Design Document for Attendance by Face Recognition System

Group #7

Shakhan Mukhammedali(MS): %marks

Muratova Inabat(IM): %marks

Aitenova Meiirzhan(MA): %marks

Myrzabekova Khanzada(KM): % marks

Kereybaev Ersayin(EK): % marks

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Change** |
| 0.1 | 10/08/04 | SM | Initial Document |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Introduction 3](#_Toc84977455)

[1.1 Purpose 3](#_Toc84977456)

[1.2 Scope 3](#_Toc84977457)

[1.3 Definitions, Acronyms, Abbreviations 3](#_Toc84977458)

[1.4 Design Goals 3](#_Toc84977459)

[2 References 4](#_Toc84977460)

[3 Decomposition Description 5](#_Toc84977461)

[3.1 Module Decomposition 5](#_Toc84977462)

[3.2 Concurrent Process 6](#_Toc84977463)

[3.3 Data Decomposition 7](#_Toc84977464)

[3.4 STATES 7](#_Toc84977465)

[4 Dependency Description 0](#_Toc84977466)

[4.1 Intermodule Dependencies 0](#_Toc84977467)

[4.2 InterProcess Dependencies 0](#_Toc84977468)

[4.3 Data Dependencies 0](#_Toc84977469)

[5 Interface Description 0](#_Toc84977470)

[5.1 Module Interface 0](#_Toc84977471)

[5.2 Process Interface 0](#_Toc84977472)

[6 Detailed Design 0](#_Toc84977473)

[7 Design Rationale 0](#_Toc84977474)

[7.1 Design Issues 0](#_Toc84977475)

[7.2 <Issue 1> 0](#_Toc84977476)

[7.3 <Issue 1> 0](#_Toc84977477)

[8 Traceability 0](#_Toc84977478)

# Introduction

## Purpose

## The purpose of this document is to explain the design and architecture of the Attendance by Face Recognition System.

## Scope

This document has descriptions of system decomposition, interfaces, and dependencies, design rationale.

## Definitions, Acronyms, Abbreviations

## None

|  |  |
| --- | --- |
| Term | Description |
|  |  |
|  |  |

## Design Goals (MS)

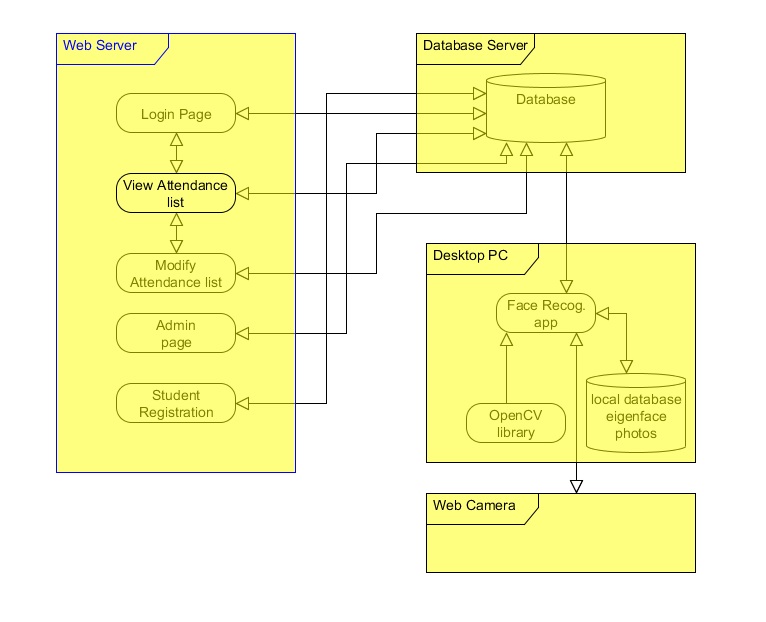
1. Reliability – The process must continually function consistently, and loss data (when sending the photo to the desktop app and the result of the face recognition to the web server) should be prevented.
2. Maintainability - Our code must be well organized with consistent syntax and relevant naming techniques.
3. Response Time – recognition, comparing face and sending the result to the database should take a max 1 minute.
4. User privacy – Instructor’s must be allowed to view and modify attendance list only for the groups that he teaches.

# References

<http://docs.opencv.org/>

# Decomposition Description

## Module Decomposition



**Arch Diagram for Attendance by Face Recognition System.**

The project includes Desktop application and server architectures, it consists of five subsystems. Server architecture consists of three subsystems. First, the subsystem is intended for instructor’s, which contains Login Page, View Attendance, and Modify Page. The second subsystem consists of the Student Registration Page. The third subsystem is Admin Page. Fourth is a Database. But the more important aspect of the architecture is the fifth desktop application subsystem. (MA)

### Desktop Page Description(IM)

Desktop will consist of Face Recognition App and Local Eigenface Photos Database. Face Recognition App will take photos of student's for checking attendance, it will be provided by Eigenface Algorithm which use Open CV library. Local Eigenface Photos Database will store list of student's photo. Photo which made by Face Recognition App will compared with photo from Local Eigenface Photos Database, if the photos matched result will be sent to Web Server.

### Instructor's Page Description(KM)

Instructor's Page Description will consist of LoginPage, View Attendance, and Modify Attendance list pages. Login page will provide entrance into instructor's page. View Attendance list which visually displays list of student's attendance. Modify Attendance List will give permit to instructor to change or modify attendance.

### Admin Page Description(EK)

Admin Page allows to Admin modifying (removing or adding new instructor, subject, can modify timetable), managing over database.

### Student Page Description(EK)

Students only allowed register to system.

### Database Description(EK)

Database (MySQL) is used to store all data. Specifically, all attendance list, information about students and instructors, list of students and instructors, timetable will be stored on the Server side. On request it will return attendance list of corresponding student or group. It will allow to instructor to modify attendance. It will interact with Desktop Page, Modify Attendance Page, Admin Page (Add/Update attendance, instructor, timetable) directly.

## Concurrent Process

### Login Page Description(MS)

### The index page of our site is Login Page. Only Instructors allowed to enter to system by typing user name and password. If user with that user name does not exist or password is incorrect corresponding message will appear. On the top right side of the Login Page we will have link to the Student Registration Page

### View Attendance Page Description(KM)

### In this page Instructor can view the attendance table of the students by selecting course and group. On the bottom right side of the attendance table there will be “Modify” button that link us to Modify Attendance Page. On the top right side of the View Attendance Page we will have link “Logout”.

### Modify Attendance Page Description(IM)

### In this page Instructor can modify the attendance table if student have some reasons to not go to this lesson at some day (if student was ill or etc.). On the bottom right side of the attendance table of this page there will be “Back” button that link us to View Attendance Page. On the top right side of the Modify Attendance Page we will have link “Logout”.

### Admin Page Description(EK)

### In this page admin can add, remove, and modify profile of the instructor. Admin also can modify schedule and courses. By clicking “Add Instructor” link admin can add instructor. Similarly by clicking “Remove Instructor” or “Modify Instructor Profile” link admin can remove selected instructors or modify the instructor’s profile.

### Student Registration Page Description(MA)

### In this page students have to register to system. Because if a student didn‘t register to system, our application can’t take attendance of that student. In registration page student have to fill all fields. Students also have to browse their photo to system and take courses to learn.

### Desktop app Page Description(EK)

### In this page the desktop app receive the photos and compares this photo with other photos in Local Eigenface Photos Database. If received photo matches with the photo in Local Eigenface Photos Database the some info will be sent to Web Server (this student is attended to this class).

## Data Decomposition

### Student table

### student\_id INTEGER(students write their id’s from portal.sdu)

### first\_name VARCHAR(30)

### last\_name VARCHAR(30)

### photo LONGBLOB NOT NULL

### faculty\_id INTEGER NOT NULL REFERENCES faculty.faculty\_id

### group\_id INTEGER NOT NULL REFERENCES groups.group\_id

### student\_course\_id INTEGER NOT NULL

### Faculty table

### faculty\_id INTEGER AUTO\_INCREMENT NOT NULL

### faculty\_name VARCHAR(30)

### Groups table

### group\_id INTEGER AUTO\_INCREMENT NOT NULL

### group\_faculty\_id INTEGER NOT NULL REFERENCES faculty.faculty\_id

### group\_name VARCHAR(30)

### Course table

### course\_id INTEGER AUTO\_INCREMENT NOT NULL

### course\_faculty\_id INTEGER NOT NULL REFERENCES faculty.faculty\_id

### course\_name VARCHAR(30)

### Student\_Course table

### student\_course\_id INTEGER AUTO\_INCREMENT NOT NULL

### student\_id INTEGER NOT NULL REFERENCES student.student\_id

### course\_id INTEGER NOT NULL REFERENCES course.course\_id

### Instructor table

### instructor\_id INTEGER AUTO\_INCREMENT NOT NULL

### first\_name VARCHAR(30)

### last\_name VARCHAR(30)

### faculty\_id INTEGER NOT NULL REFERENCES faculty.faculty\_id

### instructor\_group\_id INTEGER NOT NULL

### instructor\_course\_id INTEGER NOT NULL

### Instructor\_Group table

### instructor\_group\_id INTEGER AUTO\_INCREMENT NOT NULL

### instructor\_id INTEGER NOT NULL REFERENCES instructor.instructor\_id

### group\_id INTEGER NOT NULL REFERENCES groups.group\_id

### Instructor\_Course table

### instructor\_course\_id INTEGER AUTO\_INCREMENT NOT NULL

### instructor\_id INTEGER NOT NULL REFERENCES instructor.instructor\_id

### course\_id INTEGER NOT NULL REFERENCES course.group\_id

## STATE

# Dependency Description

## Intermodule Dependencies

Java code module is dependent on OpenCV library of C++, because main operations made in java code is taken from OpenCV library. It is also dependent on internet connection to send a result of the checks made in desktop app to Server.

Database module is dependent on Libraries module to be able to connect to desktop app.

## InterProcess Dependencies

See sequence diagrams in SRS document

## Data Dependencies

## [NONE]

# Interface Description

## Module Interface

### Administration page Interface

#### public function addInstructor(name, surname, faculty, groups, subjects);

This method crates new Instructor and saves it with information faculty, group and his subjects in database.

#### public function updateInstructor(name, surname, faculty, groups, subjects);

This method updates information about instructor.

#### public function removeInstructor(id);

This method removes instructor existing by this id;

### Login page Interface

#### public function check(login, password);

This function will check is in database exists instructor with this login and is the password correct;

### Instructor page Interface

#### public function showTable();

This function will be executed when instructor selects subject (or course) and group. And shows to user attendance table of selected group.

### Student registration page Interface

#### addStudent(name, surname, faculty, group);

This method adds new student to database with his name, surname, faculty, group

### Desktop application

#### public function saveStudentPresent(sid, currentTime);

This function will save student with this id in database as present in the lecture in current time.

## PROCESS INTERFACE

#### Main process

### Description: This process shows all graphical interface of the system

### 5.2.1.1 Process is created when the application started

### 5.2.1.2 Terminated when applications close button is pressed

### 5.2.1.3 All other threads will be killed if this main thread stops

#### Face detection process

Description: Tries to detect face. When face will detected, sends photo to Main process to find match.

### Process is created when the application started

### Terminated when applications close button is pressed

# Detailed Design

NOT REQUIRED <Java Docs to be used instead>

# Design Rationale

## ATTENDANCE BY FACE RECOGNITION

### **Description**

### We will use MySQL database in our project

### **Factors affecting issue**

#### The database type can’t be changed

#### We have to have additional libraries to use C++ OpenCV library in our project

### **Alternatives and their pros and cons**

#### We must always have the internet connection

* + - * 1. If we don’t have the internet connection than we can’t send the results of the checking to the Server
        2. In order to resolve this problem we will save the results of checking’s in database

#### Write separate Database connection class

### **Resolution of Issue**

### We decided to save results of checking’s to the database if we don’t have internet connection.

### 

## Database connection and connection with camera

### Description

Our app should provide one stable connection to the database and to Web Server.

### Factors affecting Issue

### 7.2.2.1 Too many opened connections may get server go down

### Alternatives and their pros and cons

#### There have to be one single connection for one user used by any user thread

7.2.3.2 For each client create separate connection to database and Web Server

7.2.3.3 Desktop app have to receive the photo from camera every specified time

7.2.3.3.1 Application will not receive the photo if check progress is not completed yet.

### Resolution of Issue

We have to create queue in order to receive the photos from camera and save it in queue. And after app completes checking’s it will take the photo from queue.

# Traceability

|  |  |  |
| --- | --- | --- |
| **No** | **Use Case/ Non-functional Description** | **Subsystem/Module/classes that handles it** |
| 1 |  |  |
| 2 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

FEEL FREE TO ADD APPENDICES AS NEEDED. UPDATE TOC BEFORE SUBMITTING