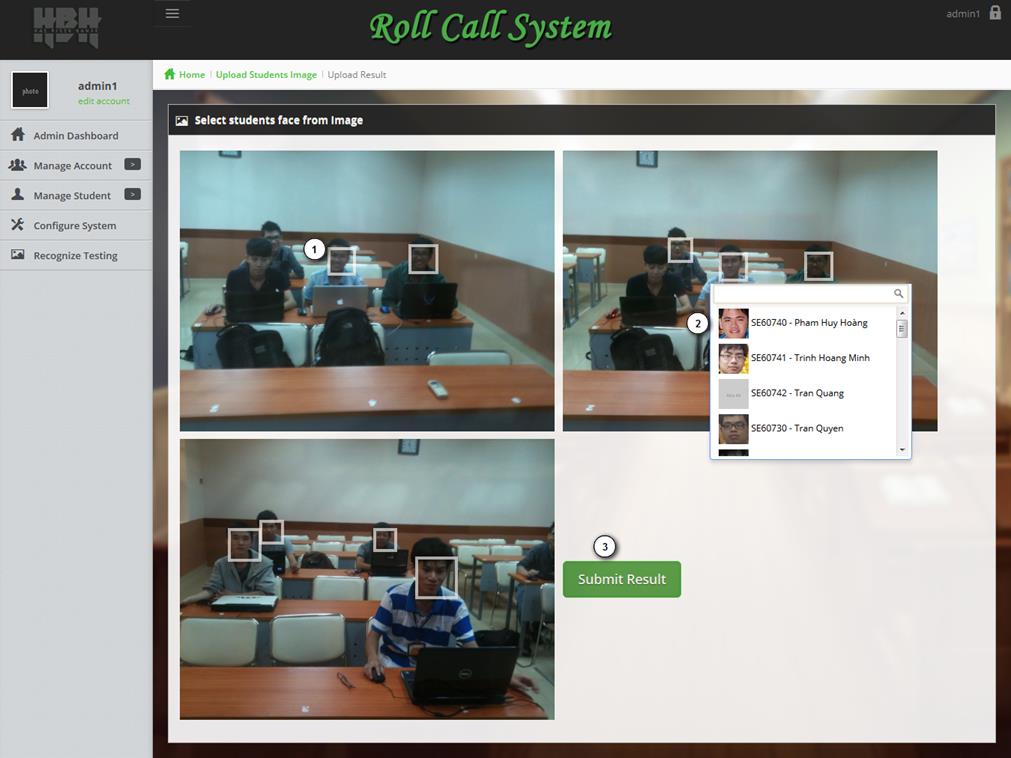


Figure 83: Admin – Select Images for Students

**Fields**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Field Name** | **Description** | **Read only** | **Mandatory** | **Control Type** | **Date Type** | **Length** |
| 1 | Face Region | Face Region in a picture. When hover, a dropdownlist will be show | Y | Y | N/A | N/A | N/A |
| 2 | Student List | A dropdownlist to select face’s owner. Contain student image, code and full name | N | Y | Dropdownlist | N/A | N/A |

**Button/ Hyperlink**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Function** | **Description** | **Available Citeria** | **Validation** | **Outcome** |
| 3 | Submit Result | Click to submit and save the result | Always | No | Back to Image Data Page |

## Database Design

### Logical Diagram

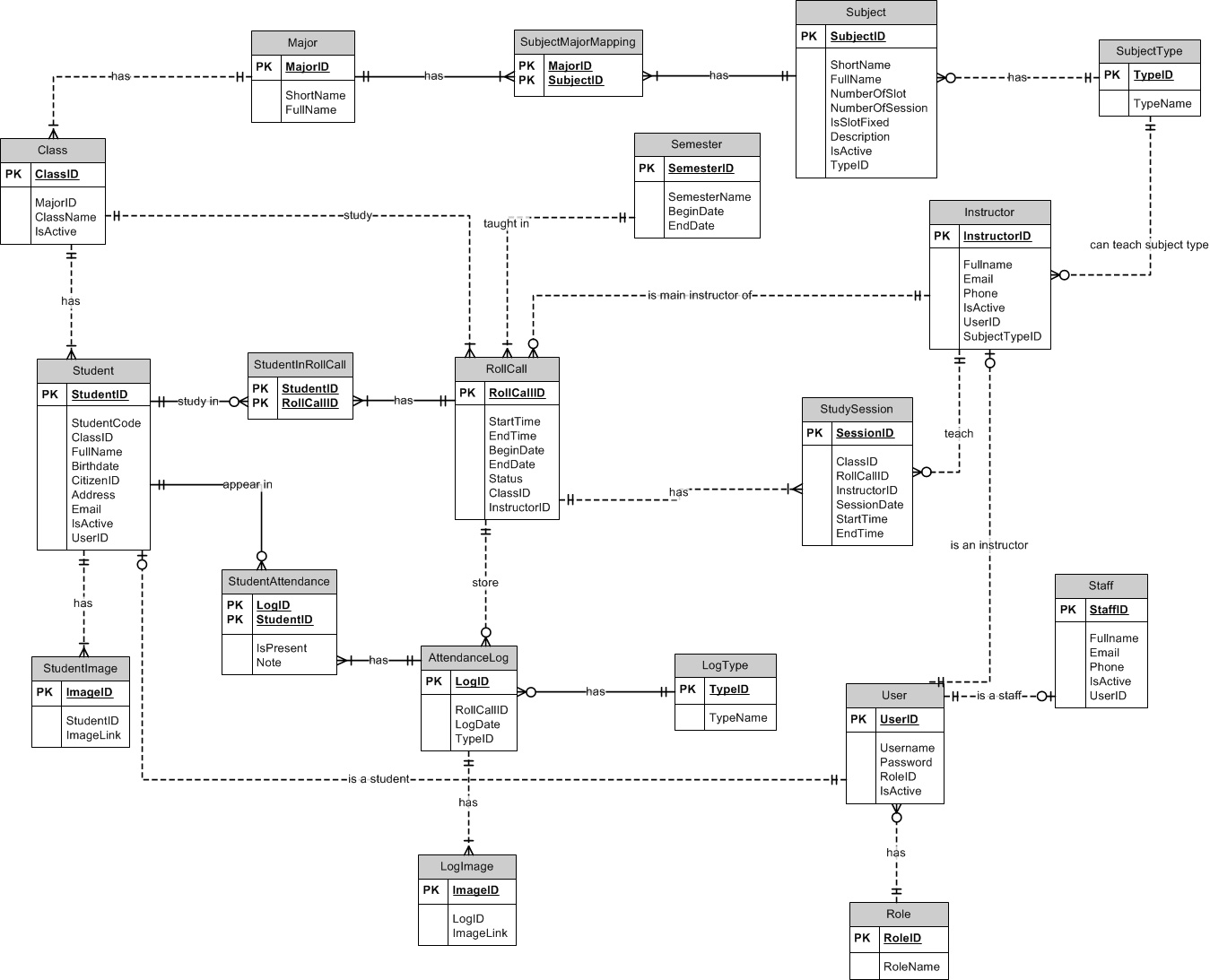


Figure 84: Logical Diagram

### Data Dictionary

|  |  |
| --- | --- |
| **Entity Data dictionary: describe content of all entities** | |
| **Entity Name** | **Description** |
| Semester | Describe all semesters in school system. |
| Major | Describe all majors in school system. |
| SubjectMajorMapping | Describe relationship between table subject and table major. One subject belongs to one or a lot of major, one major has one or a lot of subject. |
| Subject | Describe all subjects in school system. |
| SubjectType | Describe type of subject (IT, economic, language ...) |
| Class | Describe all class in school system. |
| Student | Describe all students in school system. |
| StudentImage | Describe all images of each student. |
| Instructor | Describe all instructors in school system. |
| StudySession | Describe all study session of roll call. |
| Rollcall | Describe all roll calls in school system. |
| StudentInRollcall | Describe student list of roll call. |
| StudentAttendance | Describe attendance of student. |
| AttendanceLog | Describe all attendance in roll call |
| LogImage | Describe all images of attendance log |
| LogType | Describe all type of attendance log |
| Staff | Describe all staffs in school system. |
| User | Describe account of user of system. User include: admin, staff, guest, student … |
| Role | Describe role of user. One user has one role. |

Table 50: Entity Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entity name** | **Attributes** | **Description** | **Domain** | **Nulls** |
| Semester | SemesterID{PK}  SemesterName  BeginDate  EndDate | Uniquely identifies a semester, auto increment.  Name of semester.  The begin date of semester.  The end date of semester. | Int  15 nvar  Datetime  Datetime | No  No  No  No |
| Major | MajorID{PK}  ShortName  FullName | Uniquely identifies a major, auto increment.  Short name of major.  Full name of major. | Int  2 nvar  30 nvar | No  Yes  No |
| SubjectMajorMapping | MajorID {FK}  SubjectID {FK} | Foreign key references to table Major  Foreign key references to table Subject | Int  Int | No  No |
| Subject | SubjectID{PK}  ShortName  FullName  NumberOfSlot  NumberOfSession  IsSlotFixed  Description  IsActive  TypeID {FK} | Uniquely identifies a subject, auto increment.  Short name of subject.  Full name of subject.  Number of slot of subject (per day).  Number of session of subject.  This value shows that a slot can fixed or not.  Description of subject  This value shows that the subject is active or not.  Foreign key references to table Type | Int  10 nvar  50 nvar  Int  Int  Bit  10 char  Bit  Int | No  Yes  No  No  No  No  Yes  No  No |
| SubjectType | TypeID {PK}  TypeName | Uniquely identifies a type, auto increment.  Name of type. | Int  50 nvar | No  No |
| Class | ClassID {PK}  MajorID {FK}  ClassName  IsActive | Uniquely identifies a class, auto increment.  Foreign key references to table Major  Name of class  This value shows that the class is active or not. | Int  Int  6 char  Bit | No  No  No  No |
| Student | StudentID {PK}  ClassID {FK}  FullName  Birthdate  CitizenID  Address  Email  IsActive  UserID {FK}  StudentCode | Uniquely identifies a student, auto increment.  Foreign key references to table Class.  Full name of student.  Birthdate of student.  CitizenID of student  Address of student  Email of student.  This value shows that student is active or not.  Foreign key references to table User  Student code of student. | Int  Int  50 nvar  Datetime  8 char  100 nvar  50 nvar  Bit  Int  7 char | No  No  No  No  No  Yes  Yes  No  Yes  No |
| StudentImage | ImageID {PK}  StudentID {FK}  ImageLink | Uniquely indentifies of image, auto increment.  Foreign key references to table Student  The link of image. | Int  Int  100 var | No  No  No |
| Instructor | InstructorID {PK}  FullName  Email  Phone  IsActive  UserID {FK}  SubjectTypeID {FK} | Uniquely identifies of instructor, auto increment.  Full name of instructor.  Email of instructor.  Phone of instructor.  This value shows the instructor is active or not.  Foreign key references to table User.  Foreign key references to table SubjectType. The instructor can only teach subject with this type | Int  50 nvar  50 nvar  12 nvar  Bit  Int  Int | No  No  Yes  Yes  No  Yes  No |
| StudySession | SessionID {PK}  RollcallID {FK}  InstructorID {FK}  SessionDate  StartTime  EndTime  ClassID {FK}  Note | Uniquely identifies of session, auto increment.  Foreign key references to table Rollcall.  Foreign key references to table Instructor.  The date of session.  The time session start.  The time session end.  Foreign key references to table Class.  Note of session. | Int  Int  Int  Date  7 time  7 time  Int  50 nvar | No  No  No  No  No  No  No  Yes |
| Rollcall | RollcallID {PK}  StartTime  EndTime  BeginDate  EndDate  SubjectID {FK}  ClassID {FK}  SemesterID {FK}  Status  InstructorID {FK} | Uniquely identifies of session, auto increment.  The time roll call start.  The time roll call end.  The date roll call begins.  The date roll call end.  Foreign key references to table Subject.  Foreign key references to table Class.  Foreign key references to table Semester.  Status of roll call. (0: created. 1: active. 2: unactive)  Foreign key references to table Instructor. The main instructor of the roll call | Int  7 time  7 time  Date  Date  Int  Int  Int  Int  Int | No  No  No  No  No  No  No  No  No  No |
| StudentInRollcall | StudentID {FK}  RollcallID {FK} | Foreign key references to table Student.  Foreign key references to table Roll Call. | Int  Int | No  No |
| StudentAttendance | LogID {FK}  StudentID {FK}  IsPresent  Note | Foreign key references to table AttendanceLog.  Foreign key references to table Student.  This value shows that student is present or not.  Note of student attendance. | Int  Int  Bit  50 nvar | No  No  No  Yes |
| AttendanceLog | LogID {PK}  RollcallID {FK}  LogDate  TypeID {FK} | Uniquely identifies of attendance, auto increment.  Foreign key references to table Rollcall.  The date of log.  Foreign key references to table LogType. | Int  Int  Date  Int | No  No  No  No |
| LogImage | ImageID {PK}  LogID {FK}  ImageLink | Uniquely identifies of image, auto increment.  Foreign key references to table AttendanceLog.  The link of image. | Int  Int  100 var | No  No  No |
| LogType | TypeID {PK}  TypeName | Uniquely identifies of log type, auto increment.  Name of type. | Int  10 nvar | No  Yes |
| Staff | StaffID {PK}  FullName  Email  Phone  IsActive  UserID {FK} | Uniquely identifies of staff, auto increment.  Full name of staff.  Email of staff.  Phone of staff.  This value shows that staff is active or not.  Foreign key references to table User. | Int  50 nvar  50 nvar  12 nvar  Bit  Int | No  No  Yes  Yes  No  Yes |
| User | UserID {PK}  UserName  Password  RoleID {FK}  IsActive | Uniquely identifies of user, auto increment.  Username of user.  Password of user.  Foreign key references to table Role.  This value shows that user is active or not. | Int  30 nvar  30 nvar  Int  Bit | No  No  No  No  No |
| Role | RoleID {PK}  RoleName | Uniquely identifies of role, auto increment.  Name of role. | Int  10 nvar | No  No |

Table 51: Attribute Data Dictionary

## Algorithms

### Face Detection

#### Definition

Face detection is a computer technology that determines the locations and sizes of human faces in arbitrary (digital) images. It detects facial features and ignores anything else, such as buildings, trees and bodies.

References: http://en.wikipedia.org/wiki/Face\_detection

#### Different method for face detection

There are many ways to detect a face in a scene - easier and harder ones. Here is a list of the most common approaches in face detection:

* Finding faces in images with controlled background
* Finding faces by color
* Finding faces by motion
* Using a mixture of the above
* Finding faces in unconstrained scenes:
* Neural Nets using statistical cluster information
* Neural Net approach
* Weak classifier cascades
* Model-based Face Tracking

References: http://www.facedetection.com/facedetection/techniques.htm

#### The method we chose

We chose the Viola-Jones method. Reason:

* Simple, fast and good accuracy.
* Already implemented in EmguCV.

**Method Introduction**

This approach to detecting objects in images combines four key concepts:

* Simple rectangular features, called Haar features
* An Integral Image for rapid feature detection
* The AdaBoost machine-learning method
* A cascaded classifier to combine many features efficiently

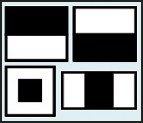


Figure 85: Example of Haar Future

The presence of a Haar feature is determined by subtracting the average dark-region pixel value from the average light-region pixel value. If the difference is above a threshold (set during learning), that feature is said to be present.



Figure 86: Apply Haar future to sub-window

To select the specific Haar features to use, and to set threshold levels, Viola and Jones use a machine-learning method called AdaBoost. AdaBoost combines many "weak" classifiers to create one "strong" classifier.

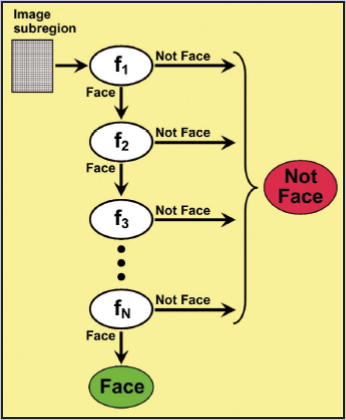


Figure 87: Classifier Cascade

During use, if any one of these filters fails to pass an image region, that region is immediately classified as "Not Face." When a filter passes an image region, it goes to the next filter in the chain. Image regions that pass through all filters in the chain are classified as "Face." Viola and Jones dubbed this filtering chain a cascade.

Reference: http://cognotics.com/opencv/servo\_2007\_series/part\_2/sidebar.html

The basic principle of the Viola-Jones face detection algorithm is to scan the detector many times through the same image – each time with a new size. Pseudo code for the algorithm:

1) Opens an image and transforms to greyscale if needed.

2) Runs a sub-window through the image.

2a) Rescales the content of the sub-window to 24\*24pixels (if needed). Check the Haar feature in each window. Move the sub-window if Haar feauture missing.

2b) Enlarges the sub-window by a given factor a goes back to 2).

2c) Continues until the sub-window size is equal to the least dimension of the image.

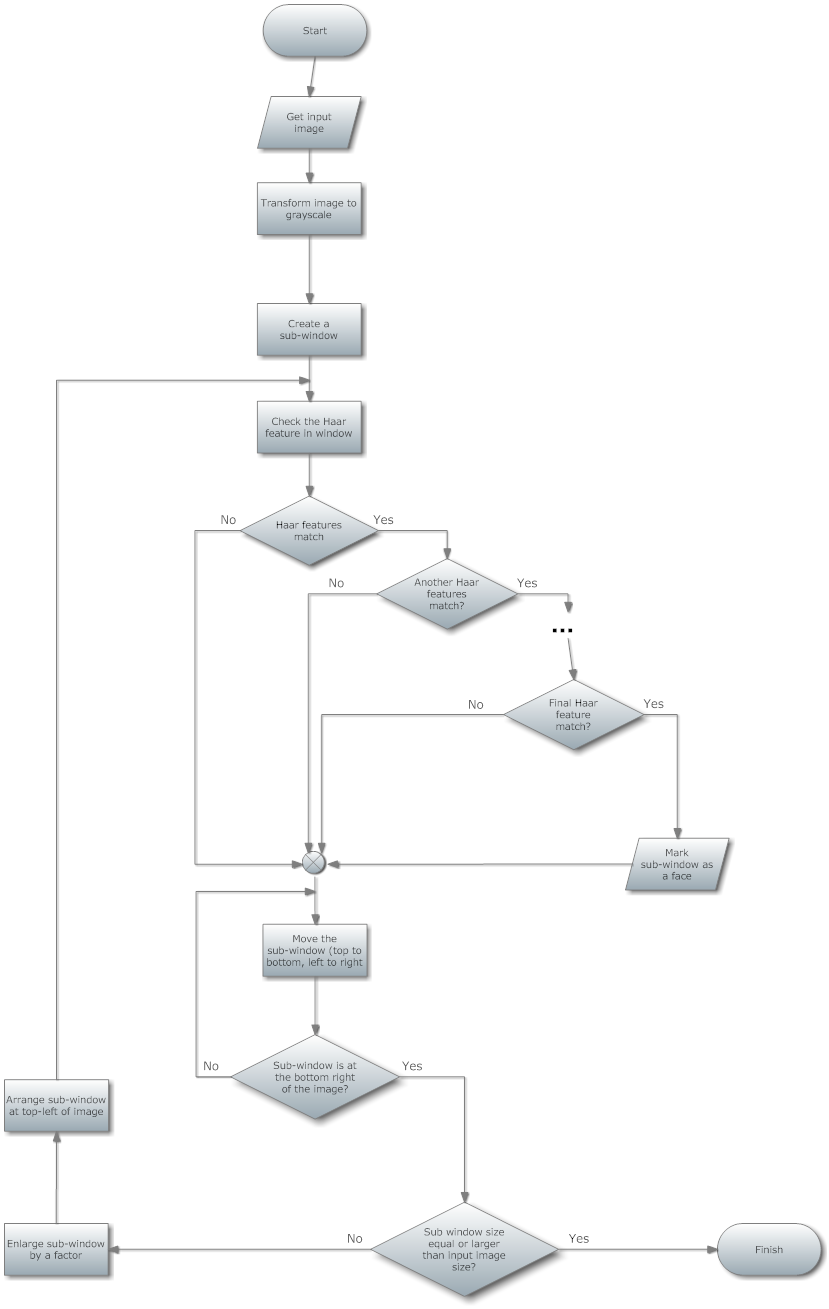


Figure 88: Viola-Jones method flowchart

### Face Recognition

#### Definition

* Face recognition is the task of identifying an already detected object as a KNOWN or UNKNOWN face, and in more advanced cases, telling EXACTLY WHO'S face it is!
* One of the ways to do this is by comparing selected facial features from the image and a facial database.

Reference: <http://en.wikipedia.org/wiki/Facial_recognition_system>

#### Difference between Face Recognition and Face Detection

* **Face Detection:** identify an object as a "face" and locate it in the input image.
* **Face Recognition**: decide if this "face" is someone KNOWN, or UNKNOWN, basing on the database of faces it uses to validate this input face.
* Face detection's output( the face) is in fact recognition's input. And face recognition's output is the final decision: face known/faceunknown!

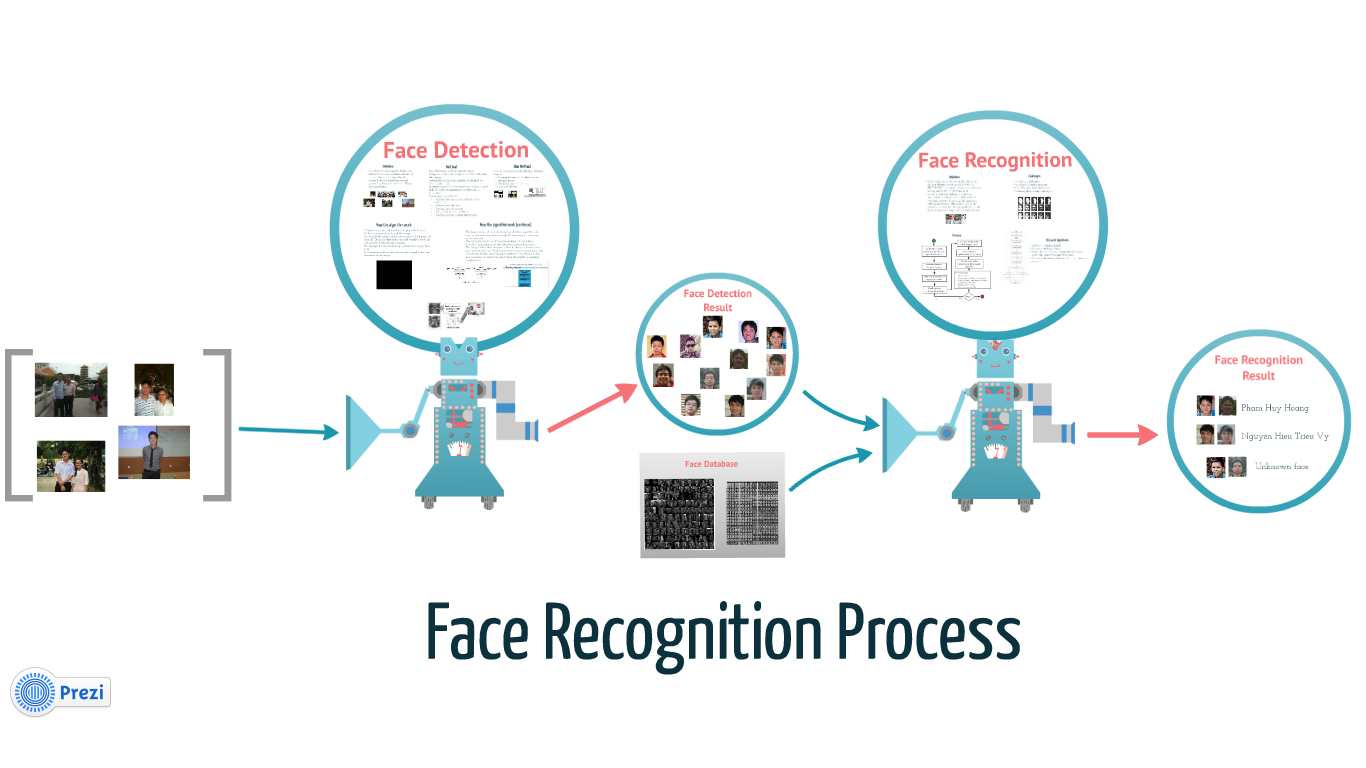


Figure 89: Face Recognition Process

#### Method for face recognition

**Recognition algorithms can be divided into two main approaches:**

* Geometric: which looks at distinguishing features.
* Photometric: which is a statistical approach that distill an image into values and comparing the values with templates to eliminate variances.

**Popular recognition algorithms** include:

1. Principal Component Analysis using Eigenfaces
2. Linear Discriminate Analysis
3. Elastic Bunch Graph Matching
4. TheHidden Markov model
5. The neuronal motivated dynamic link matching*.*

References: <http://www.biometricscatalog.org/NSTCSubcommittee>

#### The Method we choose

We chose Fisherface method, reason:

* Already implemented in openCV.
* Has the best result, according to report. (<http://asp.eurasipjournals.com/content/2012/1/92>)
* Has the best result in our automation testing, with our current face database.

**Method Introduction**

Linear Discriminant Analysis (LDA), derived from an idea suggested by R.A. Fisher in 1936. When LDA is used to find the subspace representation of a set of face images, the resulting basis vectors defining that space are known as Fisherfaces.

**Algorithm description:**

* Construct the Imagematrix X with each column representing an image. Each image is a assigned to a class in the corresponding class vector C.
* Project X into the (N-c)-dimensional subspace as P with the rotation matrix WPca identified by a Principal Component Analysis, where
* N is the number of samples in X
* c is unique number of classes (length(unique(C)))
* Calculate the between-classes scatter of the projection P as Sb = \sum\_{i=1}^{c} N\_i\*(mean\_i - mean)\*(mean\_i - mean)^T, where
* mean is the total mean of P
* mean\_i is the mean of class i in P
* N\_i is the number of samples for class i
* Calculate the within-classes scatter of P as Sw = \sum\_{i=1}^{c} \sum\_{x\_k \in X\_i} (x\_k - mean\_i) \* (x\_k - mean\_i)^T, where
* X\_i are the samples of class i
* x\_k is a sample of X\_i
* mean\_i is the mean of class i in P
* Apply a standard Linear Discriminant Analysis and maximize the ratio of the determinant of between-class scatter and within-class scatter. The solution is given by the set of generalized eigenvectors Wfld of Sb and Sw corresponding to their eigenvalue. The rank of Sb is atmost (c-1), so there are only (c-1) non-zero eigenvalues, cut off the rest.
* Finally obtain the Fisherfaces by W = WPca \* Wfld.

Reference: <http://www.scholarpedia.org/article/Fisherfaces>, <http://www.bytefish.de/blog/fisherfaces/>

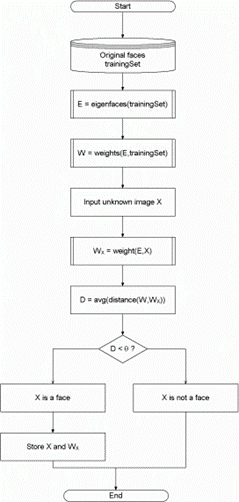


Figure 90: Face Recognition Algorithm Flowchart