Software Systems Lab

CSP203

Design Document

Smart Attendance using face recognition

Aditya Tiwari - 2016csb1029
Himanshu Parihar - 2014csb1014
Prerna Garg - 2016csb1050
Vineet Mehta - 2016csb1063

March 8, 2018

Contents

1 Introduction 2
1.1 Objective 2
1.2 Features 2
1.2.1 Web Application 2
1.2.2 Android Application 2
1.2.3 Server 2
1.3 Limitations 2

2 Technology Description 2
2.1 Languages 2
2.2 Tools 2

3 Functional Description 3
3.1 Algorithm 3
3.1.1 Face Detection 3
3.1.2 Face Recognition 5

4 Application Interface 5
4.1 Web Interface 5
4.2 Android Interface 5
4.2.1 Code Snippets 5
4.2.2 Android Windows 7
4.3 Database Model 13
4.4 Server Interface 13

5 Learning Sources 14

6 Memberwise Role 14
6.1 Aditya Tiwari: 14
6.2 Prerna Garg: 14
6.3 Vineet Mehta: 14
6.4 Himanshu Parihar: 14

1 Introduction

1.1 Objective

The aim of this application is to improvise the attendance system used in college using one camera click of the entire class and detecting students through face detection and recognition algorithms.

1.2 Features

1.2.1 Web Application

The web application allows any user(professor/student) to view the attendance records of any student enrolled in any course. The application is a view-only interface. The only editable option is given to a professor who can add a student's attendance if it has not been recongnised by the software.

1.2.2 Android Application

The android application allows a user to take a picture and upload on the server to detect people using python code pre-hosted on the server. It also displays the results obtained after processing in the form of names of students detected.

1.2.3 Server

The basic of function of server is to host and execute python code, allow the admin to update the database i.e add/delete/update courses/students/instructors and create instructor ID's and password for login purpose. It manages the input/output from both web and android interface. It also stores the database which is used by the python code and web application.

1.3 Limitations

- The image quality has to be fairly good enough so that every face is clearly visible.
- The image should not be overly crowded.
- The image format should strictly be .jpeg or .jpg
- The accuracy of face detection is **99.99%**
- The accuracy of face recognition is **99.38%**

2 Technology Description

2.1 Languages

- Python
- Java, XML, Kotlin
- MySQL, PHP, HTML , CSS
- JavaScript
- DJango/Flask

2.2 Tools

- dlib library and its documentation
- numpy library and its documentation
- scikit-image
- Android Studio

3 Functional Description

3.1 Algorithm

3.1.1 Face Detection

Face detection is basically a program that decids whether an image is a face or not. The output is a simple 0 for NO FACE and 1 for FACE. In order to detect faces in a huge group of students we have used a function from python library dlib that detects the landmarks in the image and returns bounding squares of all possible human faces. It creates segmentations for the image and searches for faces in all segments. It uses following facial landmarks to localize and represent salient features of the face.

- Eyes
- Eyebrows
- Nose
- Mouth
- Jawline

The alorithm is an essemble of regression trees trained on manually labeled facial landmarks on an image. It also uses the probability on distance between the pairs of input pixels in order to cope with mulitple faces in the same image. The algorithm gives 99.99% accurate results. However if a face is half-hidden, the algorithm sometimes may not detect it which is reasonably fair. The output is a bounding square around the detected face and the feature points.

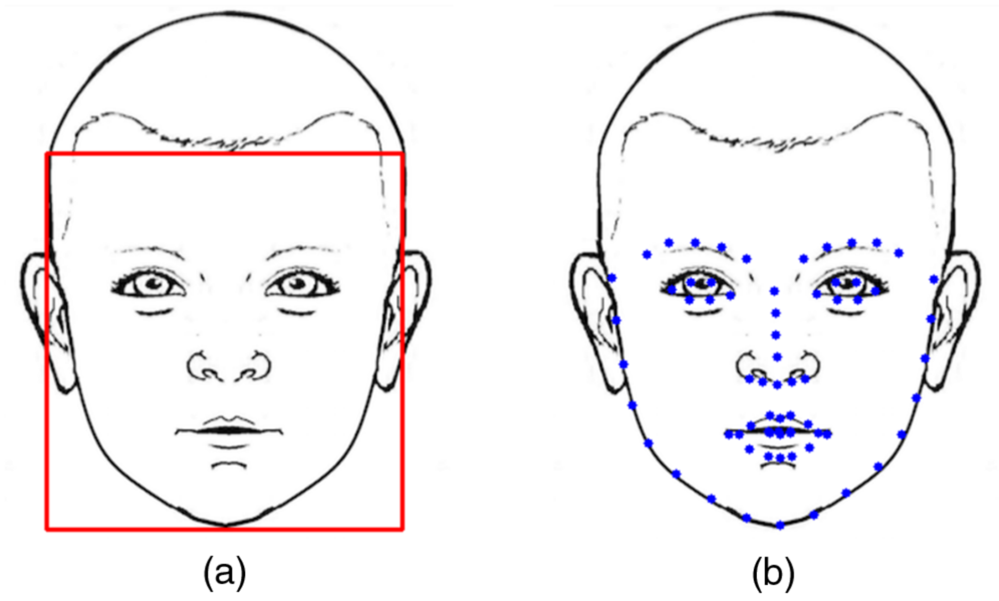
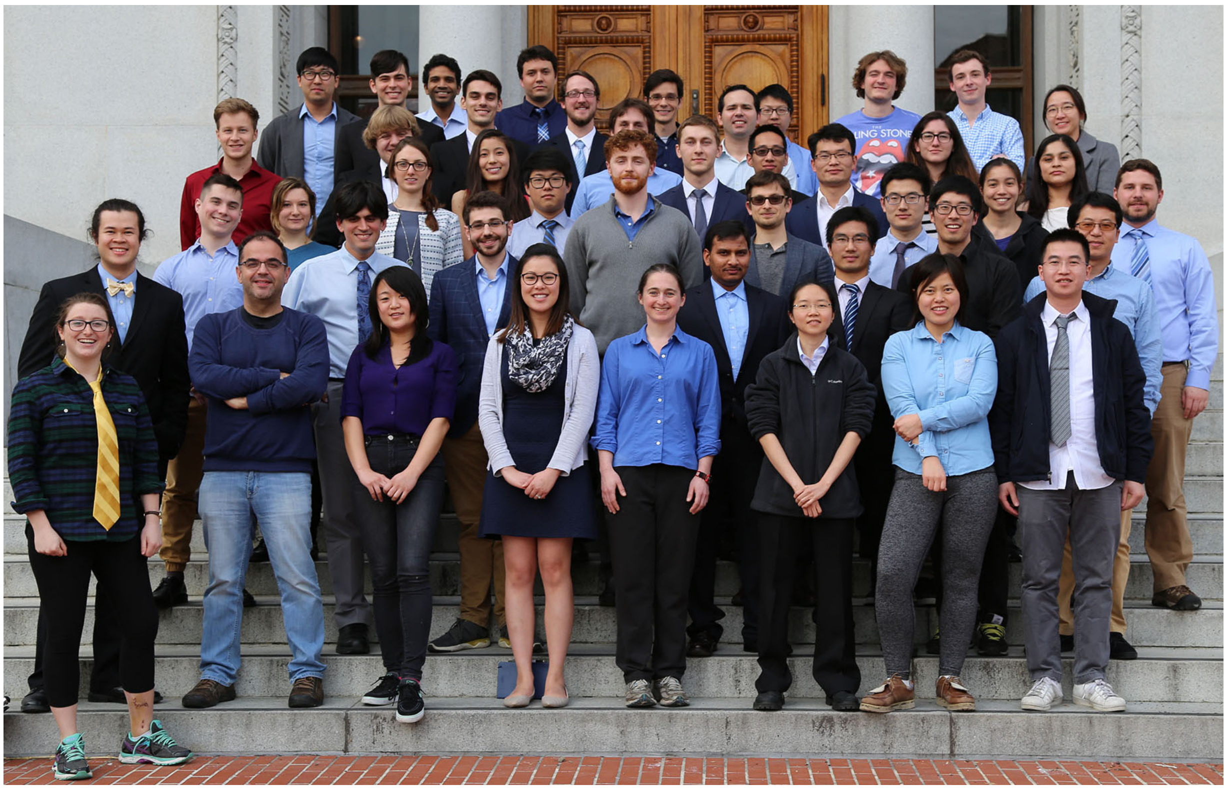


Figure 1: **Face Detection Abstract Functioning**


(a) **Input Image for face detector**

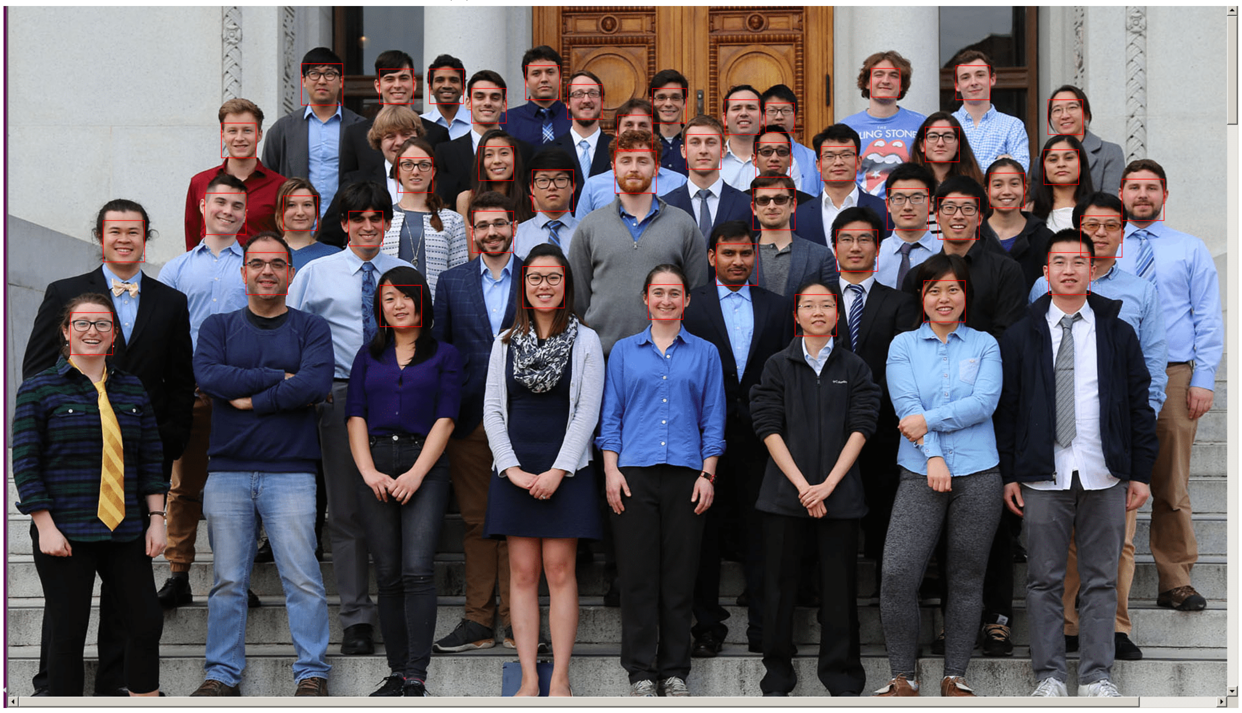

(b) **Output Image**

Figure 2: **Face Detection using dlib**

3.1.2 Face Recognition

For face recognition we aim to use the dlib library. The basic functions are as follows:

- **Generate Feature Vector:** The input image returns some m number of images with 128 feature points each. These feature points would be converted into a vector format for further computation.
- **Retrieve Images/Vector from Database** On the basis of courseID given as input by the android application, a student list of n students would be selected from the database and their respective feature vectors would be returned to the Python API.
- **mxn Comparisons:** The input m images will be transformed into their respective feature vectors using a previously well-trained neural network model in dlib library. These mxn comparisons yield m outputs depicting the name of the person in all of the m input image segments. The best match is found by calculating errors between the generated vectors. The maximum threshold for positive result is 0.6.

4 Application Interface

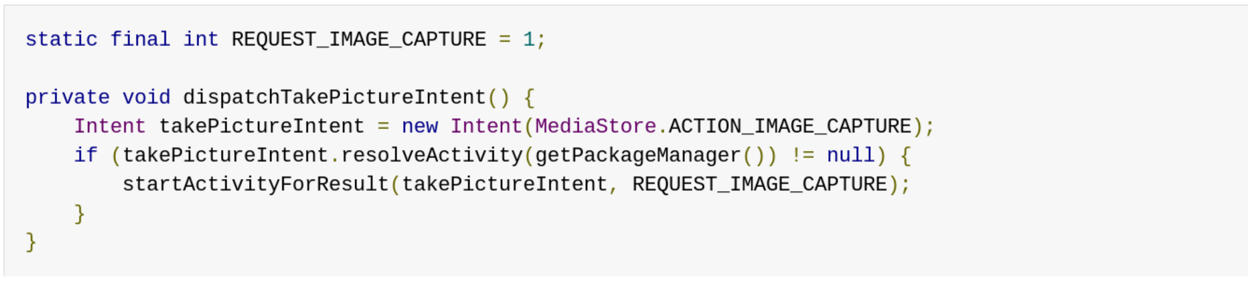
4.1 Web Interface

The web interface is primarily used for viewing the attendance data for nay course on any date. The homepage opens with a login page. On valid access it directs the professor to the courses he is currently taking in the particular semester and students to the course they have been enrolled in for the current semester. Student is given view-only access to the data while the professor can edit the attendance records in case a student has been missed by the face recognition routine. The data displayed on the webpage is retreived from the database itself allowing it to be isolated from any update overheads.

4.2 Android Interface

The home page of android app opens with an option to click a picture that directs the user to the camera of his phone and collects the clicked image which is sent to the server along with the course ID. The image is processed by the python code hosted on the server. The output is collected by the android app in the form of a string of space separated names of people detected which will be displayed on the user's android screen.

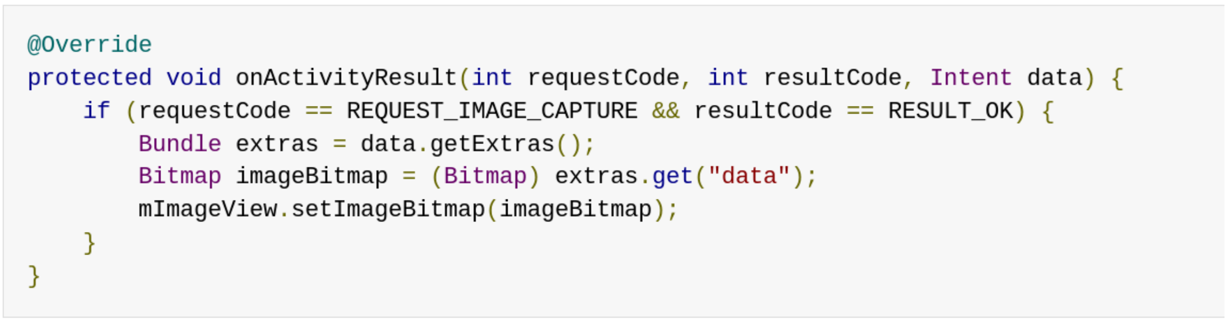
4.2.1 Code Snippets



```java
static final int REQUEST_IMAGE_CAPTURE = 1;

private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
    }
}
```

Figure 3: **Invoke Intent to Capture**



```java
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");
        mImageView.setImageBitmap(imageBitmap);
    }
}
```

Figure 4: **On click Function**

```java
String mCurrentPhotoPath;

private File createImageFile() throws IOException {
    // Create an image file name
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
    String imageFileName = "JPEG_" + timeStamp + "_";
    File storageDir = getExternalFilesDir(Environment.DIRECTORY_PICTURES);
    File image = File.createTempFile(
        imageFileName,  /* prefix */
        ".jpg",         /* suffix */
        storageDir      /* directory */
    );

    // Save a file: path for use with ACTION_VIEW intents
    mCurrentPhotoPath = image.getAbsolutePath();
    return image;
}
```

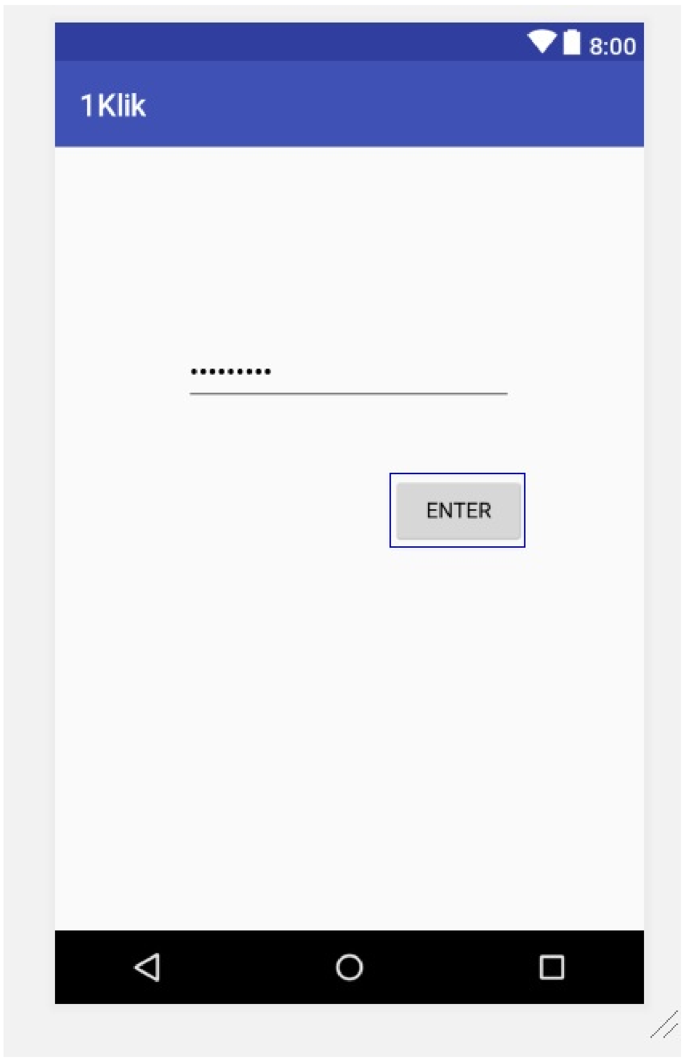
Figure 5: **Create image file**

```java
private void galleryAddPic() {
    Intent mediaScanIntent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
    File f = new File(mCurrentPhotoPath);
    Uri contentUri = Uri.fromFile(f);
    mediaScanIntent.setData(contentUri);
    this.sendBroadcast(mediaScanIntent);
}
```

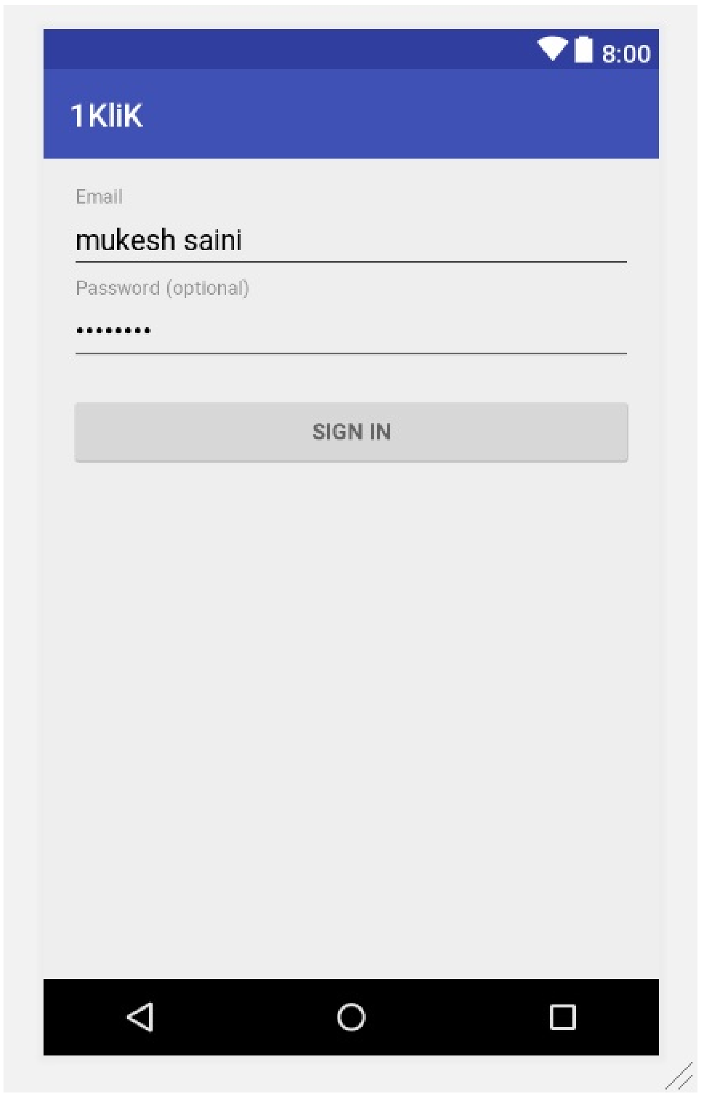
Figure 6: **Save picture to image gallery**

4.2.2 Android Windows

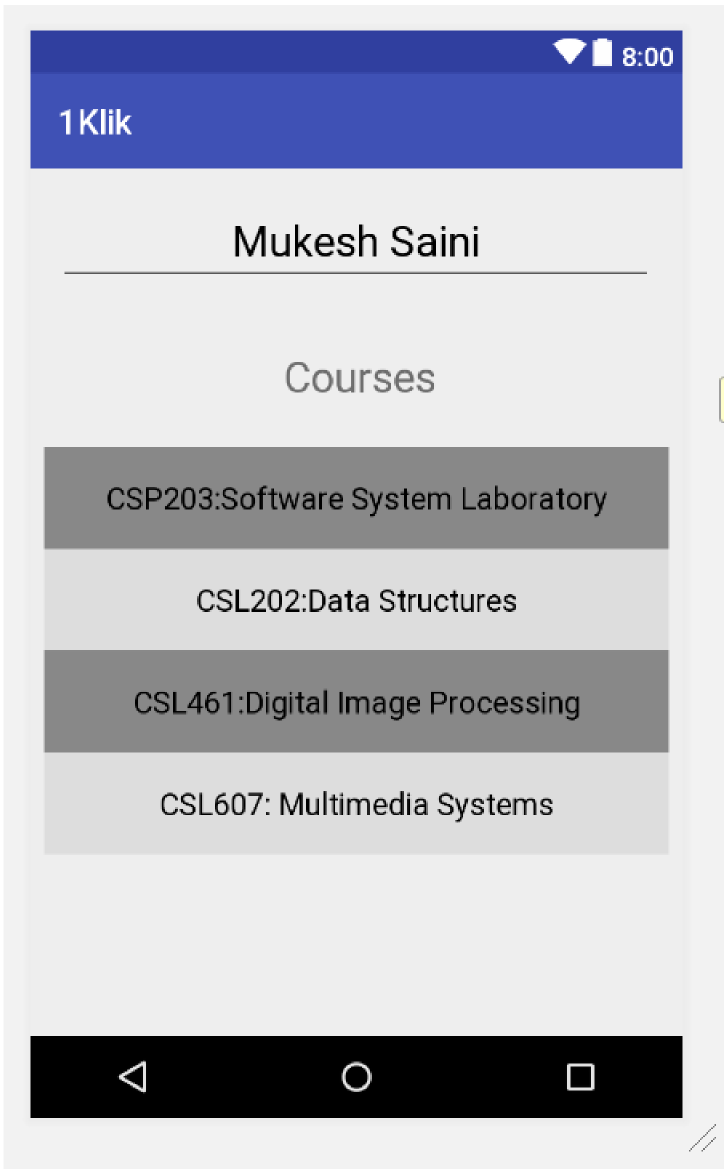
The android app works as follows:



The homepage requires the user to enter a passKey that is common to all the professors and known only to them. This gives basic access to the application and prevents any non-professor user to access the system. Hence, this passKey protects the application.



On valid access to the application the professor enters his unique ID and password to view his courses and do the further processing.


8:00

1Klik

Mukesh Saini

Courses

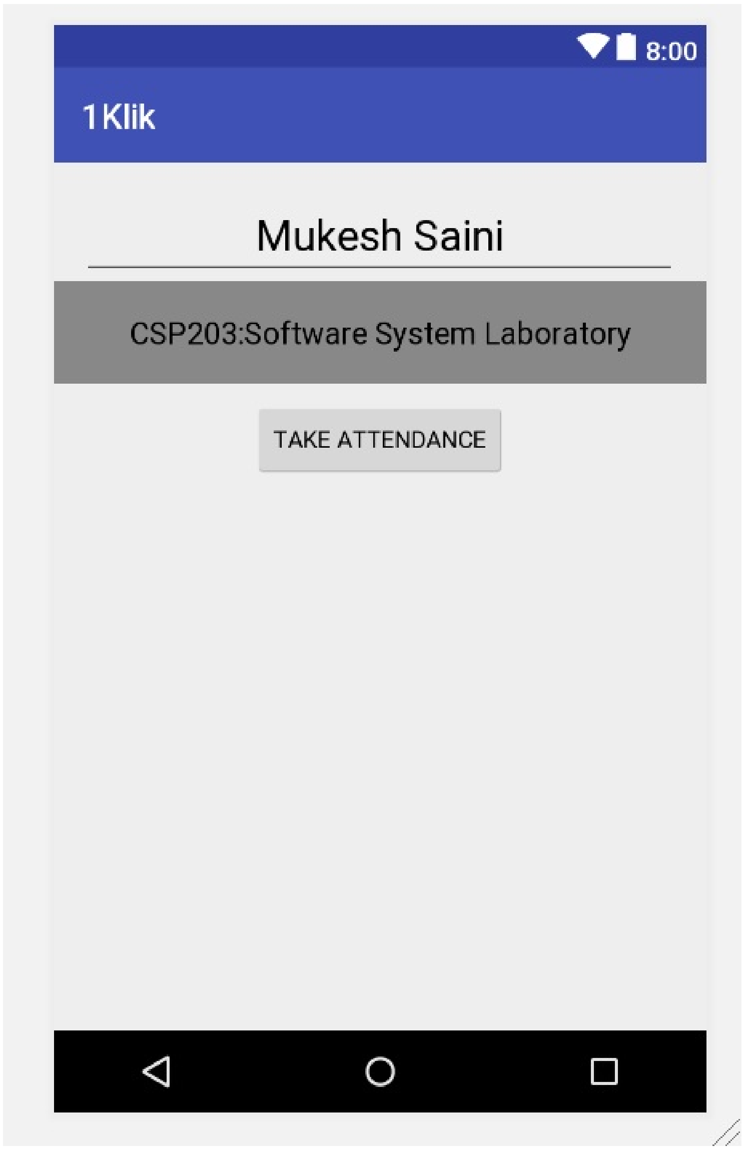
CSP203:Software System Laboratory

CSL202:Data Structures

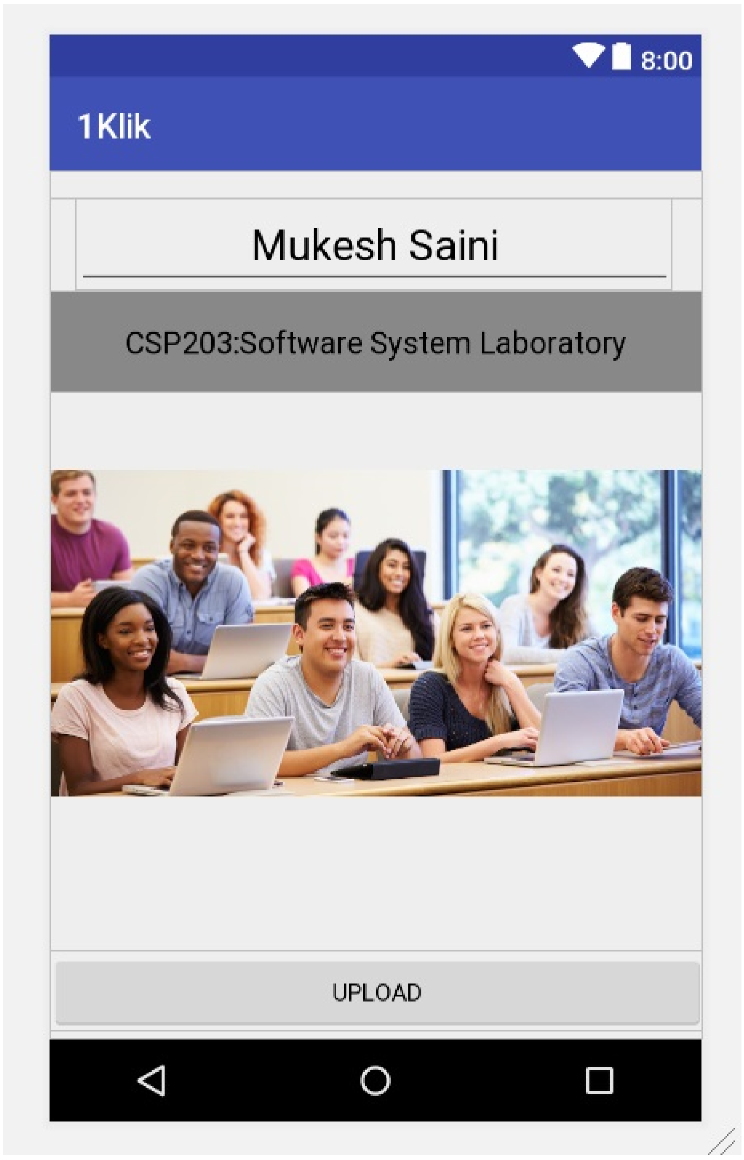
CSL461:Digital Image Processing

CSL607: Multimedia Systems

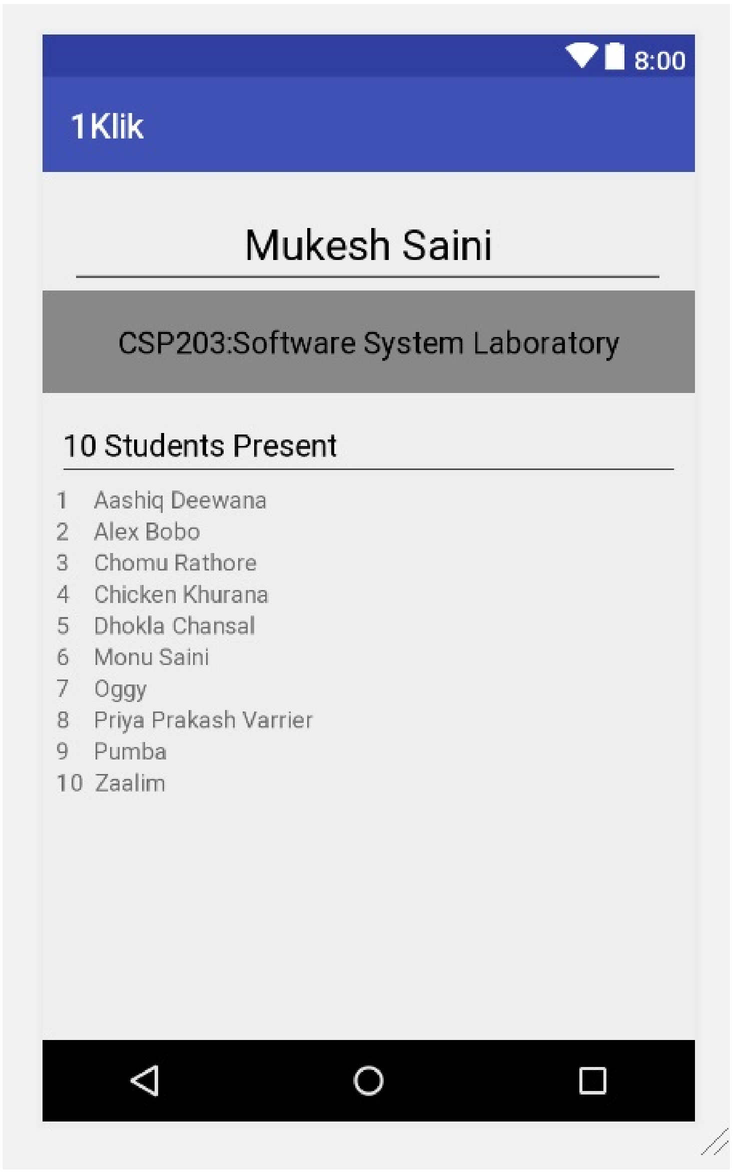
Entering a valid ID would directs the professor to the list of courses he is taking in that particular semester. He can select the course he wants to take attendance of.



In this window clicking on take Attendance would invoke the camera of his phone and hence the functioing would begin. The image would be captured and prepared to be uploaded.



This window displays the captured image and upload photo option would send the image to the server for face detection.



Mukesh Saini

CSP203:Software System Laboratory

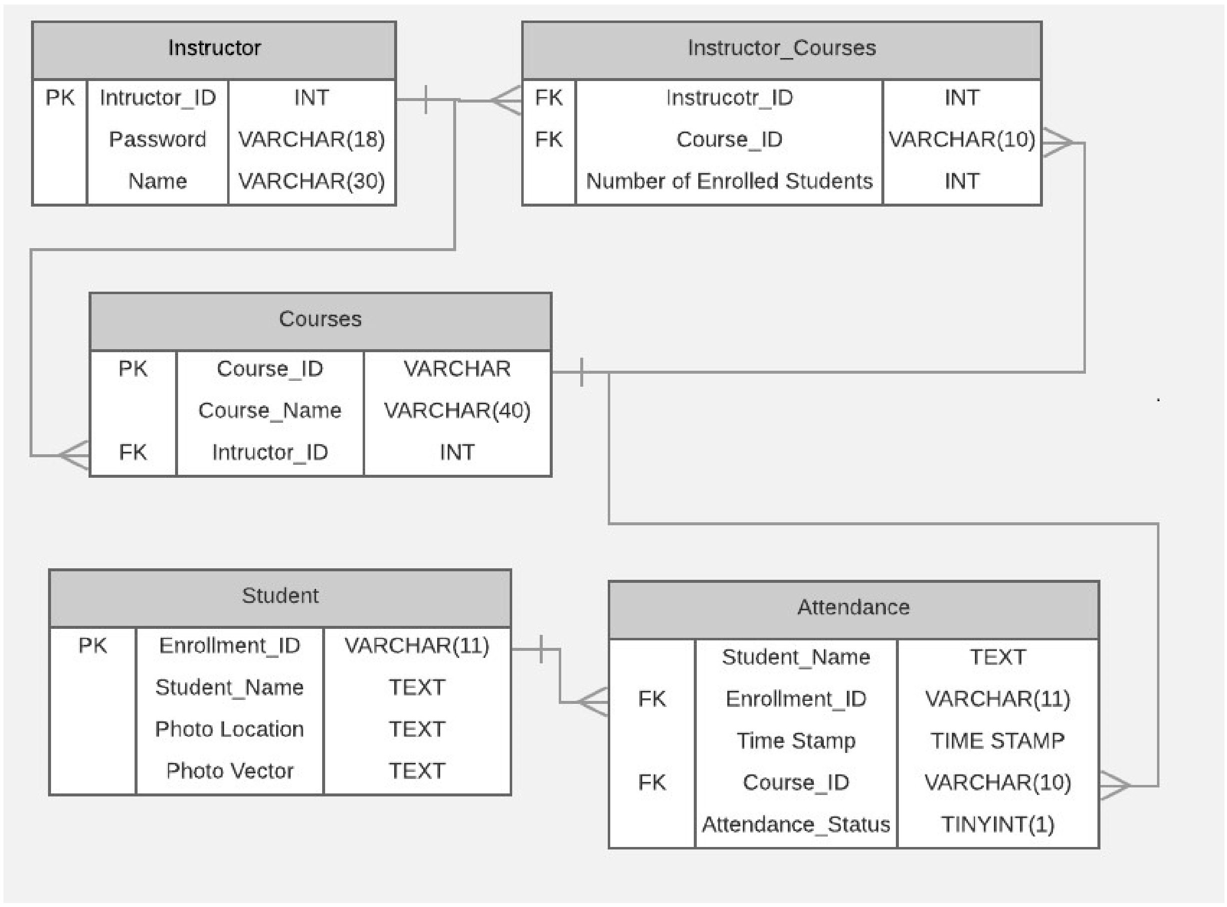
10 Students Present

1 Aashiq Deewana
2 Alex Bobo
3 Chomu Rathore
4 Chicken Khurana
5 Dhokla Chansal
6 Monu Saini
7 Oggy
8 Priya Prakash Varrier
9 Pumba
10 Zaalim

Finally the server would return a list of names of the faces detected using the existing data from the database and display the results in the form of list.

4.3 Database Model

The database stores the data of all instructors, students and courses using permanent join tables for serving many to many relationships. The join tables have been made a permanent part of the database as the join query is used everytime the database is invoked. The admin is responsible for creating/updating the tables,columns etc. into the database. The basic entities are instructors, students and courses. Rest of the entities are join tables. The model has been shown below:



Instructor		
PK	Intructor_ID	INT
	Password	VARCHAR(18)
	Name	VARCHAR(30)

Instructor_Courses		
FK	Instrucotr_ID	INT
FK	Course_ID	VARCHAR(10)
	Number of Enrolled Students	INT

Courses		
PK	Course_ID	VARCHAR
	Course_Name	VARCHAR(40)
FK	Intructor_ID	INT

Student		
PK	Enrollment_ID	VARCHAR(11)
	Student_Name	TEXT
	Photo Location	TEXT
	Photo Vector	TEXT

Attendance		
	Student_Name	TEXT
FK	Enrollment_ID	VARCHAR(11)
	Time Stamp	TIME STAMP
FK	Course_ID	VARCHAR(10)
	Attendance_Status	TINYINT(1)

Figure 7: **Entity Relationship Diagram**

4.4 Server Interface

The server is the brain of the application. The python routines for face detection, feature extraction and face recognition are hosted on the server.

1. **Interaction with Android Interface** It takes an image as input from the android app, and feeds to the face detection algorithm. The algorithm processes the image and returns a python dictionary consiting of face boundary boxes as well as pixels.

2. **Interaction with Database** The database sends a set of images with name labels to the server as per the course ID given as input by the user. The python face recognition routine perform the comparisons and yields results for the input segments and updates the results in the database using flask on the basis of timestamp.

5 Learning Sources

- Flask Tutorial
- dlib Library
- Android Documentation

6 Memberwise Role

Our Project has the following broad to-do list:

1. Android Application
2. Web Application
3. Neural Networks
4. Flask and Database programming

6.1 Aditya Tiwari:

Android and Web Application Programming

6.2 Prerna Garg:

Neural Networks and Web Application Programming

6.3 Vineet Mehta:

Flask and Web Application Programming

6.4 Himanshu Parihar:

Database Modelling