** MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Capstone Project Document

The Traffic Sign Recognition and Training

|  |  |
| --- | --- |
| **Group 2** | |
| **Group member** | Mai Văn Tân – Team Leader – SE90061  Bùi Việt Phong – Team Member - SE60747  Hồ Đắc Nghĩa – Team Member - SE60628  Trần Lê Tuấn – Team Member – 60350 |
| **Supervisor** | Mr. Kiều Trọng Khánh |
| **Ext. Supervisor** | N/A |
| **Capstone Project code** | TSRT |

-Ho Chi Minh City, 01/2014-

*This page is intentionally left blank*

***ACKNOWLEDGEMENTS***

# Table of Contents

[**Table of Contents 4**](#_Toc369941326)

[**List of Tables 5**](#_Toc369941327)

[**List of Figures 7**](#_Toc369941328)

[Definitions, Acronyms, and Abbreviations 9](#_Toc369941329)

[Report No.4 Software Desgin Description 10](#_Toc369941330)

[1. Design Overview 10](#_Toc369941331)

[2. System Architectural Design 11](#_Toc369941332)

[3. Component Diagram 12](#_Toc369941333)

[4. Detailed Description of Components 13](#_Toc369941334)

[4.1 Class Diagram 13](#_Toc369941335)

[4.2 Class Diagram Explanation 13](#_Toc369941336)

[4.3 Sequence Diagram 21](#_Toc369941337)

[5. Database Design 33](#_Toc369941338)

[5.1 Logical Diagram 33](#_Toc369941339)

[5.2 Data Dictionary 34](#_Toc369941340)

[6. Algorithms 40](#_Toc369941341)

[6.1 Face Detection 40](#_Toc369941342)

[6.2 Face Recognition 44](#_Toc369941343)

# List of Tables

[Table 1: Entity Data Dictionary 32](#_Toc369941379)

[Table 2: Attribute Data Dictionary 37](#_Toc369941380)

# List of Figures

[Figure 1: MVC Architecture 9](#_Toc369941449)

[Figure 2: Component Diagram 10](#_Toc369941450)

[Figure 3: Class Diagram 11](#_Toc369941451)

[Figure 4: Add Image for Singe Student Sequence Diagram 19](#_Toc369941452)

[Figure 5: Add Image for Many Students Sequence Diagram 19](#_Toc369941453)

[Figure 6: Delete Student Image Sequence Diagram 20](#_Toc369941454)

[Figure 7: Create Account Sequence Diagram 20](#_Toc369941455)

[Figure 8: Configure System Sequence Diagram 21](#_Toc369941456)

[Figure 9: Face Detection Sequence Diagram 21](#_Toc369941457)

[Figure 10: Face Recognition Sequence Diagram 22](#_Toc369941458)

[Figure 11: Auto Free Storage Space Sequence Diagram 22](#_Toc369941459)

[Figure 12: Auto Active Roll Call Sequence Diagram 23](#_Toc369941460)

[Figure 13: Add Roll Call Sequence Diagram 23](#_Toc369941461)

[Figure 14: Edit Roll Call Sequence Diagram 24](#_Toc369941462)

[Figure 15: Import Student List Sequence Diagram 24](#_Toc369941463)

[Figure 16: Export Report Sequence Diagram 25](#_Toc369941464)

[Figure 17: Add Student Sequence Diagram 25](#_Toc369941465)

[Figure 18: Add Class Sequence Diagram 26](#_Toc369941466)

[Figure 19:Edit Class Sequence Diagram 26](#_Toc369941467)

[Figure 20: Add Subject Sequence Diagram 27](#_Toc369941468)

[Figure 21: Edit Subject Sequence Diagram 27](#_Toc369941469)

[Figure 22: Take Attendance Auto Sequence Diagram 28](#_Toc369941470)

[Figure 23: Take Attendance Manual Sequence Diagram 28](#_Toc369941471)

[Figure 24: View Roll Call Detail Sequence Diagram 29](#_Toc369941472)

[Figure 25: Report Attendance by Class Sequence Diagram 29](#_Toc369941473)

[Figure 26: Check Present Rate Sequence Diagram 30](#_Toc369941474)

[Figure 27: Logical Diagram 31](#_Toc369941475)

[Figure 28: Example of Haar Future 38](#_Toc369941476)

[Figure 29: Apply Haar future to sub-window 39](#_Toc369941477)

[Figure 30: Classifier Cascade 39](#_Toc369941478)

[Figure 31: Viola-Jones method flowchart 41](#_Toc369941479)

[Figure 32: Face Recognition Process 42](#_Toc369941480)

[Figure 33: Face Recognition Algorithm Flowchart 44](#_Toc369941481)

# Definitions, Acronyms, and Abbreviations

# Report No.4 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

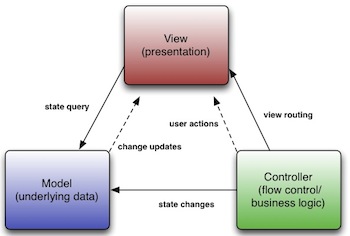


Figure 1: MVC Architecture

<http://www.java-forums.org/ocmjea/57933-tutorial-review-web-tier-application-architecture-java-architect-exam.html>

* **Model** - The model represents data and the rules that govern access to and updates of this data. In enterprise software, a model often serves as a software approximation of a real-world process.
* **View** - The view renders the contents of a model. It specifies exactly how the model data should be presented. If the model data changes, the view must update its presentation as needed. This can be achieved by using a push model, in which the view registers itself with the model for change notifications, or a pull model, in which the view is responsible for calling the model when it needs to retrieve the most current data.
* **Controller** - The controller translates the user's interactions with the view into actions that the model will perform. In a stand-alone GUI client, user interactions could be button clicks or menu selections, whereas in an enterprise web application, they appear as GET and POST HTTP requests. Depending on the context, a controller may also select a new view -- for example, a web page of results -- to present back to the user.

<http://www.oracle.com/technetwork/articles/javase/index-142890.html>

## Component Diagram

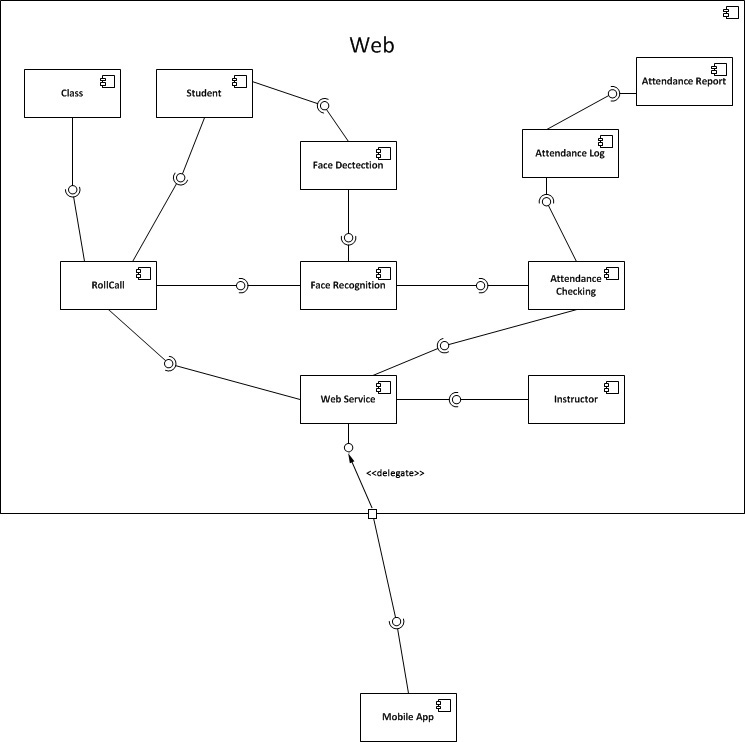


Figure 2: Component Diagram

## Detailed Description of Components

### Class Diagram

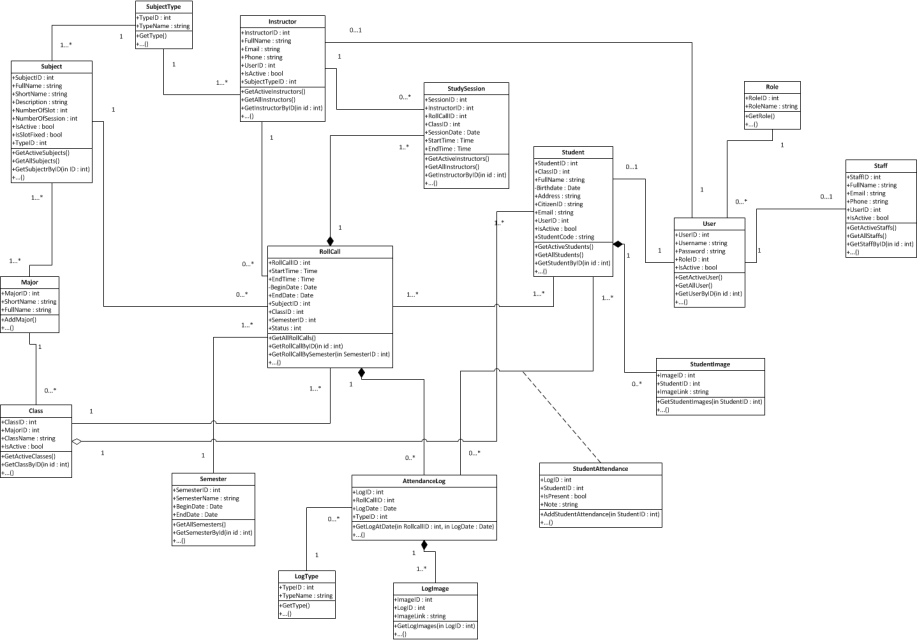


Figure 3: Class Diagram

### 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

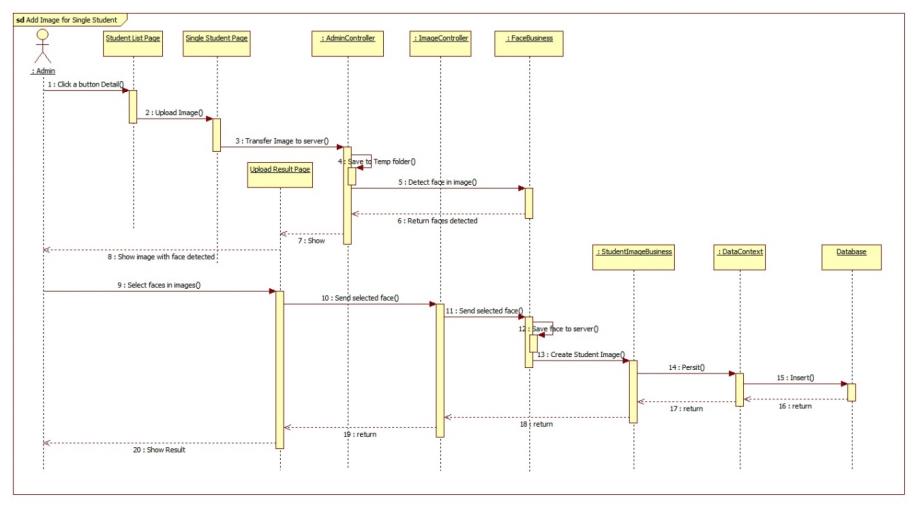


Figure 4: Add Image for Singe Student Sequence Diagram

#### Add Image for Many Students

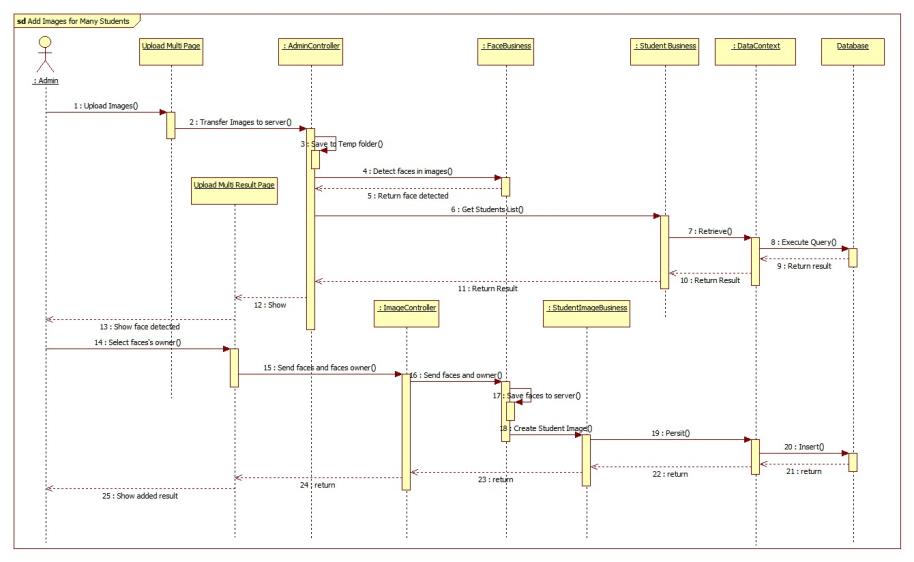


Figure 5: Add Image for Many Students Sequence Diagram

#### Delete Student Image

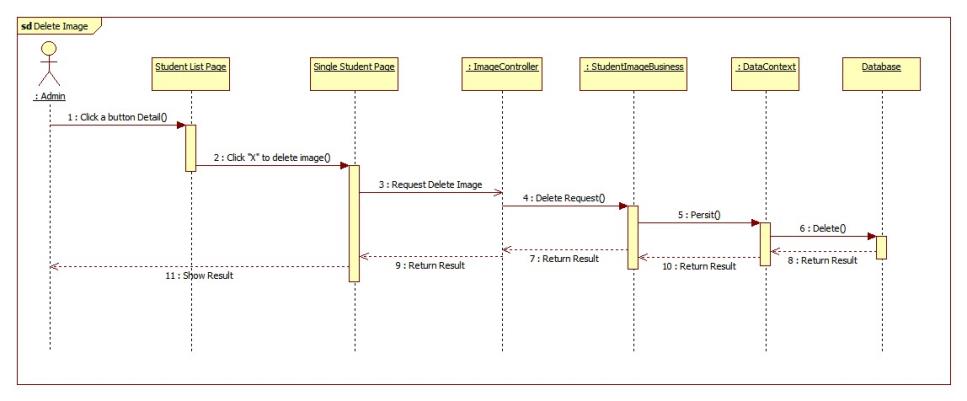


Figure 6: Delete Student Image Sequence Diagram

#### Create Account

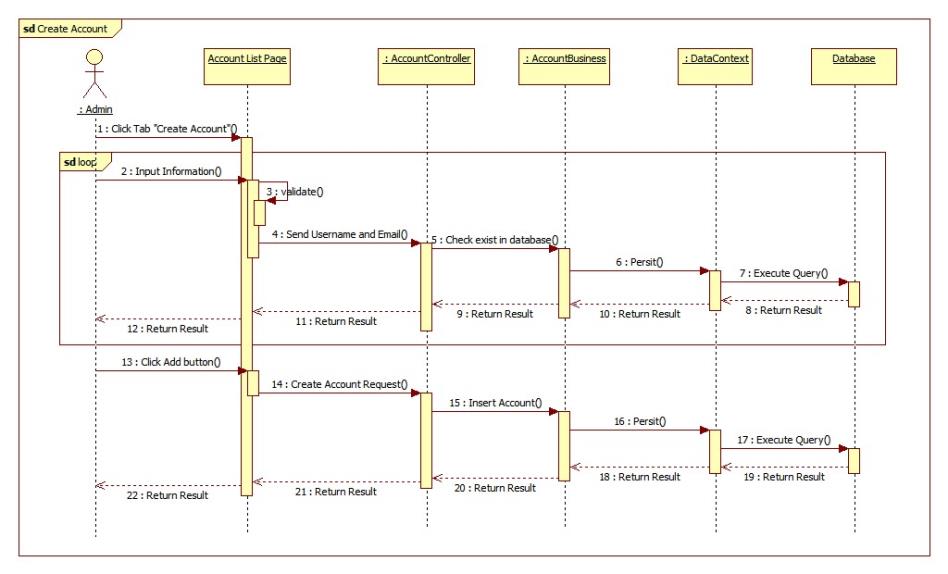


Figure 7: Create Account Sequence Diagram

#### Configure System

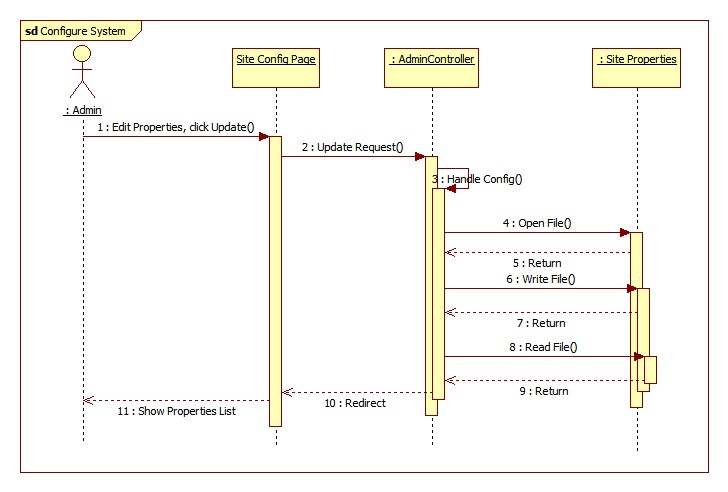


Figure 8: Configure System Sequence Diagram

#### Face Detection

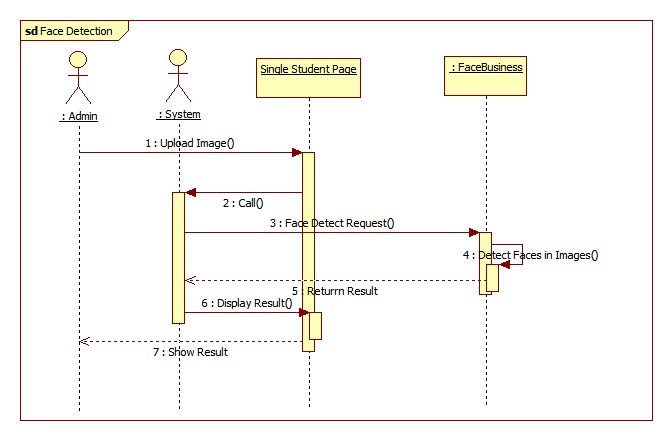


Figure 9: Face Detection Sequence Diagram

#### Face Recognition

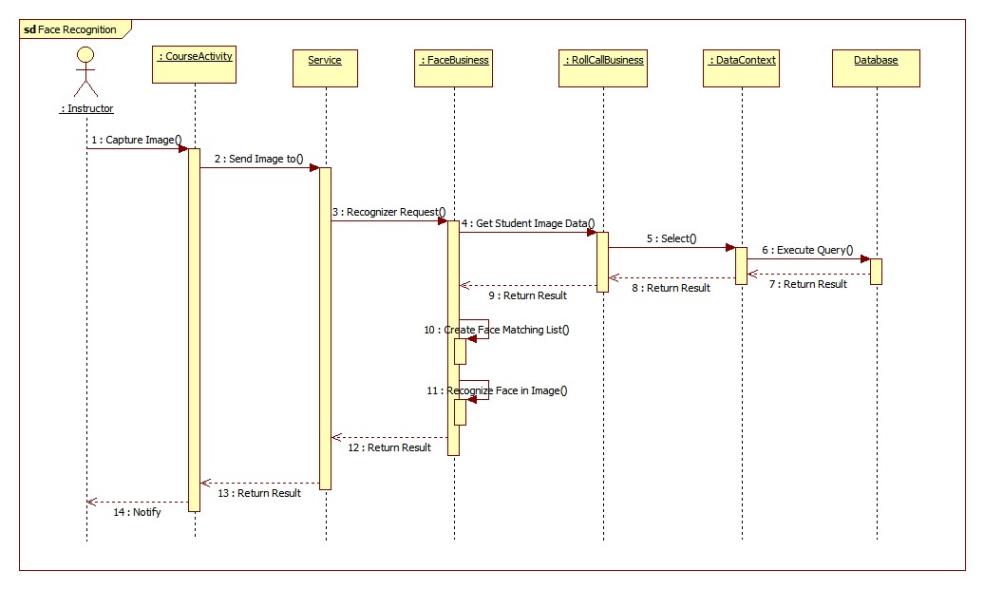


Figure 10: Face Recognition Sequence Diagram

#### Auto Free Storage Space

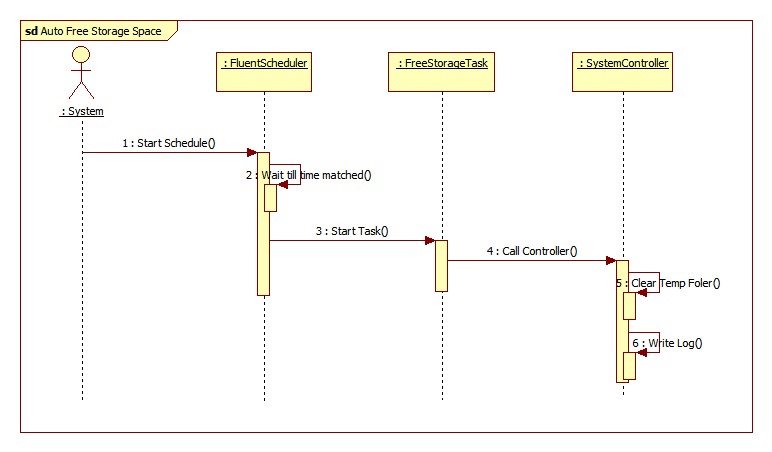


Figure 11: Auto Free Storage Space Sequence Diagram

#### Auto Active Roll Call

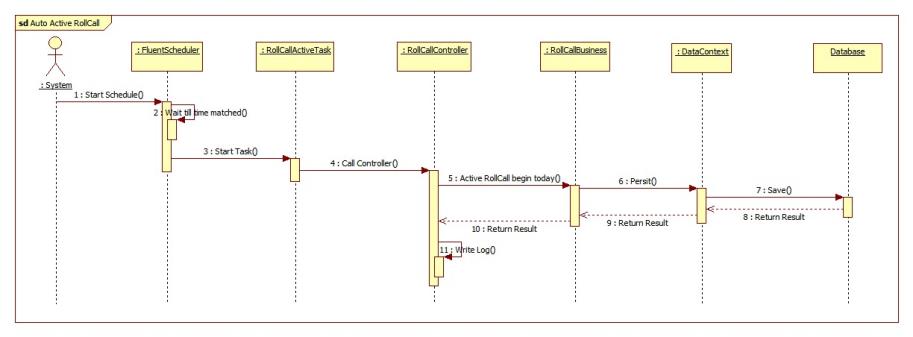


Figure 12: Auto Active Roll Call Sequence Diagram

#### Add Roll Call

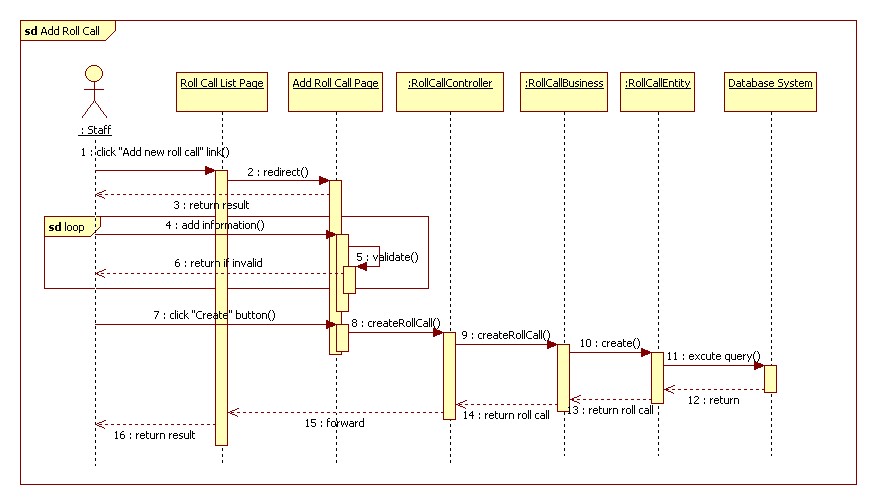
****

Figure 13: Add Roll Call Sequence Diagram

#### Edit Roll Call

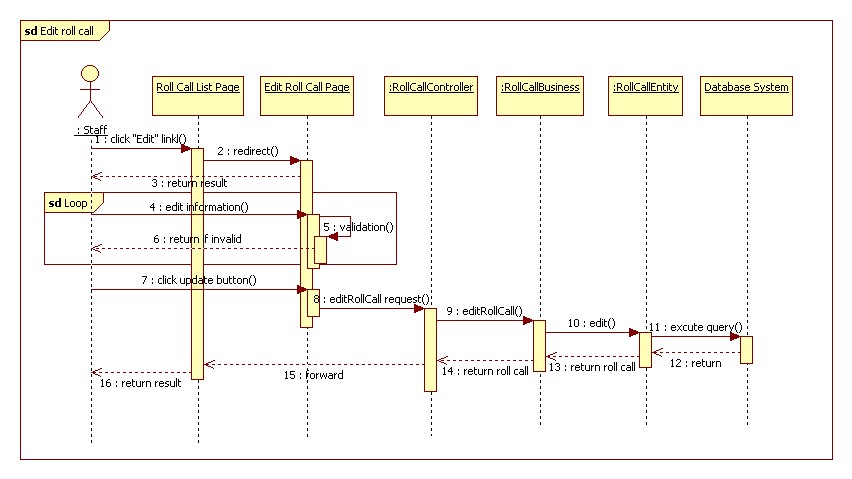
****

Figure 14: Edit Roll Call Sequence Diagram

#### Import Student List

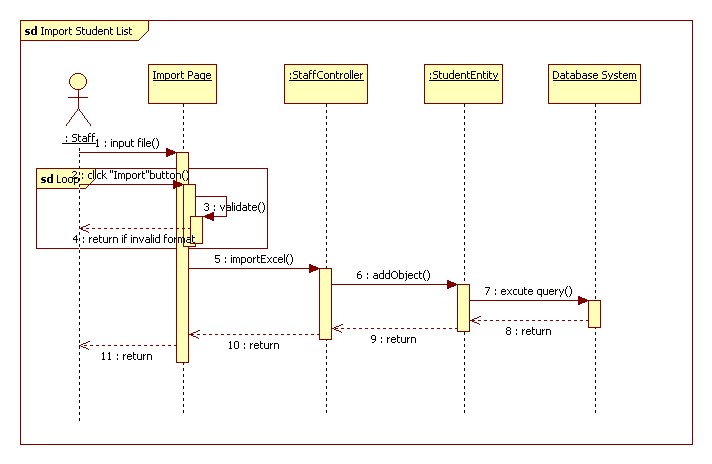
****

Figure 15: Import Student List Sequence Diagram

#### Export Report

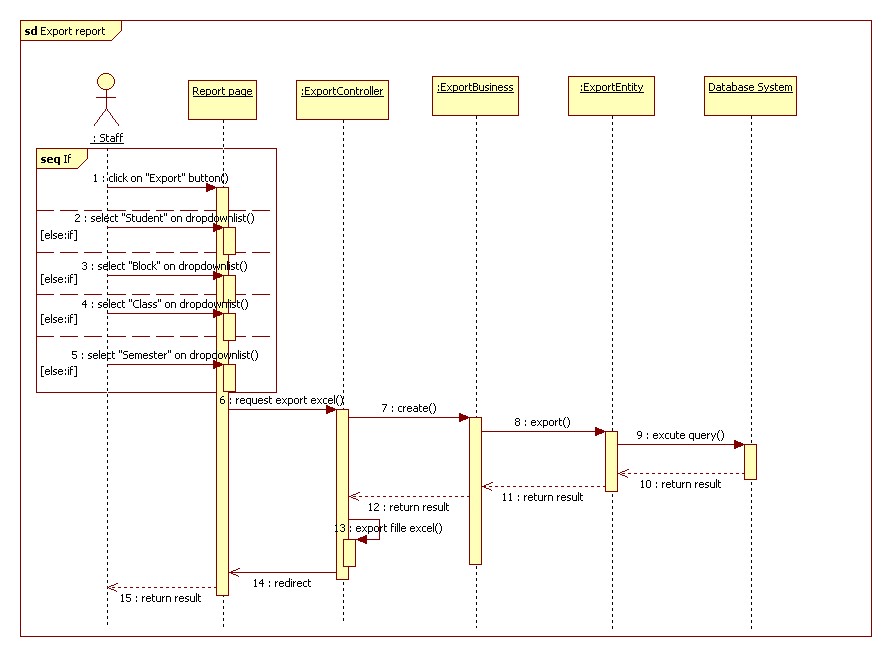
****

Figure 16: Export Report Sequence Diagram

#### Add Student

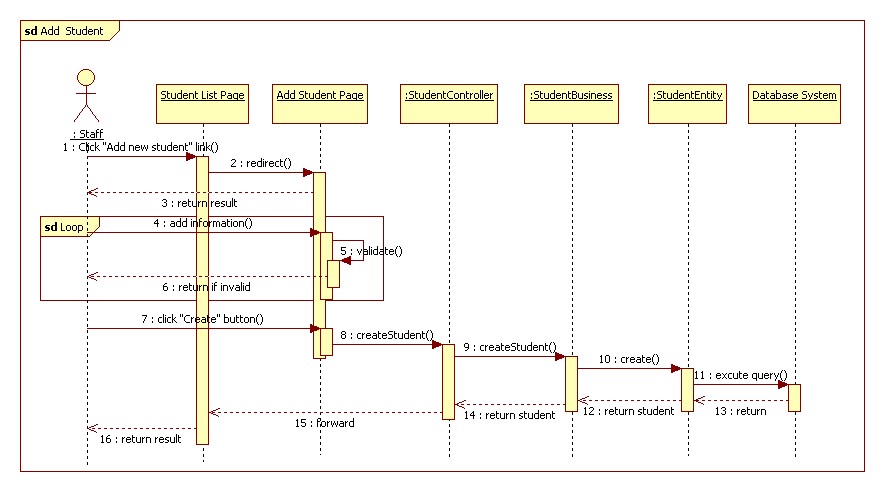
****

Figure 17: Add Student Sequence Diagram

#### Add Class

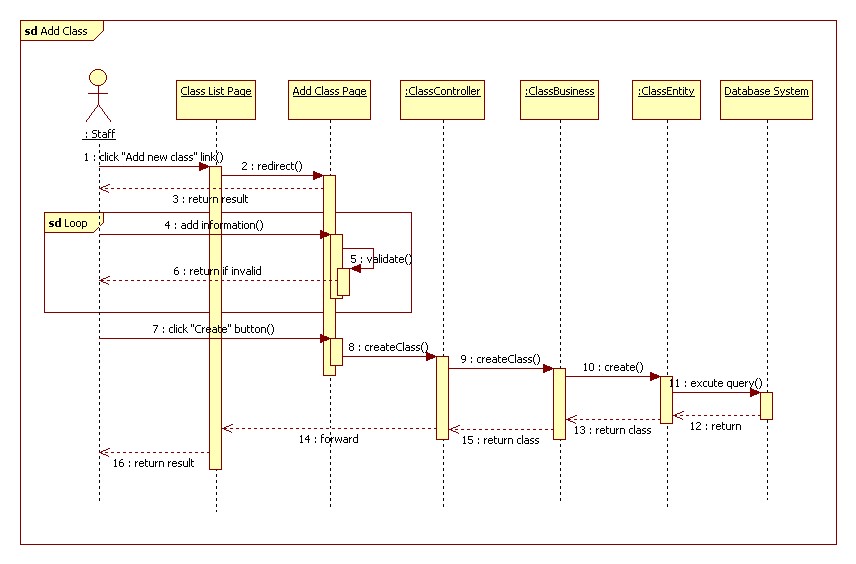
****

Figure 18: Add Class Sequence Diagram

#### Edit Class

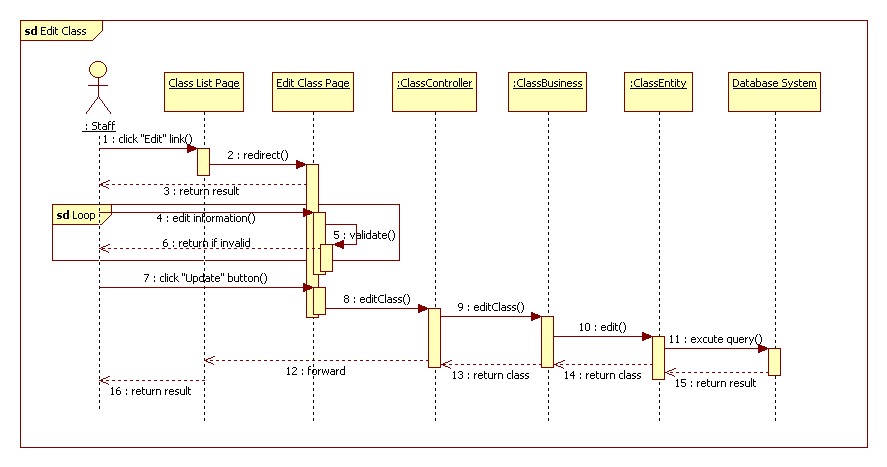
****

Figure 19:Edit Class Sequence Diagram

#### Add Subject

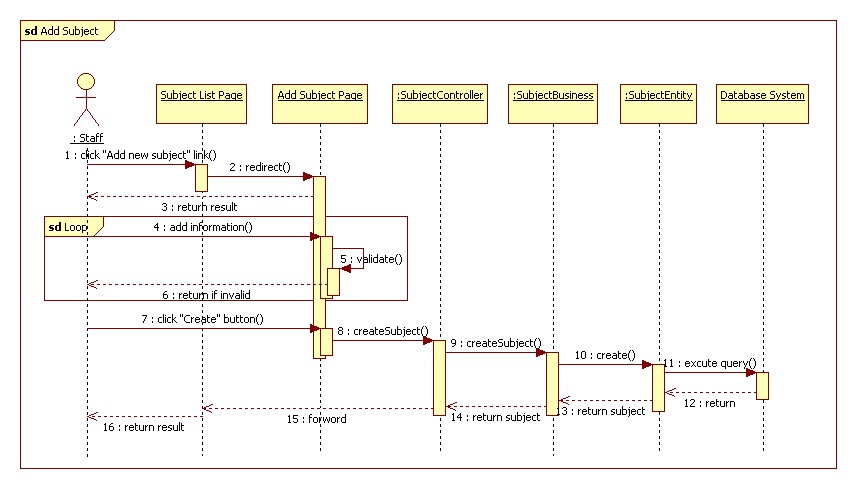
****

Figure 20: Add Subject Sequence Diagram

#### Edit Subject

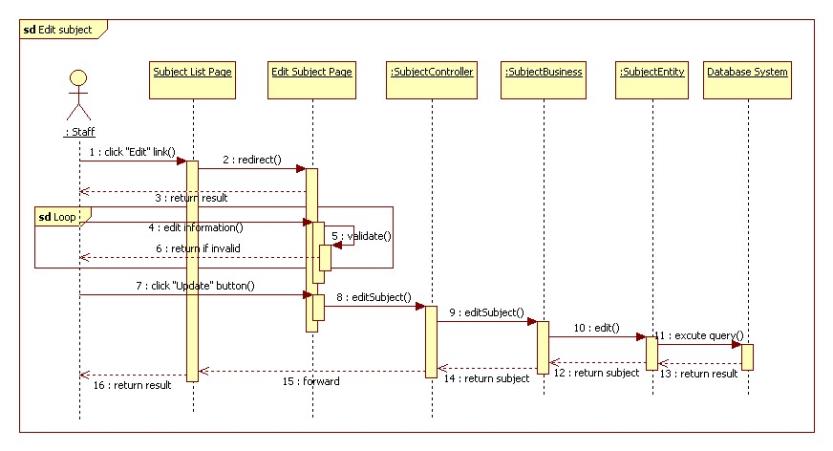
****

Figure 21: Edit Subject Sequence Diagram

#### Take Attendance Auto

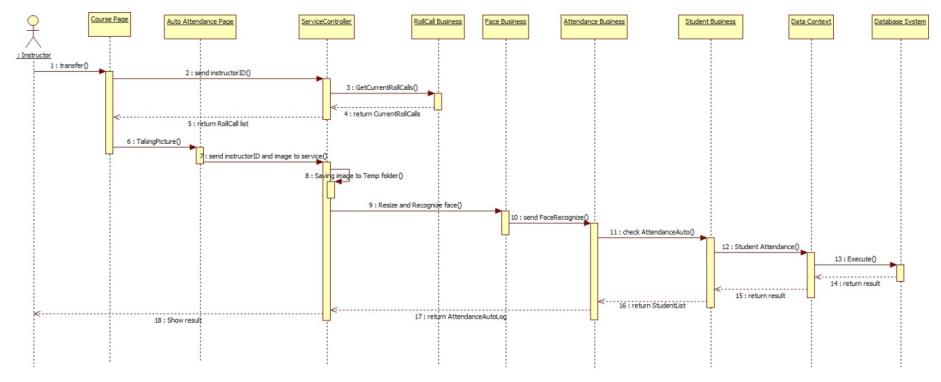
****

Figure 22: Take Attendance Auto Sequence Diagram

#### Take Attendance Manual

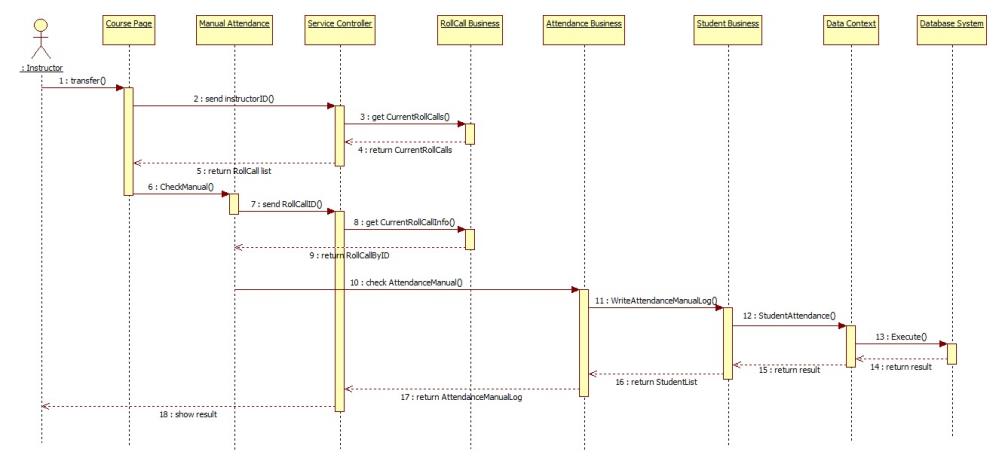
****

Figure 23: Take Attendance Manual Sequence Diagram

#### View Roll Call Detail

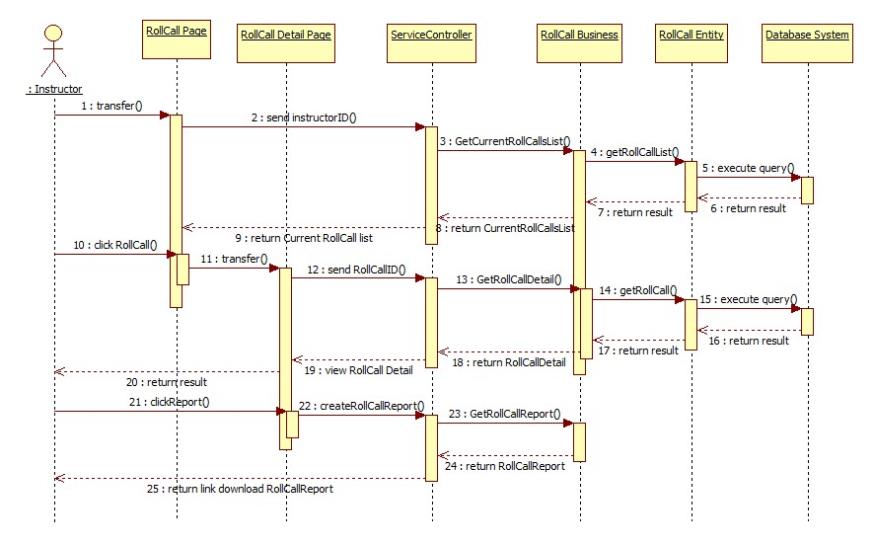
****

Figure 24: View Roll Call Detail Sequence Diagram

#### Report Rollcall by Class

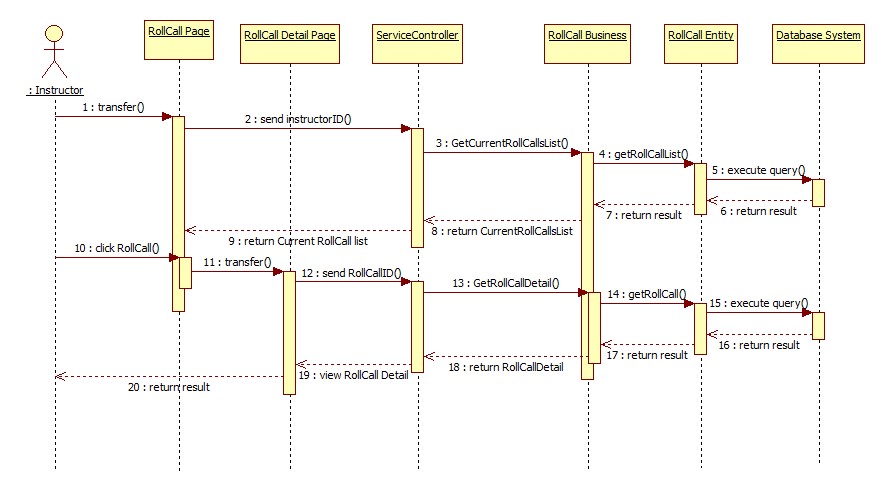
****

Figure 25: Report Attendance by Class Sequence Diagram

#### Check Present Rate

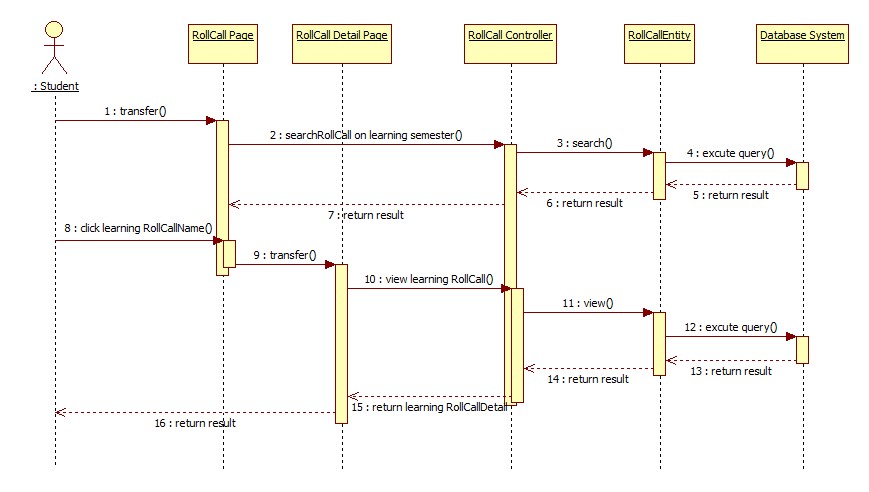
****

Figure 26: Check Present Rate Sequence Diagram

## Database Design

### Logical Diagram

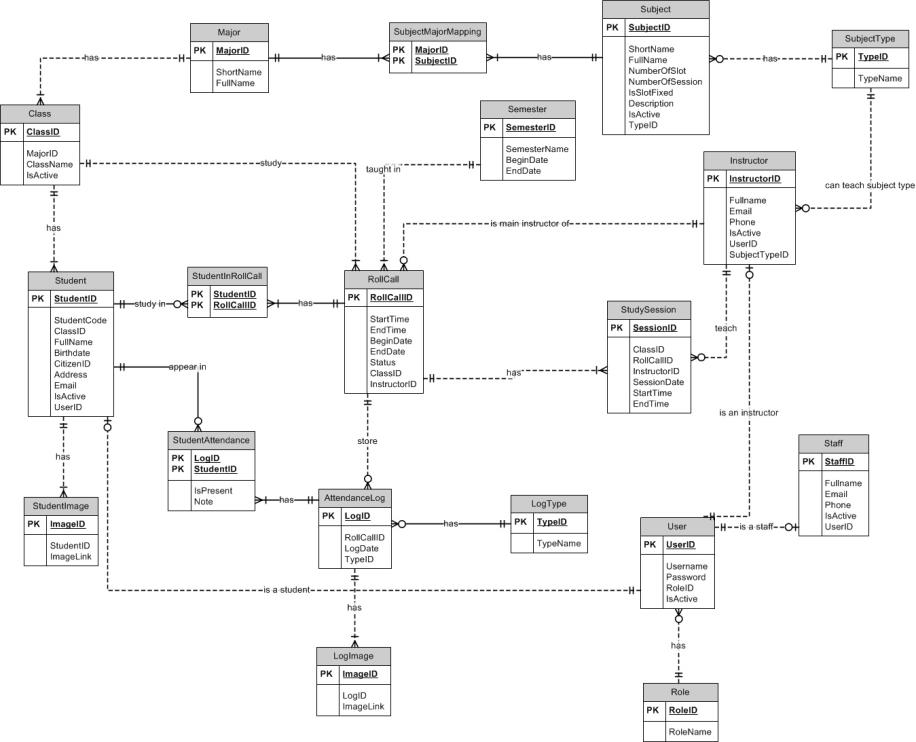


Figure 27: Logical Diagram

### 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 1: 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 2: 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

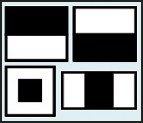


Figure 28: Example of Haar Future

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 29: 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.

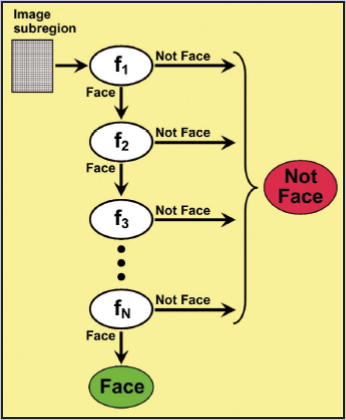


Figure 30: Classifier Cascade

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.

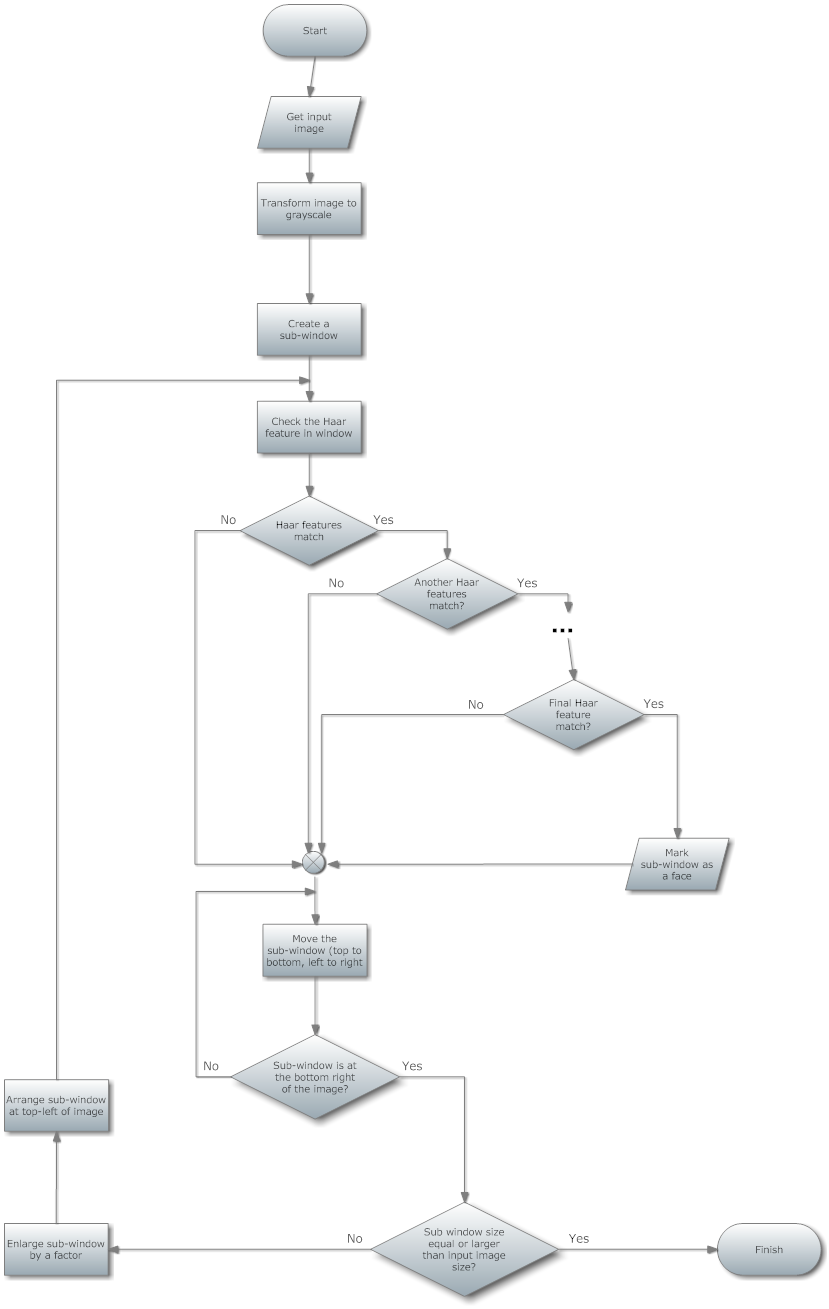


Figure 31: Viola-Jones method flowchart

### 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!

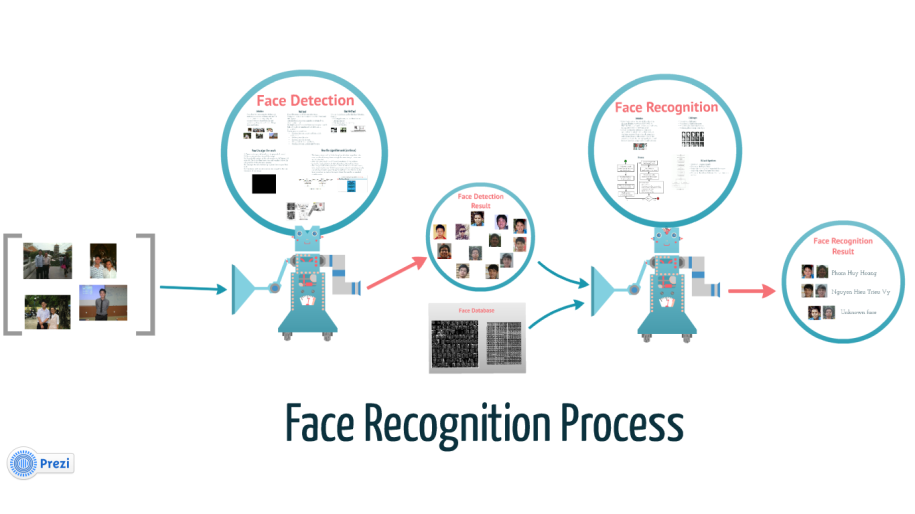


Figure 32: Face Recognition Process

#### 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/>

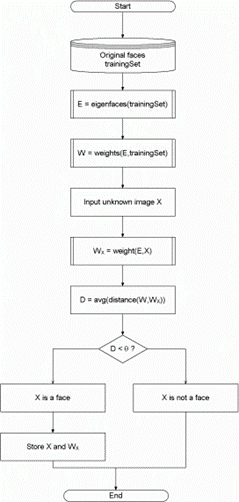


Figure 33: Face Recognition Algorithm Flowchart