

A Project Report on
IdClick: Identify Student Digitally

Submitted in partial fulfillment of the requirements for
the award of the degree of

Bachelor of Engineering

in

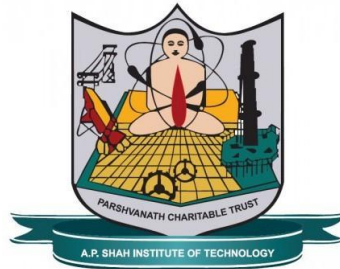
Information Technology

By

Karan Thakkar(17104039)
Arun Pandey(17104020)
Gunasekar Naikar(17104055)

Under the Guidance of

Ms. Rujata Chaudhari
Ms. Geetanjali Kalme



Department of Information Technology

NBA Accredited

A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W), Mumbai- 400615
UNIVERSITY OF MUMBAI

Academic Year 2020-2021

Approval Sheet

This Project Report entitled ***“IdClick: Identify Student Digitally”*** Submitted by ***“Karan Thakkar”(17104039), “Arun Pandey”(17104020), “Gunasekar Naikar”(17104055)*** is approved for the partial fulfillment of the requirement for the award of the degree of ***Bachelor of Engineering*** in ***Information Technology*** from ***University of Mumbai*** .

Ms.Geetanjali Kalme
Co-Guide

Ms.Rujata Chaudhari
Guide

Prof. Kiran Deshpande
Head Deartment of Information Technology

Place: A.P.Shah Institute of Technology, Thane
Date:

CERTIFICATE

This is to certify that the project entitled ***“IdClick: Identify Student Digitally”*** submitted by ***“Karan Thakkar”(17104039), “Arun Pandey”(17104020), “Gunasekar Naikar”(17104055)*** for the partial fulfillment of the requirement for award of a degree ***Bachelor of Engineering in Information Technology*** ,to the University of Mumbai,is a bonafide work carried out during academic year 2020-2021.

Ms.Geetanjali Kalme
Co-Guide

Ms.Rujata Chaudhari
Guide

Prof. Kiran Deshpande
Head Department of Information Technology

Dr. Uttam D.Kolekar
Principal

External Examiner(s)

1.

2.

Place:A.P.Shah Institute of Technology,Thane

Date:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Karan Thakkar 17104039

Arun Pandey 17104020

Gunasekar Naikar 17104055

Date:

Abstract

Face recognition has been one of the most interesting and important research fields in the past two decades. The reasons come from the need of automatic recognitions and surveillance systems, the interest in human visual system on face recognition, and the design of human-computer interface, etc. These researches involve knowledge and researchers from disciplines such as neuroscience, psychology, computer vision, pattern recognition, image processing, and machine learning, etc. A bunch of papers have been published to overcome difference factors (such as illumination, expression, scale, pose,) and achieve better recognition rate, while there is still no robust technique against uncontrolled practical cases which may involve kinds of factors simultaneously. A face recognition system is one of the biometric information processes, its applicability is easier and working range is larger than others, i.e.; fingerprint, iris scanning, signature, etc. It is known to be reliable, effective, secure then why not use it for college as well. In this project we will be developing a face recognition application which will give details about the students by capturing their face. In this report we will see how face recognition is implemented, how this we help staff & teacher in identifying students. For our project, we will be implementing face recognition using reactjs and nodejs which are two most popular frameworks and also the faceapi.js which helps us in detection and recognition of face without actually training our own model with thousands of image. The application will reduce the work of staff to identify a student or an unknown entity seeking permission for access to the college premises.

Contents

1.Introduction	1
1.1 Scope.....	2
1.2 Objective.....	2
1.3 Overview.....	3
2. Literature Review	5
3. Proposed System	6
3.1 Step By Step Process.....	8
3.1: Input Image.....	8
3.2: Face Detection.....	8
3.3: Extraction.....	9
3.4: Face Recognition.....	9
3.5: Identification.....	10
3.2 Project Scheduling.....	11
4. Design	
4.1: Use-case Diagram.....	12
4.2: Activity Diagram.....	14
4.2: Sequence Diagram.....	15
5. Implementation	16
6. Testing	26
7. Result	27
8. Conclusions and Future Scope	34
Bibliography	35
Appendices	36
Appendix-A: Install Node.js and NPM.....	36

List of Figures

1.1 : Configuration of a general face recognition structure.....	1
2.1 : Configuration of a generic face recognition system.....	5
3.1: Face Recognition Process.....	8
3.1: An example of how the three steps work on an input image.....	9
3.2: Gantt Chart.....	11
4.1.1: Use-Case for Users.....	12
4.1.2: Use-Case for Service Provider.....	13
4.2: Activity Diagram.....	14
4.3: Sequence Diagram.....	15

List of Tables

6.1 Functionality Test	26
------------------------------	----

List of Abbreviations

- 1) 128-d array: A list of 128-dimensional face encodings (one for each face in the image)
- 2) CNN: Convolutional neural network
- 3) MTCNN: Multi-Task Cascaded Convolutional Neural Network

Chapter 1

Introduction

Face recognition is the task of identifying an already detected object as a known or unknown face. Often the problem of face recognition is confused with the problem of face detection. Face Recognition on the other hand is to decide if the "face" is someone known, or unknown, using for this purpose a database of faces in order to validate this input face. Face recognition system is a complex image-processing problem in real world applications with complex effects of illumination, occlusion, and imaging condition on the live images. It is a combination of face detection and recognition techniques in image analysis.

In this report, we focus on image-based face recognition. Given a picture taken from a digital camera, we'd like to know if there is any person inside, where his/her face locates at, and who he/she is. Towards this goal, we generally separate the face recognition procedure into three steps: Face Detection, Feature Extraction, and Face Recognition (shown at Fig. 1).

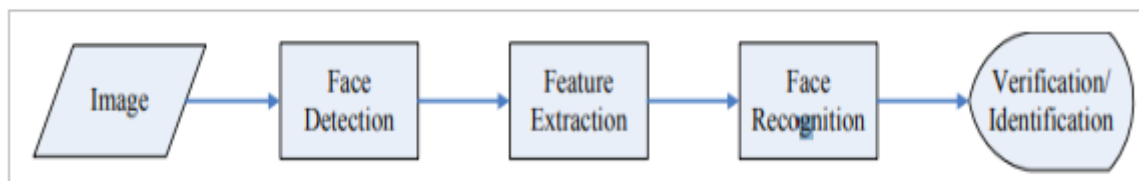


Figure1.1 : Configuration of a general face recognition structure.

Carrying a photo id on campus may soon become a thing of the past as advances in artificial intelligence have paved the way for making facial recognition technology available and worth implementing on campus. The three main steps include in this project are:

1. Detection: Identifying whether face is present in an image or not.
2. Embeddings: To extract the 128-d feature vectors that quantify each face in an image.
3. Recognizer: This is done which face recognition model which is provide in faceapi.js.

The project is a WebApp where the intended user can upload image of a student and identify him/her if already registered or can add new person details. The project aims to overcome the problems faced in reality like student entering without Id card into college or some visitor entering into college without actually entering his/her details at the gate. This WebApp aims to help college staff & teachers to help identify student if they are found misbehaving or not

wearing an Id card and can report that student to respective HOD.

The intended users for this project are teachers, college staff who wants to know the details of student with just a photo click.

1.1 Scope

In reality, if a student enters into college without Id Card they have to write their details into registration book which wastes time of student as well as gate personnel for which gate staff can use our app. Second, if teachers find student misbehaving inside campus or bunking lectures instead of arguing with the student, teachers can click student photo can see their name and can report that to respective HOD. Using our system, all this would be possible. Also, sometimes visitor come to meet faculties and they have to register their details on gate which is waste of time, what gate personnel can do is click their photo, add their name and can send to respectively faculties who he/she has come to meet.

1.2 Objective

In this project we aim to build a face recognition app to help teachers and other college staff to identify students just by taking a snap of student face. Specific objective of this project is:

1 Platform for Teachers & College Staff:

Using IdClick Application for identifying student effortlessly can be very beneficial to intended user. A student will be free of registering themselves on gate when they forget their Id card.

2 Visitors from outside:

Consider a situation where an unknown person wants an entry into the institution he or she should be verifiable including their purpose of visit. The security personnel's can also identify strangers who are not students but might be seeking permission to the college premises.

3 Authenticate student:

If we are able to provide the information to the concerned teacher and verify student identity it leads to informed decision making.

4 Cost Effective:

Since all the data will be saved into a JSON file we don't have extra cost of databases to store

data. Also the image details will be stored in form of numeric array, so this also reduces extra space, in spite of storing actual image of students.

5 Time Effective:

Sometimes teachers or staff need to take action against other department students but they don't know their proper details, which wastes their time asking students their details. For this there must be proper and secured method to identify students and take actions accordingly.

1.3 Overview

The project is nothing but an WebApp where indented users can identify students based on their photo. This app will have an interface to upload image, if the person is not registered they can register them and if already registered name of the person will be shown. Additionally there is also an functionality where user can send details to whats app whosoever already registered.

The following software's & hardware and libraries were used to create the Id-click Application:

Hardware

System with

1 GB RAM

80 GB HARD DISK

AMD OR INTEL PROCESSOR

Software

1 VS Code:

Visual Studio Code is a freeware source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

Here are the Visual Studio Code system requirements (minimum):

1. 1.6 GHz or faster processor
2. 1 GB of RAM
3. Microsoft .NET Framework 4.5.2 is required for vs code. If you are using windows 7, please make sure .NET Framework 4.5.2 is installed.

Libraries

1 ReactJs:

React is an open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications.

2 Nodejs:

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.

3 Faceapi.js:

JavaScript API for face detection and face recognition in the browser implemented on top of the tensorflow.js core API.

4 Twilio:

Twilio allows software developers to programmatically make and receive phone calls, send and receive text messages, and perform other communication functions using its web service APIs.

Chapter 2

Literature Review

Authors W. ZHAO, R. CHELLAPPA, P. J. PHILLIPS, A. ROSENFELD have published a paper in 2003 ACM Computing Surveys, Vol. 35 entitled [“Face Recognition: A Literature Survey”](#). The authors in this have explained steps as shown in below image to detect and recognition faces from still image as well as from video.

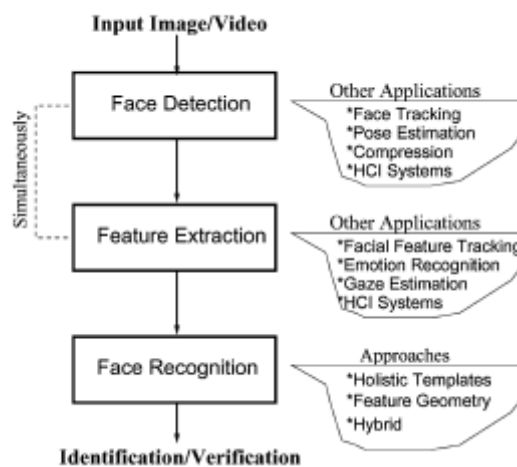


Figure2.1 : Configuration of a generic face recognition system

As illustrated in Figure 2.1, the problem of automatic face recognition involves three key steps/subtasks: (1) detection and rough normalization of faces, (2) feature extraction and accurate normalization of faces, (3) identification and/or verification. Sometimes, different subtasks are not totally separated. For example, the facial features (eyes, nose, mouth) used for face recognition are often used in face detection. Face detection and feature extraction can be achieved simultaneously, as indicated in Figure 1. Depending on the nature of the application, for example, the sizes of the training and testing databases, clutter and variability of the background, noise, occlusion, and speed requirements, some of the subtasks can be very challenging.

Learning from this paper we have used same 3 steps with few modification in this method and also new tech stake.

Chapter 3

Proposed System

Given a picture taken from a digital camera, we'd like to know if there is any person inside, where his/her face locates at, and who he/she is. Towards this goal, we generally separate the face recognition procedure into three steps: Face Detection, Feature Extraction, and Face Recognition. Now, let imagine when you go some government office and ask for a copy of your personal document. The officer behind the counter will usually ask you to proof who you are. You show them your ID card. She looks at your name and photo, then check your face, make sure you are the same person you claim to be.

For this project we have used the following models provided by face-api.js:-

1. **Tiny Face Detector:-** The Tiny Face Detector is a very performant, realtime face detector, which is much faster, smaller and less resource consuming compared to the SSD Mobilenet V1 face detector, in return it performs slightly less well on detecting small faces.
2. **Face Landmark Network:-** This package implements a very lightweight and fast, yet accurate 68 point face landmark detector. The default model has a size of only 350kb.
3. **Face Descriptor:-** Face Descriptor is a unique value of each face. Face Descriptors of same person from different image sources should be very close when we compare them.
4. **Face Recognition Network:-** For face recognition, a ResNet-34 like architecture is implemented to compute a face descriptor (a feature vector with 128 values) from any given face image, which is used to describe the characteristics of a persons face.

Likewise, facial recognition system should already stored your name altogether with your reference facial information. Then when you feed another photo to identify the system will, firstly try to detect if any face present on the image, at this step Face Detection Network will do the work. The model we use in this project is **Tiny Face Detector**, for its tiny size and mobile friendly. (The API also provide SSD mobileNet and MTCNN for face detector but let forget about them for now.)

Back to our system. Once a face (or faces) detected, face detector model will return with bounding boxes of each face, telling us where the face is in the image. We then use **Face Landmark**

Network to mark 68 points face landmark and use alignment-model to make sure the face is centered before feed to **Face Recognition Network**. The **Face Recognition Network** is another neural network (ResNet-34 like neural network, to be precise) return a **Face Descriptor** (feature vector contain 128 values) that we can use to compare and identify person in the image. Just like fingerprint, **Face Descriptor** is a unique value of each face. Face Descriptors of same person from different image sources should be very close when we compare them. In this project we use Euclidean Distance to compare. If the distance less than threshold that we set, we determine that they are likely to be same person. (the lower the distance, the higher confident).

Usually, the system will store **Face Descriptor** of each person as reference together with his or her name as label. When we feed a query image, the system will compare Face Descriptor of new image with all reference Descriptors and identify the person with the lowest one. If none of comparison lower than the threshold, the person will be identified as Unknown.

In this project, we will make Single Page App with React and face-api.js library to detect and recognize the Idol face. Since [Vincent](#) did all the hard parts for us in his API that comes with pre-trained face detection, face-landmarks, face-alignment, and face-recognition models, so we don't have to train models by ourselves. We don't even need to write DL model in TensorFlow either. Indeed, you don't really need to know how Deep-learning or CNN work to make this App. All you need to know is at least basic concept of JavaScript and React.

In order to build our face recognition pipeline, we'll be applying faceapi.js in two key steps:

- 1.To apply face detection, which detects the presence and location of a face in an image, but does not identify it.
- 2.To extract the 128-d feature vectors (called embeddings) that quantify each face in an image.

3.1: Step by step process:

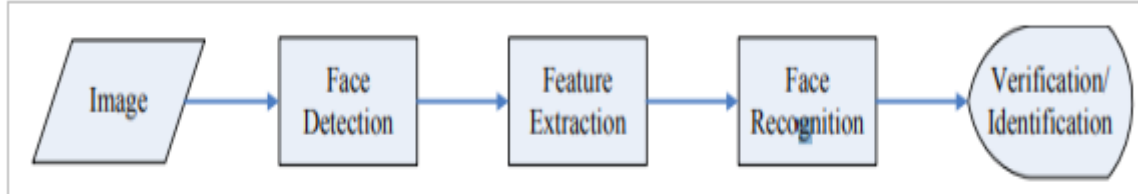


Figure 3.1: Face Recognition Process

3.1: Input Image:

First step is to have set of images for identification. This set should contain images which are clear, face aligned to center and its face is been registered 1st time then image should contain only single face not multiple.

3.2: Face Detection:

The main function of this step is to determine (1) whether human faces appear in a given image, and (2) where these faces are located at. The expected outputs of this step are patches containing each face in the input image. In order to make further face recognition system more robust and easy to design, face alignment are performed to justify the scales and orientations of these patches. Besides serving as the pre-processing for face recognition, face detection could be used for region-of-interest detection, re-targeting, video and image classification, etc.

This is done using pretrained model which is Tiny Face Detector. The Tiny Face Detector is a very performant, realtime face detector, which is much faster, smaller and less resource consuming compared to the SSD Mobilenet V1 face detector, in return it performs slightly less well on detecting small faces. This model is extremely mobile and web friendly, thus it should be your GO-TO face detector on mobile devices and resource limited clients. The size of the quantized model is only 190 KB (**tiny_face_detector_model**).

3.3: Extraction:

In this we process image with ‘68 Point Face Landmark Detection Model’. This package implements a very lightweight and fast, yet accurate 68 point face landmark detector.

The default model has a size of only 350kb(**face_landmark_68_model** and the tiny model is only 80kb (**face_landmark_68_tiny_model**). Both models employ the ideas of depthwise separable convolutions as well as densely connected blocks. The models have been trained on a dataset of ~35k face images labeled with 68 face landmark points.

3.4: Face Recognition:

After formulating the representation of each face, the last step is to recognize the identities of these faces. Then when an input face image comes in, we perform face detection and feature extraction, and compare its feature to each face class stored in the database. There are two general applications of face recognition, one is called identification and another one is called verification. Face identification means given a face image, we want the system to tell who he / she is or the most probable identification; while in face verification, given a face image and a guess of the identification, we want the system to tell true or false about the guess. In fig. 3.1, we show an example of how these three steps work on an input image.

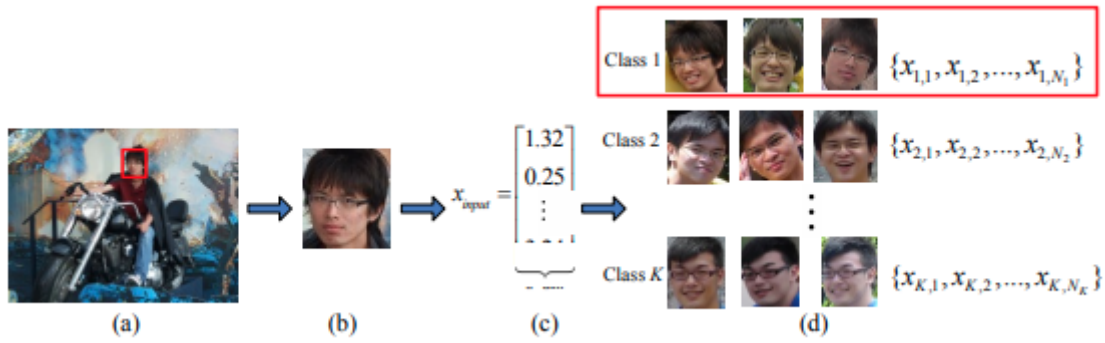


Figure 3.1: : An example of how the three steps work on an input image

- (a) The input image and the result of face detection (the red rectangle).
- (b) The extracted face patch.
- (c) The feature vector after feature extraction.
- (d) Comparing the input vector with the stored vectors in the database by classification techniques and

determine the most probable class (the red rectangle). Here we express each face patch as a d-dimensional vector, the vector as the , and as the number of faces stored in the.

In this after all the findings are gathered such as landmark,detection we furthur calculate 128-d of face and recognition basics on it who the person is in the image. For this we have used 'Face Recognition Model' which is provided in faceapi.js. For face recognition, a ResNet-34 like architecture is implemented to compute a face descriptor (a feature vector with 128 values) from any given face image, which is used to describe the characteristics of a persons face. The model is **not** limited to the set of faces used for training, meaning you can use it for face recognition of any person, for example yourself. You can determine the similarity of two arbitrary faces by comparing their face descriptors, for example by computing the euclidean distance or using any other classifier of your choice.

The neural net is equivalent to the **FaceRecognizerNet** used in [face-recognition.js](#) and the net used in the [dlib](#) face recognition example. The weights have been trained by [davisking](#) and the model achieves a prediction accuracy of 99.38% on the LFW (Labeled Faces in the Wild) benchmark for face recognition.

The size of the quantized model is roughly 6.2 MB (**face_recognition_model**).

3.5: Identification:

Finally after the above steps we have our 128-d array of the person, with help of this if the different image is uploaded of the same person we can match newly calculated array & previously saved array and match both the array, if we get threshold above 0.5 we can say they are same person or not.

3.2 Project Scheduling

PROJECT TITLE		iClick: Identify Students Digitally				INSTITUTE & DEPARTMENT NAME		A.P Shah Institute of Technology																						
PROJECT GUIDE		Mr. Rajata Chaudhary				DATE		2/10/21																						
WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION (Weeks)	PCT OF TASK COMPLETE	PHASE ONE			PHASE TWO			PHASE THREE			PHASE FOUR														
							WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12												
							M	T	W	R	F	S	M	T	W	R	F	S	M	T	W	R	F	S	M	T	W	R	F	S
1 Project Conception and Initiation																														
1.2	Project Title	Karan,Arun,Gunasekar	8/25/20	8/28/20	3	100%																								
1.3	Abstract	Gunasekar	8/26/20	8/27/20	1	100%																								
1.4	Objectives	Karan,Arun,Gunasekar	10/25/20	8/28/20	1	100%																								
1.5	Problem Definition	Karan	8/27/20	8/29/20	1	100%																								
1.6	Scope	Arun	8/28/20	8/28/20	1	100%																								
1.7	Technology stack	Karan,Gunasekar	8/26/20	8/31/20	1	80%																								
1.8	Benefits for society	Arun	8/29/20	8/29/20	1	80%																								
1.11	Applications	Karan	8/29/20	8/31/20	1	100%																								
2 Project Design																														
2.1	Proposed System	Karan, Arun	9/18/20	10/6/20	2	70%																								
2.2	Design(Flow Of Modules)	Karan,Arun,Gunasekar	9/19/20	10/6/20	2	70%																								
2.3	Modules																													
2.3.1	Module-1	Karan,Arun,Gunasekar	9/19/20	10/6/20	2	100%																								
2.3.2	Module-2	Gunasekar	9/19/20	10/6/20	2	100%																								
2.3.3	Module-3	Arun	9/19/20	10/6/20	2	100%																								
2.3.4	Module-4	Karan	9/19/20		2	40%																								
2.4	Preparation Of Report	Gunasekar	9/19/20	10/6/20	2	75%																								
3 Project Implementation																														
3.1	Module-1	Karan,Arun,Gunasekar	9/19/20	10/6/21	0	100%																								
3.2	Module-2	Gunasekar	9/19/20	10/6/21	0	100%																								
3.3	Module-3	Arun	9/19/20	10/6/21	0	100%																								
3.4	Module-4	Karan	9/19/20		0	40%																								
4 Testing																														
4.1	Design of Test Cases	Karan, Arun	10/15/20	10/21/20	1	70%																								
4.2	Architecture	Karan,Arun,Gunasekar	10/18/20	10/14/20	1	100%																								
4.3	Prototyping	Karan,Arun,Gunasekar	10/17/20	10/23/20	1	100%																								
4.4	Development	Karan	10/24/20	10/30/20	1	100%																								
4.5	Testing QA	Karan, Gunasekar	10/31/20	11/6/20	1	100%																								
4.6	Installation React js and Node.js	Karan, Arun	10/27/20	11/13/20	1	100%																								
4.7	Detection and Webpage design	Gunasekar	11/14/20	11/20/20	1	100%																								
4.8	Worked on Front end and detection	Karan	11/21/20	11/27/20	1	100%																								
4.9	Implementation of Backend	Arun	11/21/20	11/27/20	1	100%																								
4.1	Implementation of Twilio	Gunasekar	11/28/20	11/4/20	1	40%																								
4.11	Accessability of photo and name	Karan,Arun,Gunasekar	11/5/20	11/11/20	1	0%																								
5 Results and Analysis																														
5.1	Report Preparation	Arun,Gunasekar	12/12/20	12/23/20	1	70%																								

Figure3.2: Gantt Chart

Chapter 4

Design

4.1 Use-case Diagram

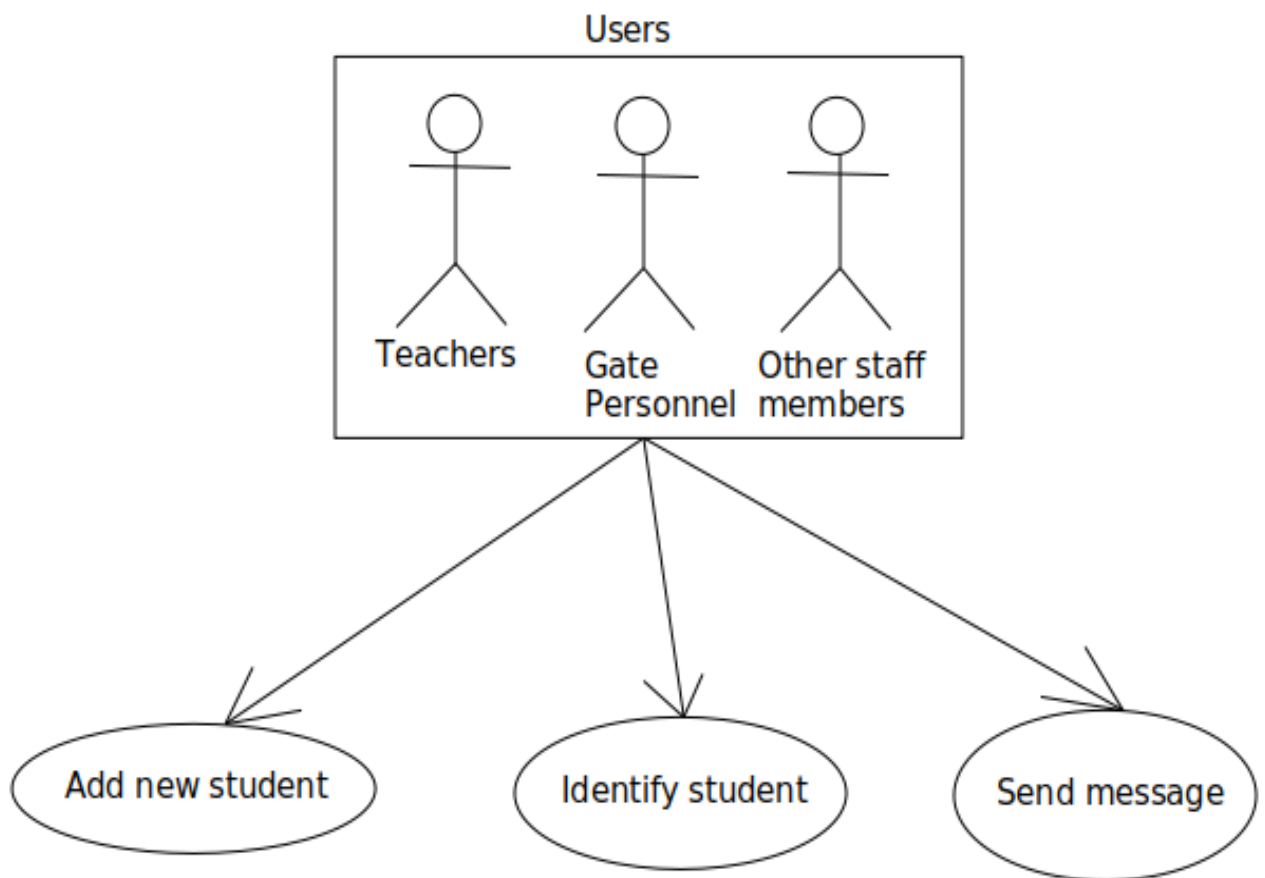


Figure 4.1.1: Use-Case for Users

Here, in the above diagram, users are teachers, gate personnel and other college staff. Above are the tasks mentioned which they can perform using this app.

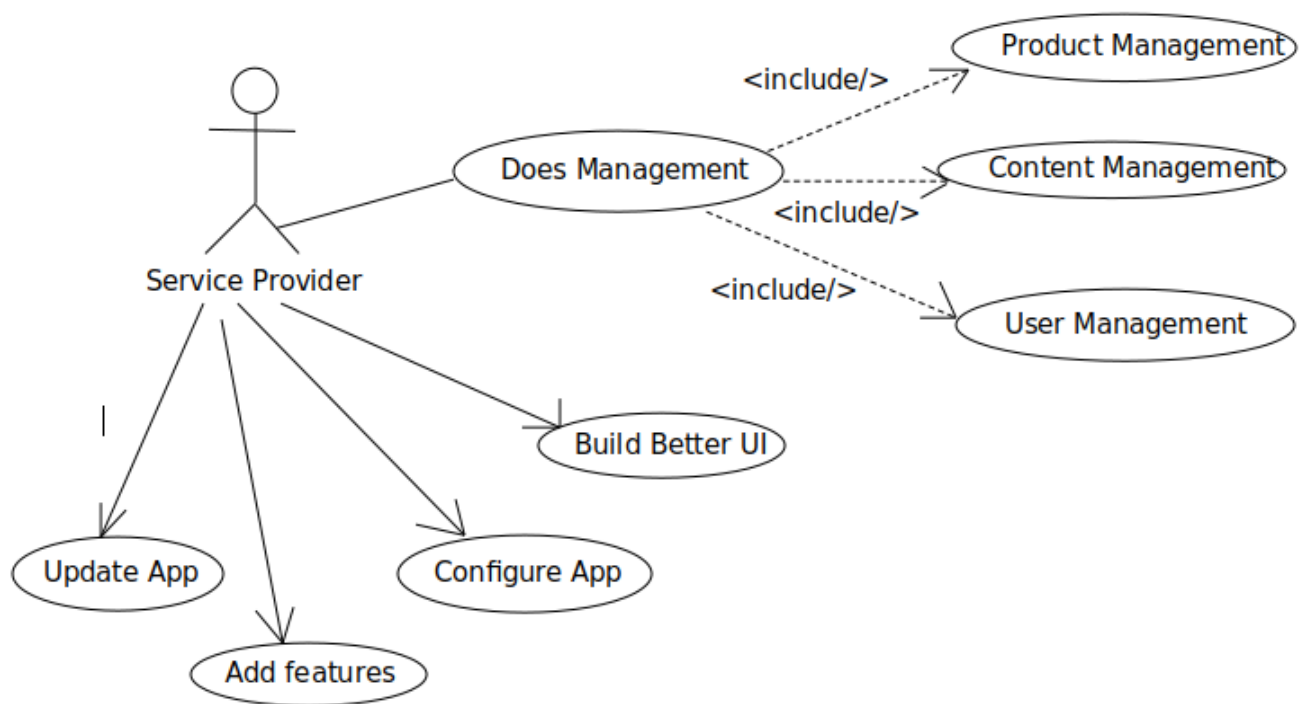


Figure 4.1.2: Use-Case for Service Provider

Here, Service Provider is the developer of the system and can do management of various services for better experience of App which includes Product Management, Content Management and User Management. He is also responsible for building new features and updating the app.

4.2 Activity Diagram

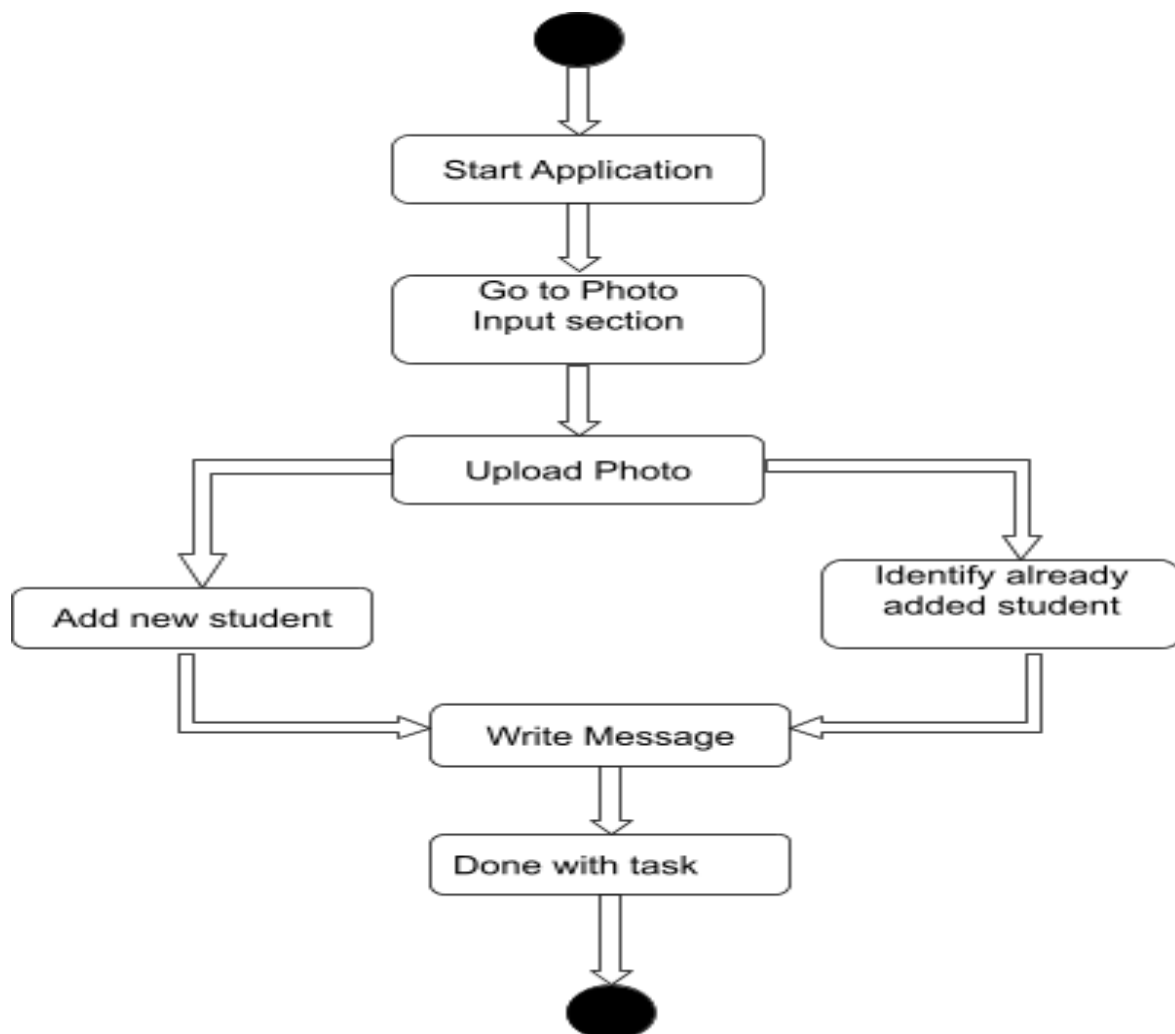


Figure 4.2: Activity Diagram

As the above activity diagram depicts, firstly the user will launch the application, then he will go to input photo section on top left where user will be able to perform all operations, then user will upload a photo and if user wants to register new student they will enter name of student and click upload or if student is already registered name will be shown on the screen. Lastly if user wants to type and send any message so they can also do that message will be send through whatsapp.

4.3 Sequence Diagram

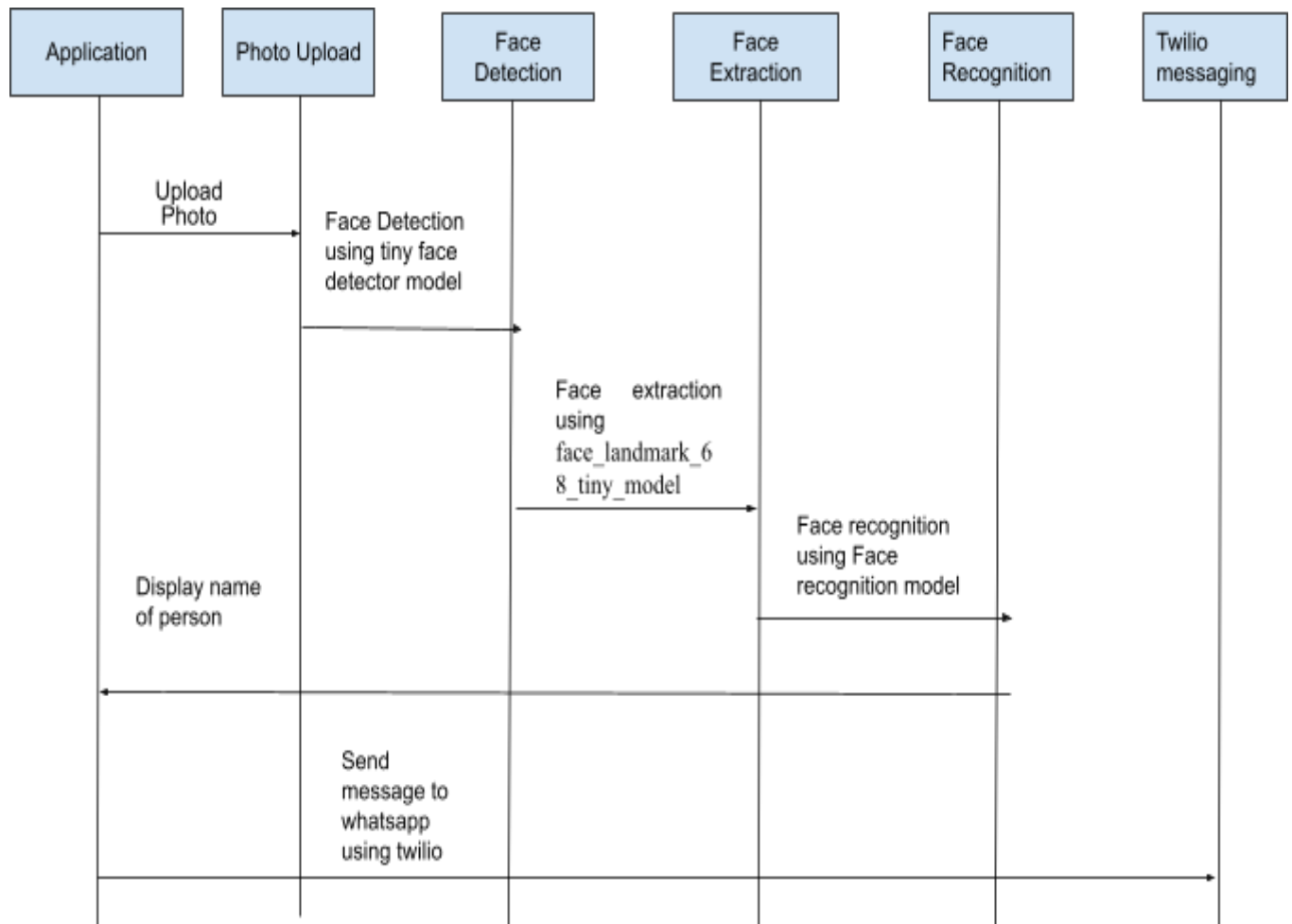


Figure 4.3: Sequence Diagram

Chapter 5

Implementation

```
1 import React, { Component } from 'react';
2 import { Route, Router } from 'react-router-dom';
3 import createHistory from 'history/createBrowserHistory';
4 import FaceRecognition from './views/faceRecognition';
5 //import CameraFaceDetect from './views/cameraFaceDetect';
6 import Home from './views/Home';
7 import Header from './components/Header';
8 import './App.css';
9
10 class App extends Component {
11   render() {
12     return (
13       <div className="App">
14         <Router history={createHistory({ basename: process.env.PUBLIC_URL })}>
15           <div className="route">
16             <Header />
17             <Route exact path="/" component={Home} />
18             <Route exact path="/photo" component={FaceRecognition} />
19             { /* <Route exact path="/camera" component={CameraFaceDetect} /> */ }
20           </div>
21         </Router>
22       </div>
23     );
24   }
25 }
26
27 export default App;
```

Home page Script

```

1 import * as faceapi from 'face-api.js';
2
3 const maxDescriptorDistance = 0.5;
4
5 export async function loadModels() {
6   const MODEL_URL = process.env.PUBLIC_URL + '/models';
7   await faceapi.loadTinyFaceDetectorModel(MODEL_URL);
8   await faceapi.loadFaceLandmarkTinyModel(MODEL_URL);
9   await faceapi.loadFaceRecognitionModel(MODEL_URL);
10 }
11
12 export async function getFullFaceDescription(blob, inputSize = 512) {
13   // tiny_face_detector options
14   let scoreThreshold = 0.5;
15   const OPTION = new faceapi.TinyFaceDetectorOptions({
16     inputSize,
17     scoreThreshold
18   });
19   const useTinyModel = true;
20
21   // fetch image to api
22   let img = await faceapi.fetchImage(blob);
23
24   // detect all faces and generate full description from image
25   // including landmark and descriptor of each face
26
27   let fullDesc = await faceapi
28     .detectAllFaces(img, OPTION)
29     .withFaceLandmarks(useTinyModel)
30     .withFaceDescriptors();
31   //console.log(fullDesc)
32   return fullDesc;
33 }
34
35 export async function createMatcher(faceProfile) {
36   // Create labeled descriptors of member from profile
37   let members = Object.keys(faceProfile);
38   let labeledDescriptors = members.map(
39     member =>
40       new faceapi.LabeledFaceDescriptors(
41         faceProfile[member].name,
42         faceProfile[member].descriptors.map(
43           descriptor => new Float32Array(descriptor)
44         )
45       )
46   );
47
48   // Create face matcher (maximum descriptor distance is 0.5)
49   let faceMatcher = new faceapi.FaceMatcher(
50     labeledDescriptors,
51     maxDescriptorDistance
52   );
53   return faceMatcher;
54 }

```

Api file Script

```

1 {
2   "Mark": {
3     "name": "Mark Zuckerberg",
4     "descriptors": [
5       [
6         -0.11509660631418228,
7         0.047304537147283554,
8         0.06996937096118927,
9         0.13533177971839905
10        ...]
11      ]
12    },
13    "Karan Thakkar": {
14      "name": "Karan Thakkar",
15      "descriptors": [
16        [
17          -0.12311732023954391,
18          0.14340874552726746,
19          ...]
20        ]
21      },
22      "Hritik Roshan": {
23        "name": "Hritik Roshan",
24        "descriptors": [
25          [
26            -0.13980115950107574,
27            0.07385019212961197,
28            0.13764673471450806,
29            -0.04106592759490013,
30            ...]
31          ]
32        ]
33      }
34    }
35  }
36 }

```

Stored data in json file

```

1 import React, { Component } from 'react';
2 import { Link } from 'react-router-dom';
3
4 class Header extends Component {
5   render() {
6     return (
7       <header>
8         <div className="Navbar">
9           <Link to="/">Home</Link>
10          <Link to="/photo">Photo Input</Link>
11          { /* <Link to="/camera">Video Camera</Link> */ }
12        </div>
13      </header>
14    );
15  }
16 }
17
18 export default Header;

```

Page Header Script

```

1 import React, { Component } from 'react';
2
3 class DrawBox extends Component {
4   constructor(props) {
5     super(props);
6     this.state = {
7       descriptors: null,
8       detections: null,
9       match: null
10    };
11  }
12
13  componentDidMount() {
14    this.getDescription();
15  }
16
17  componentWillReceiveProps(newProps) {
18    this.getDescription(newProps);
19  }
20
21  getDescription = async (props = this.props) => {
22    const { fullDesc, faceMatcher } = props;
23    if (!!fullDesc) {
24      await this.setState({
25        descriptors: fullDesc.map(fd => fd.descriptor),
26        detections: fullDesc.map(fd => fd.detection)
27      });
28      if (!!this.state.descriptors && !!faceMatcher) {
29        let match = await this.state.descriptors.map(descriptor =>
30          faceMatcher.findBestMatch(descriptor)
31        );
32        this.setState({ match });
33        //console.log(match)
34      }
35    }
36  };
37
38  render() {
39    const { imageWidth, boxColor } = this.props;
40    const { detections, match } = this.state;
41    let box = null;
42
43    if (!!detections) {
44      box = detections.map((detection, i) => {
45        const relativeBox = detection.relativeBox;
46        const dimension = detection._imageDims;
47        let _X = imageWidth * relativeBox._x;
48        let _Y =
49          (relativeBox._y * imageWidth * dimension._height) / dimension._width;
50        let W = imageWidth * relativeBox.width;

```

For box if face detected Script

```

51     let _H =
52       (relativeBox.height * imageWidth * dimension._height) /
53       dimension._width;
54     return (
55       <div key={i}>
56         <div
57           style={{
58             position: 'absolute',
59             border: 'solid',
60             borderColor: boxColor,
61             height: _H,
62             width: _W,
63             transform: `translate(${_X}px,${_Y}px)`
64           }}
65         >
66           {!!match && match[i] && match[i]._label !== 'unknown' ? (
67             <p
68               style={{
69                 backgroundColor: boxColor,
70                 border: 'solid',
71                 borderColor: boxColor,
72                 width: _W,
73                 marginTop: 0,
74                 color: '#fff',
75                 transform: `translate(-3px,${_H}px)`
76               }}
77             >
78               {match[i]._label}
79             </p>
80             ) : null}
81           </div>
82         </div>
83       );
84     });
85   }
86   // console.log("box",box.props.children.props.children.props.children)
87   //console.log("box",box)
88   return <div>{box}</div>;
89 }
90 }
91 }
92
93 export default DrawBox;

```

For box if face detected Script

```

1 |import React, { Component } from 'react';
2
3 |const exampleImage = require('../img/React.png');
4
5 |export default class Home extends Component {
6 |  render() {
7 |    const WIDTH = document.documentElement.clientWidth;
8 |    return (
9 |      <div
10 |        style={{
11 |          border: 'solid',
12 |          borderRadius: 8,
13 |          width: { WIDTH },
14 |          margin: 10,
15 |          padding: 5
16 |        }}
17 |      >
18 |        <h2>IdClick: Identify Student Digitally</h2>
19 |        <h4>
20 |          Single Page App for face detection and recognition of student
21 |          , running in front-end browser using React and{' '}
22 |          <a href="https://github.com/justadudewhohacks/face-api.js">
23 |            face-api.js
24 |          </a>{' '}
25 |          with nodejs
26 |        </h4>
27 |        <img src={exampleImage} alt="example" width="350" />
28 |
29 |        <div
30 |          style={{
31 |            display: 'flex',
32 |            flexDirection: 'column',
33 |            textAlign: 'left',
34 |            margin: 'auto',
35 |            marginLeft: 10
36 |          }}
37 |        >
38 |
39 |          <div>
40 |            <ul>
41 |              <h4>Requirement:</h4>
42 |              <li>
43 |                Support any PC browser with Javascript enabled (Chrome, IE,
44 |                Safari)
45 |              </li>
46 |              <li>Support Android phone for Photo Input</li> { /*and Video Camera</li> */ }
47 |              { /* <li>Support Iphone on Safari and Chrome only for Photo Input</li> */ }
48 |
49 |            </ul>
50 |            <ul>
51 |              <h4>Photo Input:</h4>
52 |              <li>Input image can be image file or URL</li>

```

Contents on Home page file Script

```

66     /* <li>
67     This App may not work well for older smartphone or in some
68     browsers. (I found my Iphone4 cannot process detection properly,
69     while iphone7 or 8 are working fine.)
70     </li> */
71 </ul>
72 /* <ul>
73 <h4>Video Camera:</h4>
74 <li>
75     Video Input works well with PC webcam or Android phone's camera.
76 </li>
77 <li>
78     Currently Iphone camera is not supported for live detection in
79     this App.
80 </li>
81 <li>
82     App will try to detect and recognize any faces, but performance
83     depends on CPU of the device.
84 </li>
85 <li>
86     Detection and Recognition with PC webcam can be fast, while
87     working on smartphone can be slower.
88 </li>
89 </ul> */
90 <ul>
91 <h4>Reference:</h4>
92 /* <li>
93     Find more information of my code and API in{' '}
94     <a href="https://github.com/supachaic/bnk48-face-recognition">
95         My Repo
96     </a>
97 </li> */
98 <li>
99     Tutorial for this App in my Medium:{' '}
100     <a href="https://medium.com/@supachaic/facial-recognition-spa-for-bnk48-idol-group-using-react-and-face-api-js-ad62b43ec5b6">
101         Facial Recognition SPA for BNK48 Idol group using React and
102         face-api.js
103     </a>
104 </li>
105 <li>
106     face-api.js API{' '}
107     <a href="https://github.com/justadudewhohacks/face-api.js">
108         repo
109     </a>
110 </li>
111 </ul>
112 </div>
113 </div>
114 </div>
115 );
116 }
117 }

```

Contents on Home page file Script


```

1 import React, { Component } from 'react';
2 import axios from 'axios';
3 import OnlyDesc from '../components/onlyDescriptors/onlyDescriptors';
4 import { withRouter } from 'react-router-dom';
5 import {
6   loadModels,
7   getFullFaceDescription,
8   createMatcher,
9   isFaceDetectionModelLoaded
10 } from '../api/face';
11 import DrawBox from '../components/drawBox';
12 import ShowDescriptors from '../components/showDescriptors';
13 import { JSON_PROFILE } from '../common/profile';
14
15
16 const MaxWidth = 600;
17 const boxColor = '#BE80B5';
18 const testImg = require('../img/Mark1.jpg');
19
20 let INIT_STATE = {
21   url: null,
22   imageURL: null,
23   fullDesc: null,
24   imageDimension: null,
25   error: null,
26   loading: false
27 };
28
29
30 class FaceRecognition extends Component {
31   constructor(props) {
32     super(props);
33     this.state = {
34       ...INIT_STATE,
35       faceMatcher: null,
36       showDescriptors: false,
37       WIDTH: null,
38       HEIGHT: 0,
39       isModelLoaded: !!isFaceDetectionModelLoaded()
40     };
41   }
42
43
44
45   componentWillMount() {
46     this.resetState();
47     let _W = document.documentElement.clientWidth;
48     if (_W > MaxWidth) _W = MaxWidth;
49     this.setState({ WIDTH: _W });
50     this.mounting();
51   }
52

```

Main Logic Script

```

53 mounting = async () => {
54   await loadModels();
55   await this.matcher();
56   await this.getImageDimension(testImg);
57   await this.setState({ imageURL: testImg, loading: true });
58   await this.handleImageChange(testImg);
59 };
60
61 matcher = async () => {
62   const faceMatcher = await createMatcher(JSON_PROFILE);
63   this.setState({ faceMatcher });
64   // console.log(faceMatcher._labeledDescriptors[0]._label)
65   // console.log(faceMatcher)
66 };
67
68 handleFileChange = async event => {
69   this.resetState();
70   await this.setState({
71     imageURL: URL.createObjectURL(event.target.files[0]),
72     loading: true
73   });
74   this.handleImageChange();
75 };
76
77 handleURLChange = event => {
78   this.setState({ url: event.target.value });
79 };
80
81 handleButtonClick = async () => {
82   this.resetState();
83   let blob = await fetch(this.state.url)
84     .then(r => r.blob())
85     .catch(error => this.setState({ error }));
86   if (!!blob && blob.type.includes('image')) {
87     this.setState({
88       imageURL: URL.createObjectURL(blob),
89       loading: true
90     });
91     this.handleImageChange();
92   }
93 };
94
95 sendMessage = (message) => {
96   // console.log(this.refs.UserName.value, message)
97   const waMsg = {
98     "msg" : message,
99   }
100   const URL= 'http://localhost:5000/message'
101   // axios.post(URL, waMsg)
102   // .then(response => {
103   //   //console.log(response);
104   // });

```

Main Logic Script

```

109 addDescriptor = (UserName, Desc) => {
110   var element = []
111   for (let index = 0; index < Desc.props.fullDesc[0]._descriptor.length;
112     element.push(Desc.props.fullDesc[0]._descriptor[index]));
113   }
114   if (UserName && Desc) {
115     var newUser = {
116       "name": UserName,
117       "descriptors": [element]
118     };
119     //console.log(newUser.descriptors)
120   }
121   let mergeData = [newUser];
122   //console.log(mergeData[0].descriptors)
123   this.saveUserJson(mergeData);
124   element = []
125   alert("Your data is saved")
126   //console.log(element)
127 }
128 }
129
130 saveUserJson = (User) => {
131   const url = 'http://localhost:5000/write'
132   axios.post(url, User)
133     .then(response => {
134       //console.log(response);
135     });
136 }
137
138 handleImageChange = async (image = this.state.imageURL) => {
139   await this.getImageDimension(image);
140   await getFullFaceDescription(image).then(fullDesc => {
141     this.setState({ fullDesc, loading: false });
142   });
143 };
144
145 getImageDimension = imageURL => {
146   let img = new Image();
147   img.onload = () => {
148     let HEIGHT = (this.state.WIDTH * img.height) / img.width;
149     this.setState({
150       HEIGHT,
151       imageDimension: {
152         width: img.width,
153         height: img.height
154       }
155     });
156   };
157   img.src = imageURL;
158 };
159
160 handleDescriptorsCheck = event => {
161   this.setState({ showDescriptors: event.target.checked });
162 }
163 };

```

Main Logic Script

Chapter 6

Testing

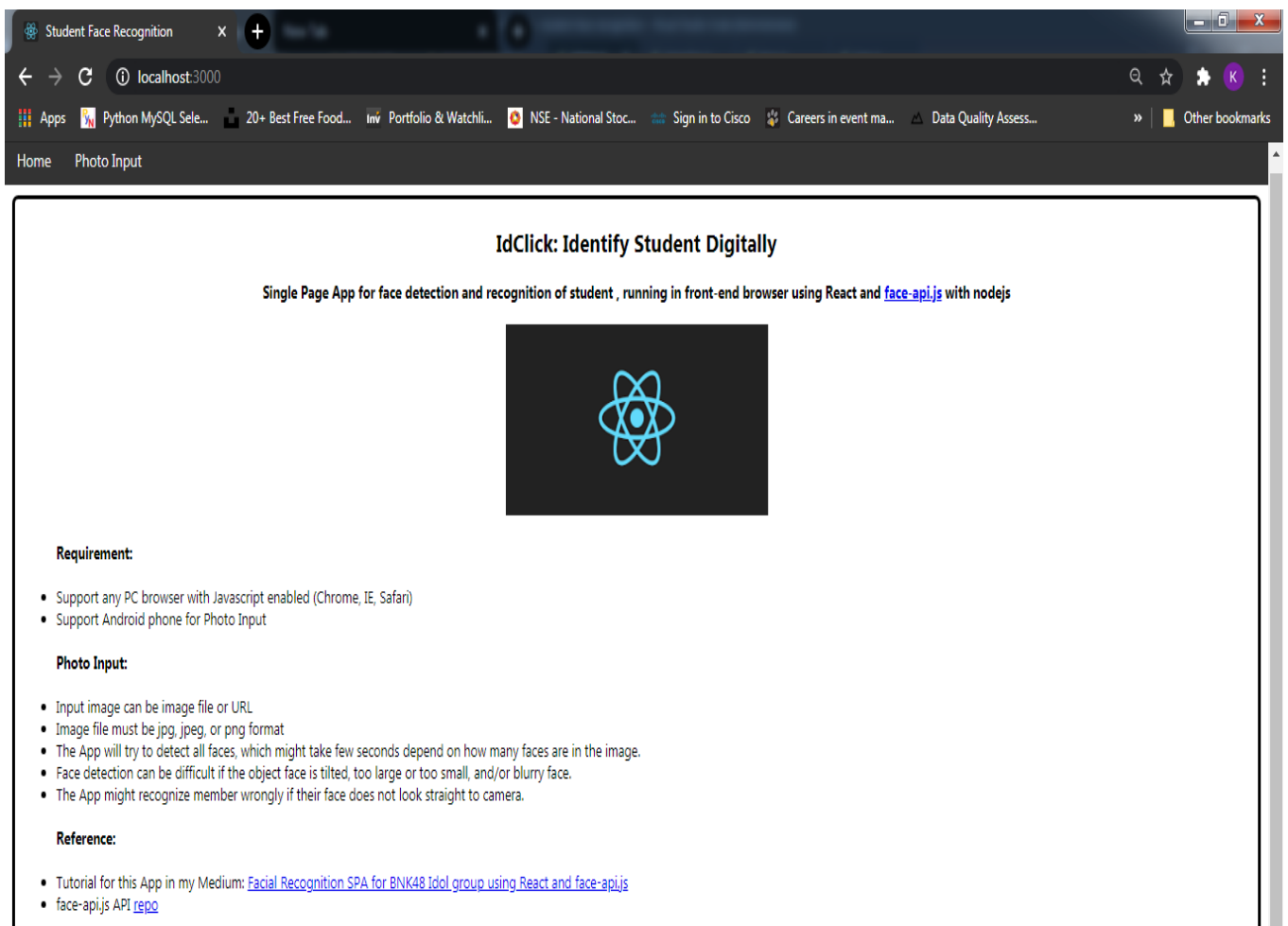
Test No.	Test Name	Expected Result	Actual Result
1	Home Screen	Photo input field, few details on home page	Successfully showing home page
2	Input a photo	Display uploaded photo on screen	Picture seen on screen successfully.
3	Face-api.js	Show calculated array in console	Successfully displaying array into the console.
4	Display box if face recognized	Blue box on face if present	Box displayed successfully
5	Recognize face based on array calculated	Show name of person in picture	Successfully displaying name of person.
6	Connect reactjs to nodejs	Able to send request from app to nodejs	Request send successfully
7	Save the calculated array into json	Saves array into json after clicking on upload	saved successfully
8	Integrate twilio with the app	Able to receive message on whats app send from app	Message received successfully.

Table 6.1: Functionality tests

Chapter 7

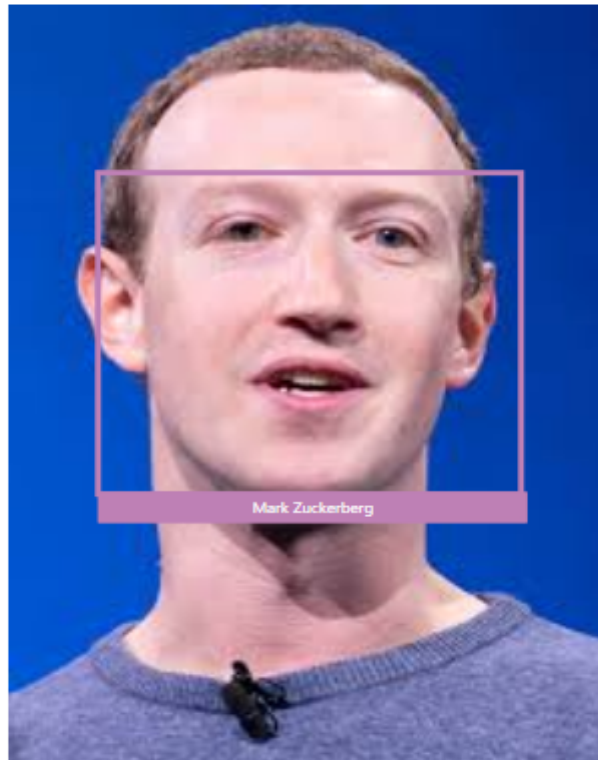
Result

The result is estimated on the objective we defined and improving face recognition using modern technologies. It can be stated that using modern technologies not only solve problems in an efficient way but also make user interaction more easy with help of modern UI/UX. Effective use of face recognition can make huge impact on daily life and can make one's work easy.



Home Page

Status: 1 Face Detect



Input Image file

No file chosen

Enter name:

☐ Show Descriptors

Photo Input Page

[This message is sent from] [Send message](#)
☒ Show Descriptors

Detail Descriptions

Descriptor_0: -0.12311732023954391,0.14340874552726746,0.021503474563360214,-0.010104609653353691,-0.06847052276134491,-0.027419663965702057,-0.059305623173713684,-0.058485787361860275,0.1493530422449112,-0.04139188304543495,0.2163170725107193,-0.09587417542934418,-0.27460673451423645,-0.11156047880649567,-0.028356464579701424,0.08512143790721893,-0.1328933984041214,-0.1328367441892624,0.00710300775244832,-0.047974154353141785,0.10911364108324051,0.024022862315177917,0.009438193403184414,0.13978448510169983,-0.20831064879894257,-0.3649642765522003,-0.07000984996557236,-0.11868169158697128,-0.026580700650811195,-0.1710844337940216,-0.08781768381595612,-0.03965432569384575,-0.21384607255458832,-0.039812903851270676,-0.02024807408452034,0.13173237442970276,0.03909853473305702,-0.016993053257465363,0.16899025440216064,-0.00015346426516771317,-0.16067847609519958,-0.029923047870397568,0.006483192555606365,0.2570006847381592,0.13129159808158875,0.05169671028852463,0.0024818554520606995,-0.004729568958282471,0.08977311104536057,-0.22434905171394348,0.0702093094587326,0.18107350170612335,0.1874159425497055,0.0011903736740350723,0.06493557244539261,-0.18588104844093323,-0.056562673300504684,0.1291418969631195,-0.16102385520935059,0.058150287717580795,-0.0014561060816049576,0.0022071008570492268,0.025376196950674057,-0.007213596720248461,0.21243959665298462,0.07579498738050461,-0.13601240515708923,-0.08302010595798492,0.09693706780672073,-0.17457497119903564,-0.00882725603878498,0.07228104770183563,-0.04791463911533356,-0.233558788895607,-0.2979535460472107,0.07977153360843658,0.37431100010871887,0.1342785805463791,-0.24797102808952332,0.015706762671470642,-0.12094986438751221,-0.0017888389993458986,0.056157417595386505,-0.03541775047779083,-0.12625284492969513,0.08382724970579147,-0.2067534476518631,0.05823372304439545,0.18518273532390594,0.007963558658957481,0.006547676865011454,0.22017724812030792,-0.030253339558839798,0.10074318945407867,0.05210038274526596,0.05702740630740527,-0.1200947666168212,0.0

128-d array of person

[Home](#) [Photo Input](#)

Status: 1 Face Detect



new photo example



Input Image file

Karan2.jpg

Enter name:

☐ Show Descriptors

Input user name



Input Image file

Karan2.jpg

Enter name:

User uploaded successfully message

Status: 1 Face Detect



Input Image file

Karan2.jpg

Lastest added user recognized

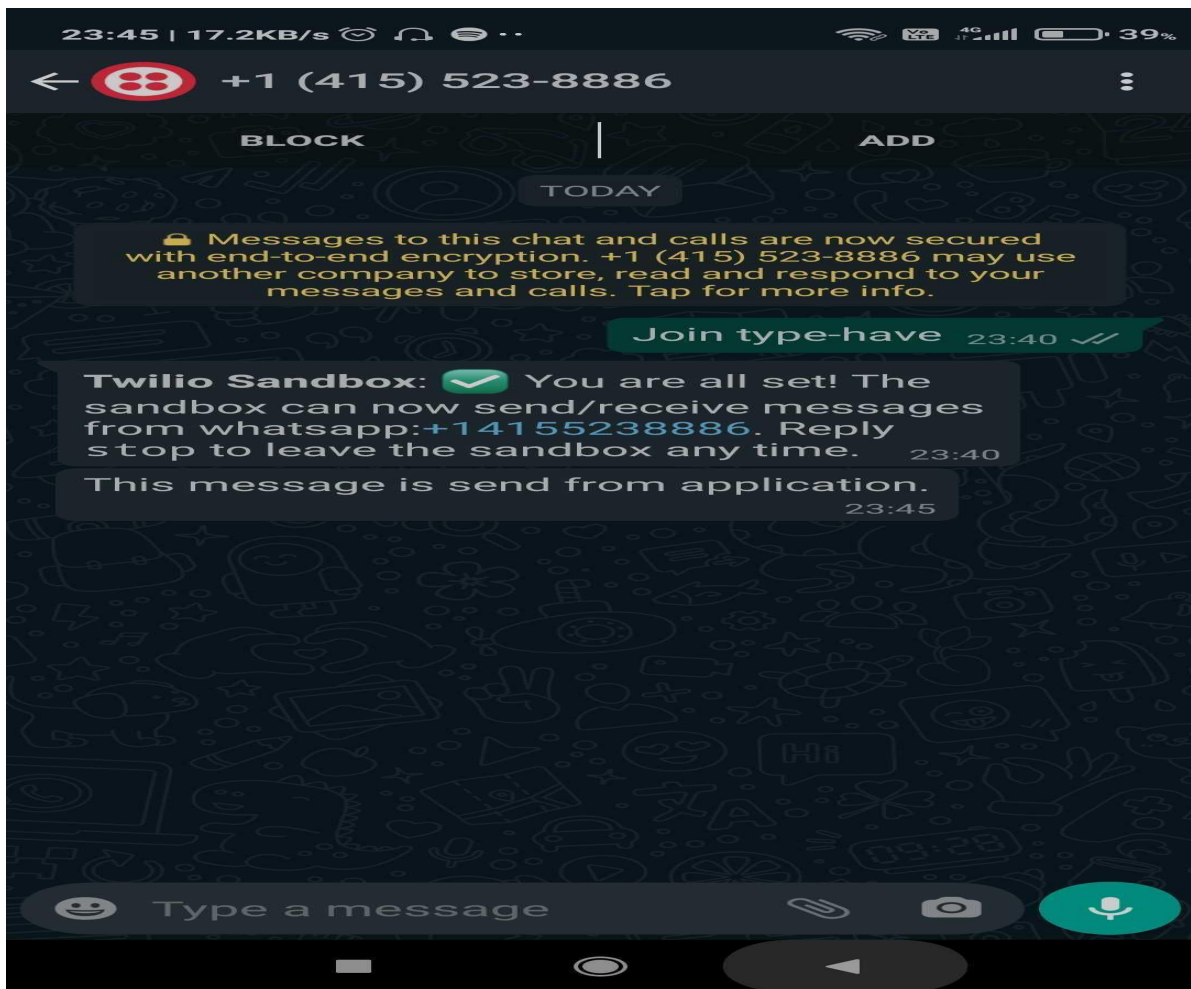
Input Image file

Karan2.jpg

Enter name:

☐ Show Descriptors

Send Message option



Message received from application




twilio


DOCS ▾Karan Thakkar ▾

My first Twilio... TRIAL ▾Messaging / Monitor /Try the beta Console


Upgrade Project

Go to...







Programmable Messaging



Monitor




Dashboard




Insights


New




Logs




Try it Out




Messaging Services



Senders



Helper Tools



Settings

DATE

SERVICE

DIRECTION

FROM

TO

SEGMENTS

STATUS

MEDIA

[2021-04-20 18:15:56 UTC](#)

—

Outgoing API

whatsapp:+14155238886

whatsapp:+918828088551

1

Read

—

[2021-04-20 18:10:52 UTC](#)

—

Incoming

whatsapp:+918828088551

whatsapp:+14155238886

1

Received

—

[2021-04-20 18:10:03 UTC](#)

—

Incoming

whatsapp:+918828088551

whatsapp:+14155238886

1

Received

—

[2021-03-07 15:05:11 UTC](#)

—

Outgoing API

whatsapp:+14155238886

whatsapp:+918828088551

1

Read

—

[2021-03-07 14:24:27 UTC](#)

—

Outgoing API

whatsapp:+14155238886

whatsapp:+918828088551

1

Read

—

[2021-03-07 14:20:47 UTC](#)

—

Outgoing API

whatsapp:+14155238886

whatsapp:+918828088551

1

Read

—

[2021-03-07 14:19:05 UTC](#)

—

Incoming

whatsapp:+918828088551

whatsapp:+14155238886

1

Received

—

[2021-03-07](#)

—

[2021-03-07](#)

—

Message Body

Join type=have

Twilio Dashboard, api log

Chapter 8

Conclusions and Future Scope

The App could detect and recognize Idol face quite accurately, but still have some error happen sometimes. This is due to the subject might not face directly to camera, their faces might tilt, or the photo was edited by some other apps. Some idols may look alike each other, which make App confuse sometime. We found that idol face can be different when come from different sources or light setting. Idols with glasses or heavy make-up can confuse our App as well.

We've tested with Chrome and Safari, and it works fine on PC. We assume that it should work with IE or Firefox as well. Testing with Android smartphone is working good for both Image Input and Video Input, but react-webcam doesn't work with iPhone due to security issue, which We're still looking for solution to workaround. Older phone tend to not work properly with TensorFlow, as it require enough computing power to run neural networks. There's always room for improvement, this app can further be improved by applying ML models for blurry images, face alignment-model, and input multiple user at the same time.

As we all know that face recognition helps to identify person seamlessly so this technology can be implemented anywhere where we want to identify person on a regular basis and make the process easy for authenticating the person. This places could be gyms, colleges, offices, Buildings, etc.

Bibliography

- 1 Authors W. ZHAO, R. CHELLAPPA, P. J. PHILLIPS, A. ROSENFELD have published a paper in 2003 ACM Computing Surveys, Vol. 35 entitled [“Face Recognition: A Literature Survey”](#).
- 2 towardsdatascience.com/facial-recognition
- 3 [Reactjs Official Documentation](#)
- 4 [Nodejs Official Documentation](#)
- 5 Youtube channel: WebStylePress, Codevolution

Appendices

Steps For Installation of Required Softwares.

Appendix-A: Install Node.js and NPM

Step 1: Download Node.js Installer :-

In a web browser, navigate to <https://nodejs.org/en/download/>. Click the **Windows Installer** button to download the latest default version.

Step 2: Install Node.js and NPM from Browser :-

1. Once the installer finishes downloading, launch it. Open the **downloads** link in your browser and click the file. Or, browse to the location where you have saved the file and double-click it to launch.
2. The system will ask if you want to run the software – click **Run**.
3. You will be welcomed to the Node.js Setup Wizard – click **Next**.
4. On the next screen, review the license agreement. Click **Next** if you agree to the terms and install the software.
5. The installer will prompt you for the installation location. Leave the default location, unless you have a specific need to install it somewhere else – then click **Next**.
6. The wizard will let you select components to include or remove from the installation. Again, unless you have a specific need, accept the defaults by clicking **Next**.
7. Finally, click the **Install** button to run the installer. When it finishes, click **Finish**.

Step 3: Verify Installation :-

Open a command prompt (or PowerShell), and enter the following:

command :- `node -v`

The system should display the Node.js version installed on your system. You can do the same for NPM:

command :- `npm -v`

Acknowledgement

We have great pleasure in presenting the report on **IdClick: Identify Student Digitally**. We take this opportunity to express my sincere thanks towards my guide **Ms. Rujata Chaudhari**, Department of IT, APSIT Thane, for providing the technical guidelines and suggestions regarding line of work. We would like to express my gratitude towards her constant encouragement, support and guidance through the development of project.

We thank **Prof. Kiran B. Deshpande**, Head of Department, IT, APSIT for his encouragement during progress meetings and providing guidelines to write this report.

We thank **Prof. Vishal S. Badgujar**, BE project co-ordinator, Department of IT, APSIT for being encouraging throughout the course and for guidance.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express my deep gratitude towards all my colleagues of APSIT for their encouragement.

Student Name 1: Karan Thakkar

Student ID 1: 17104039

Student Name 2: Arun Pandey

Student ID 2: 17104020

Student Name 3: Gunasekar Naikar

Student ID 3: 17104055