

STUDENT ATTENDANCE SYSTEM USING FACIAL RECOGNITION

*The Main Project submitted
in partial fulfillment of the requirements
for the award of the degree of*

BACHELOR OF TECHNOLOGY In COMPUTER SCIENCE AND ENGINEERING

Submitted by

- | | |
|---|---|
| 1. P. B. Siva Kumar (16PA1A05C3) | 2. S. Sajid Vali Rehman (16PA1A05E9) |
| 3. Y. Ajay (16PA1A05H3) | 4. V. Ramya (16PA1A05G7) |

**Under the esteemed guidance of
Mr. K. Bhargav Kiran
Assistant Professor**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VISHNU INSTITUTE OF TECHNOLOGY
(Autonomous)**

(Approved by AICTE, Accredited by NBA & NAAC and permanently affiliated to JNTU Kakinada)

BHIMAVARAM-534202

2019-2020

VISHNU INSTITUTE OF TECHNOLOGY

(Autonomous)

(Approved by AICTE, Accredited by NBA & NAAC and permanently affiliated to JNTU Kakinada)

BHIMAVARAM-534202

2019-2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the project entitled “**STUDENT ATTENDANCE SYSTEM USING FACIAL RECOGNITION**”, is being submitted by ***P.B.SIVA KUMAR, S.SAJID VALI REHMAN, Y.AJAY AND V.RAMYA***, bearing the **REGD.NOS: 16PA1A05C3, 16PA1A05E9, 16PA1A05H3, and 16PA1A05G7** submitted in fulfillment for the award of the degree of “**BACHELOR OF TECHNOLOGY**” in “**COMPUTER SCIENCE AND ENGINEERING**” is a record of bonafide work carried out by them under my guidance and supervision during the academic year 2019-2020 and it has been found worthy of acceptance according to the requirements of university.

Internal Guide

Mr. K. Bhargav Kiran

Head of the Department

Dr. Sumit Gupta

External Examiner

ACKNOWLEDGEMENT

It is natural and inevitable that the thoughts and ideas of other people tend to drift into the subconscious due to various human parameters, where one feels acknowledge the help and guidance derived from others. We acknowledge each of those who have contributed to the fulfillment of this project.

We take the opportunity to express our sincere gratitude to **Dr. D.Suryanarayana**, director and principal, VIT, Bhimavaram whose guidance from time to time helped us to complete this project successfully.

We are very much thankful to **Dr. Sumit Gupta**, Head of the Department, Department of Computer Science and Engineering for his continuous and unrelenting support and guidance. We thank and acknowledge our gratitude to him for his valuable guidance and support expended to us right from the conception of the idea to the completion of this project.

We are very much thankful to **Mr. K. Bhargav Kiran**, Assistant Professor, our internal guide whose guidance from time to time helped us to complete this project successfully.

Project Associates

P. B. Siva Kumar	(16PA1A05C3)
S. Sajid Vali Rehman	(16PA1A05E9)
Y. Ajay	(16PA1A05H3)
V. Ramya	(16PA1A05G7)

ABSTRACT

The education system has undergone a lot of technological advancements these days by utilizing the latest techniques and innovation in every possible way. Despite these developments, the process of taking attendance for students still follows the traditional methods. This problem is mostly seen in schools and colleges. The problem associated with this method is time taking and practices of giving false attendance can be possible. Our project address this key issue by developing a “*Student Attendance System Using Face Recognition*”. Initially, the teacher or the respective faculty needs to take a video of the entire class either in a single snapshot or by a series of videos and uploads them into a web application developed by us, which is equipped with facial recognition capabilities. On uploading, these videos are sent to a system working in the backend which can recognize the faces in the frames and gives attendance to the students or people based on the faces identified in the frames. This lets faculty save time and also reduces the probability of getting a fake attendance. This system can not only be useful for schools or colleges but can also be extended to other workplaces where taking attendance or headcount of people attended on a particular day is mandatory. From this project, we hope to build a system that is quite helpful for people by utilizing the latest technological advancements possible and we strive to add more updates to make this system highly reliable and performant.

TABLE OF CONTENTS

Sl. No	Contents	Page Numbers
1	Introduction	1
	1.1 Problem Statement	1
	1.2 Requirements	1
2	Literature Survey	2
	2.1 Convolution Neural Networks	2
	2.2 Transfer Learning	9
	2.3 Resnet	10
3	Dataset Explanation	12
	3.1 Dataset Description	12
4	Data Pre-Processing	14
	4.1 Data Analysis	14
	4.2 Data Cleaning	14
5	Design & Data Flow Diagram	15
	5.1 Design	16
	5.2 Data Flow	17
6	Database Architecture	18
	6.1 Database Explanation	18
7	Face Recognition Model & DataBase Models Utilization	20
	7.1 Face Recognition Model	20

	7.2 Django Models	21
8	Experimental Setup	25
	8.1 PostMan	25
	8.2 Localhost Environment	26
	8.3 Jupyter Notebook	26
9	Evaluating Results	27
	9.1 Adding Students	27
	9.2 Allocating Class for Faculty	28
	9.3 Listing the Attendance	29
10	Front Screens & Explanation	31
	10.1 Login System	31
	10.2 Dashboard Component	32
	10.3 Display Data	34
11	Future Work	36
	11.1 Short Term Goal	36
	11.2 Long Term Goal	36
12	Bibliography	38
	12.1 Resources and Citations Listing	38
13	Appendix	40
	13.1 Login Component Angular Code HTML	40
	13.2 Login Component Angular Code Typescript	41
	13.3 Django Models Code in models.py	42

LIST OF DIAGRAMS

Sl. No	Contents	Page Number
2.1.1	A CNN sequence to classify handwritten digits	2
2.1.2	Flattening of a 3x3 image matrix into a 9x1 vector	3
2.1.3	4x4x3 RGB Image	4
2.1.4	Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolution	4
2.1.5	Movement of the Kernel	5
2.1.6	Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel	6
2.1.7	Convolution Operation with Stride Length=2	7
2.1.8	Same padding: 5x5x1 image is padded with 0s to create a 6x6x1 image	8
2.2.1	Image representing Inception in Transfers Learning	9
2.3.1	Image representing the Neural Network inside ResNet	11
3.1.1	An image illustrating the encoding of a student image with registered no. 17PA1A0501	12

3.1.2	An image illustrating the dataset of a student images with their registered numbers.	13
5.0.1	Design Diagram	15
6.1.1	Image representing the structure of database	18
7.2.1	Image illustrating our Database Models for efficiently storing all the necessary details.	21
7.2.2	An image describing the class allocated to a faculty in CSE 2nd Year 2nd Semester with Section B on 4th Period of every Wednesday.	22
7.2.3	Image illustrating how each student is stored	23
7.2.4	Image illustrating how each user is stored.	24
8.1.1	An image to illustrate the interface and working of postman	25
9.1.1	Image illustrating the request sent to API	27
9.1.2	16PA1A05D0 in the database and his unique Encoding	28
9.2.1	Request for allocating classes to Faculty	29
9.3.1	Image indicating the list of students identified in the video	30
10.1.1	Common Login Screen for Teacher and Student	32
10.2.1	Front screen of attendance upload with ID of a faculty	34
10.3.1	Image of identified students as output in frontend application	35

INTRODUCTION

1. Introduction

1.1 Problem Statement

The education system has undergone a lot of technological advancements these days by utilizing the latest techniques and innovation in every possible way. Despite these developments, the process of taking attendance for students still follows the traditional methods. This problem is mostly seen in schools and colleges. The problem associated with this method is time taking and practices of giving false attendance can be possible. Our idea is to address this key issue by proposing an “Student Attendance System Using Face Recognition” which can recognize the faces of students upon taking a video of the entire class and uploading it through an application interface so that our facial recognition model can evaluate the people found in that video and gives them attendance in an automated manner.

1.2 Requirements

1.2.1 Hardware Requirements

- Android Mobile
- Minimum of 3GB RAM
- A System with a minimum of 4GB RAM

1.2.2 Software Requirements

- Student Image Dataset
- TensorFlow
- Angular
- Python
- Django
- Django Rest Framework
- Postman API Tool
- Jupyter Notebook
- Face Recognition Model
- Visual Studio for Coding Purpose
- Dlib Package as a Dependency

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 Convolutional Neural Network

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

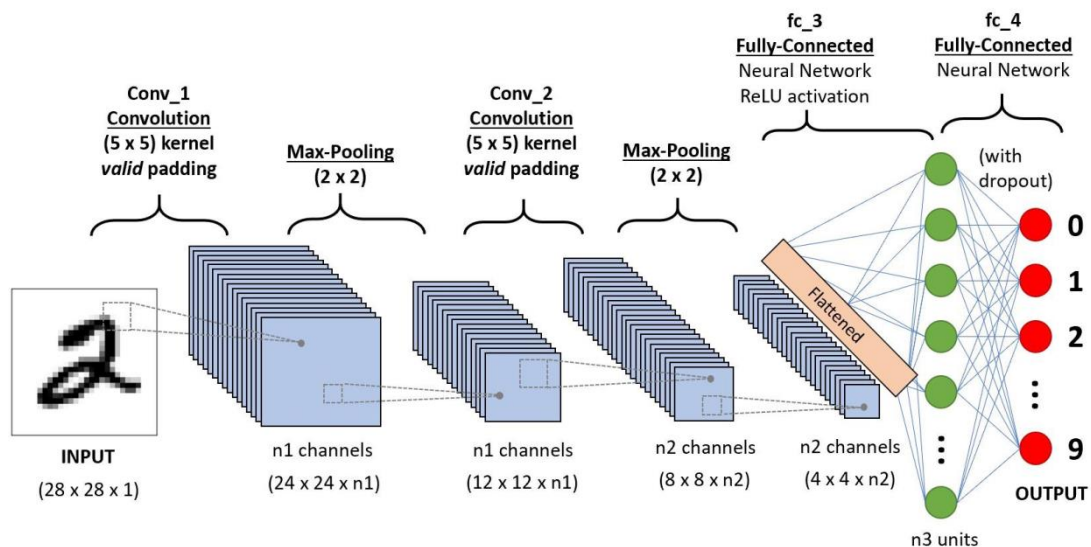


Figure 2.1.1: A CNN sequence to classify handwritten digits

In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

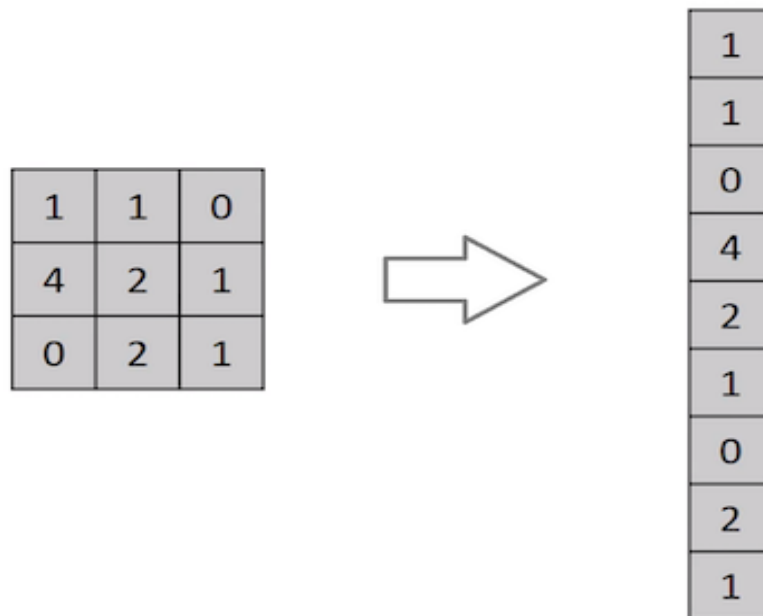


Figure 2.1.2: Flattening of a 3x3 image matrix into a 9x1 vector

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to over fitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns.

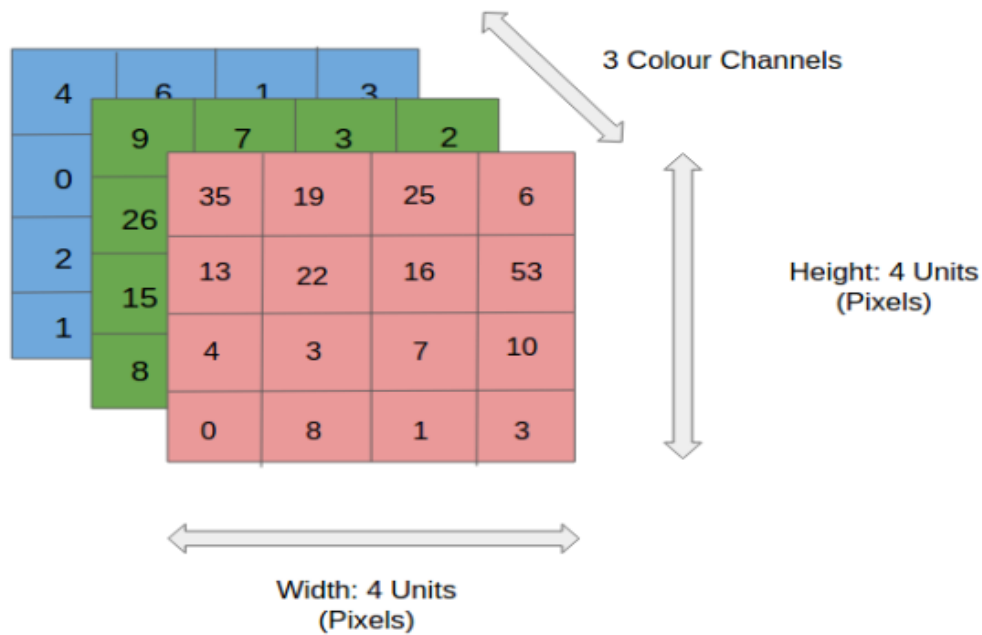


Figure 2.1.3: 4x4x3 RGB Image

In the figure, we have an RGB image which has been separated by its three color planes Red, Green, and Blue. There are a number of such color spaces in which images exist rayscale, RGB, HSV, CMYK, etc.

You can imagine how computationally intensive things would get once the images reach dimensions, say 8K (7680×4320). This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

2.1.1 Convolution Layer — The Kernel

Image Dimensions = 5 (Height) x 5 (Breadth) x 1 (Number of channels, eg. RGB)

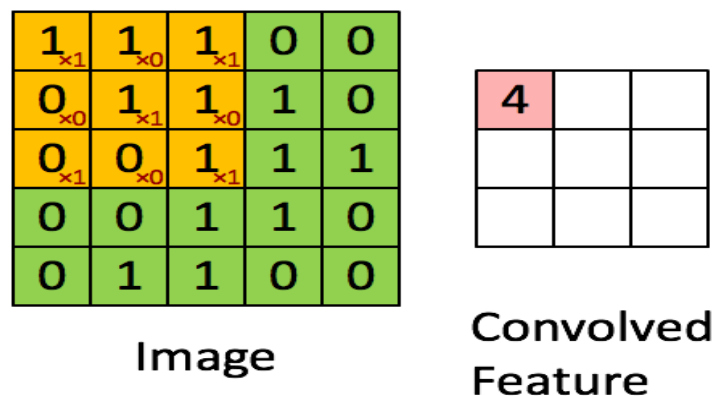


Figure 2.1.4: Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolution

In the above demonstration, the green section resembles our $5 \times 5 \times 1$ input image, I. The element involved in carrying out the convolution operation in the first part of a Convolutional layer is called the Kernel/Filter, K, represented in the color yellow. We have selected K as a $3 \times 3 \times 1$ matrix.

The Kernel shifts 9 times because of Stride Length = 1 (Non-Strided), every time performing a matrix multiplication operation between K and the portion P of the image over which the kernel is hovering.

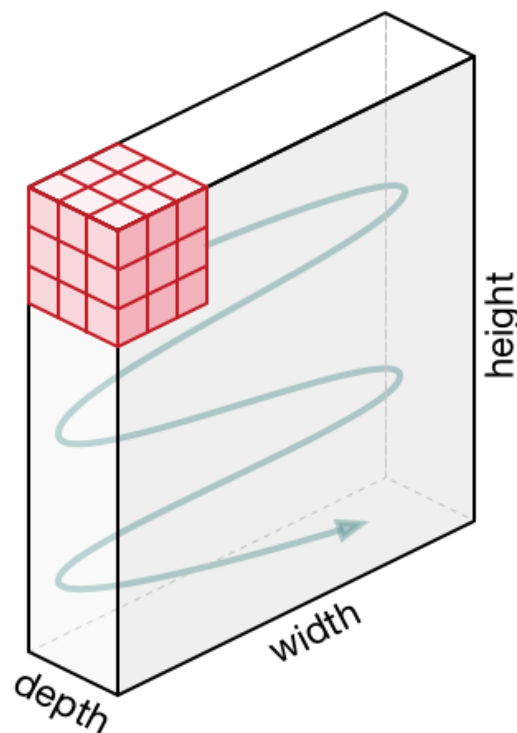


Figure 2.1.5: Movement of the Kernel

The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

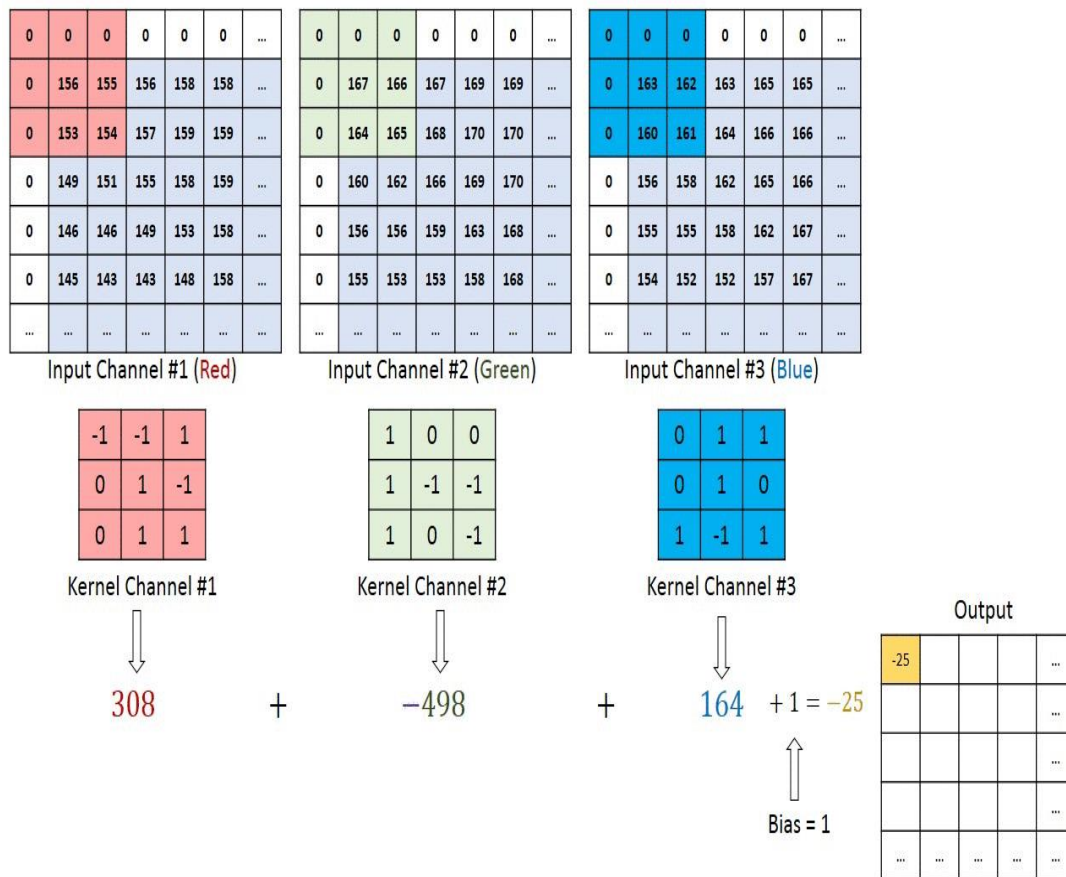
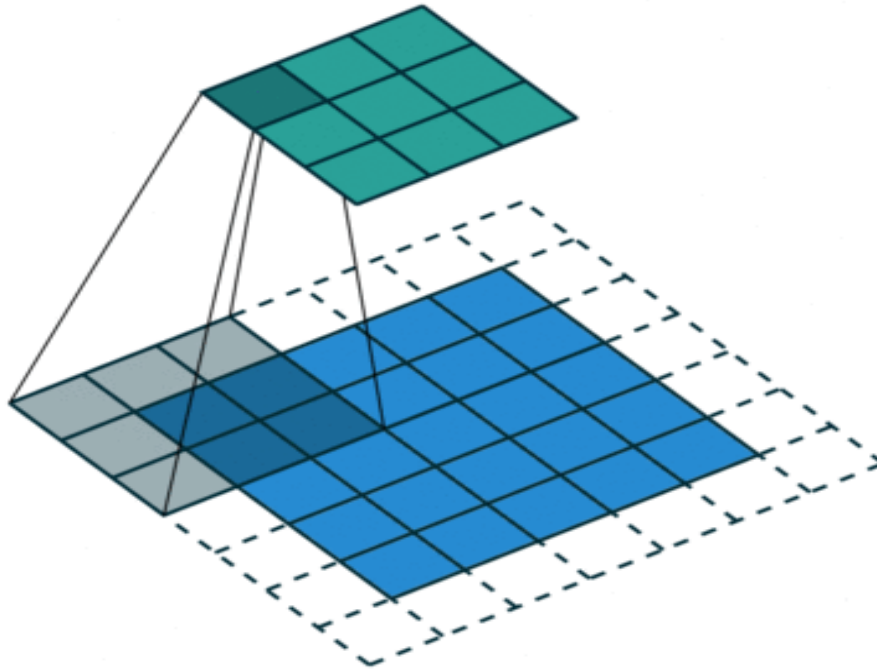


Figure 2.1.6: Convolution operation on a $M \times N \times 3$ image matrix with a $3 \times 3 \times 3$ Kernel

In the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image. Matrix Multiplication is performed between K_n and I_n stack ($[K_1, I_1]$; $[K_2, I_2]$; $[K_3, I_3]$) and all the results are summed with the bias to give us a squashed one-depth channel Convolved Feature Output.

There's another use for convolution matrix, which is actually part of the reason why they are called “filters”. The word here is used in the same sense we use it when talking about Instagram filters. You can actually use a convolution matrix to adjust an image. So it can be used for blurring, embossing, edge detection, sharpening and much more.

In case of multiple layers The filters that operate on the output of the first line layers may extract features that are combinations of lower-level features, such as features that comprise multiple lines to express shapes. This process continues until very deep layers are extracting faces, animals, houses, and so on.



Figur 2.1.7. Convolution Operation with Stride Length = 2

The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network which has the wholesome understanding of images in the dataset, similar to how we would.

There are two types of results to the operation — one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying Valid Padding in case of the former, or Same Padding in the case of the latter.

However, we can increase the size of kernel if you need to focus on larger parts of the image instead the small ones. The distance by which the window moves each time is called the stride. Here, the window moves by 1 position at a time (i.e. it shifts from left to right by 1 pixel at a time), hence stride=1. The size of output image depends on the values of size of image and kernel; and the values of stride and padding.

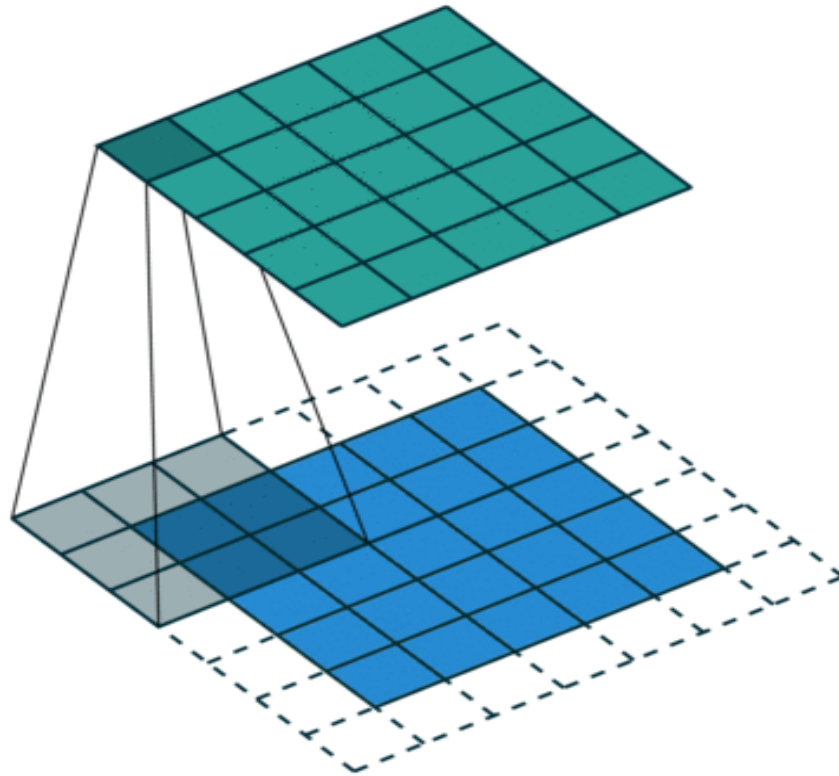


Figure 2.1.8: Same padding: 5x5x1 image is padded with 0s to create a 6x6x1 image

When we augment the 5x5x1 image into a 6x6x1 image and then apply the 3x3x1 kernel over it, we find that the convolved matrix turns out to be of dimensions 5x5x1. Hence the name Same Padding. On the other hand, if we perform the same operation without padding, we are presented with a matrix which has dimensions of the Kernel (3x3x1) itself Valid Padding.

Transfer learning allows you to retrain the final layer of an existing model, resulting in a significant decrease in not only training time, but also the size of the dataset required. One of the most famous models that can be used for transfer learning is Inception V3. As mentioned above, this model was originally trained on over a million images from 1,000 classes on some very powerful machines.

Being able to retrain the final layer means that you can maintain the knowledge that the model had learned during its original training and apply it to your smaller dataset, resulting in highly accurate classifications without the need for extensive training and computational power.

2.2 Transfer Learning

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task. Human learners appear to have inherent ways to transfer knowledge between tasks. That is, we recognize and apply relevant knowledge from previous learning experience when we encounter new tasks. The more related a new task is to our previous experience, the more easily we can master it.

Transfer learning involves the approach in which knowledge learned in one or more source tasks is transferred and used to improve the learning of a related target task. While most machine learning algorithms are designed to address single tasks, the development of algorithms that facilitate transfer learning is a topic of ongoing interest in the machine-learning community.

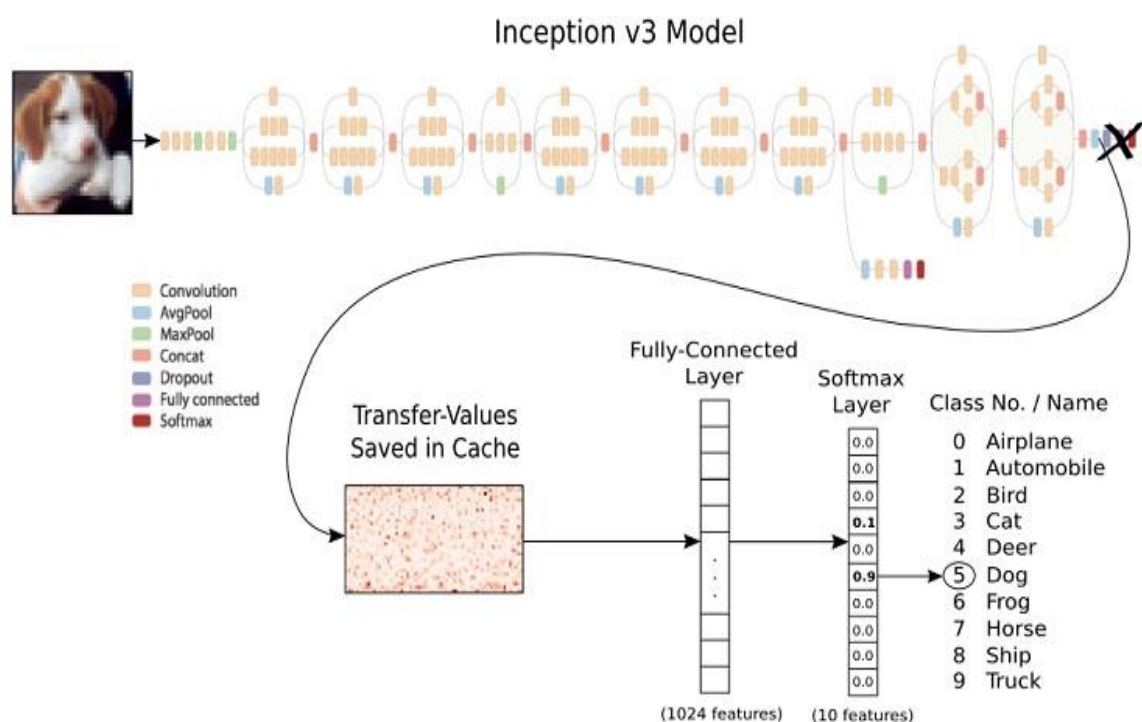


Figure 2.2.1: Image representing inception in transfers learning

Transfer learning allows you to retrain the final layer of an existing model, resulting in a significant decrease in not only training time, but also the size of the dataset required. One of the most famous models that can be used for transfer learning is Inception V3. As mentioned

above, this model was originally trained on over a million images from 1,000 classes on some very powerful machines. Being able to retrain the final layer means that you can maintain the knowledge that the model had learned during its original training and apply it to your smaller dataset, resulting in highly accurate classifications without the need for extensive training and computational power.

2.3 ResNet

ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. This model was the winner of ImageNet challenge in 2015. The fundamental break through with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully. Prior to ResNet training very deep neural networks was difficult due to the problem of vanishing gradients.

In general, in a deep convolutional neural network, several layers are stacked and are trained to the task at hand. The network learns several low/mid/high level features at the end of its layers. In residual learning, instead of trying to learn some features, we try to learn some residual. Residual can be simply understood as subtraction of feature learned from input of that layer. ResNet does this using shortcut connections (directly connecting input of nth layer to some $(n+x)$ th layer.

It has proved that training this form of networks is easier than training simple deep convolutional neural networks and also the problem of degrading accuracy is resolved. This is the fundamental concept of ResNet. ResNet50 is a 50 layer Residual Network. There are other variants like ResNet101 and ResNet152 also.

ResNet-50 is a convolutional neural network that is trained on more than a million images from the ImageNet database. The network is 50 layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.

However, increasing network depth does not work by simply stacking layers together. Deep networks are hard to train because of the notorious vanishing gradient problem — as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient extremely small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly.

The ResNet-50 model consists of 5 stages each with a convolution and Identity block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers. The ResNet-50 has over 23 million trainable parameters.

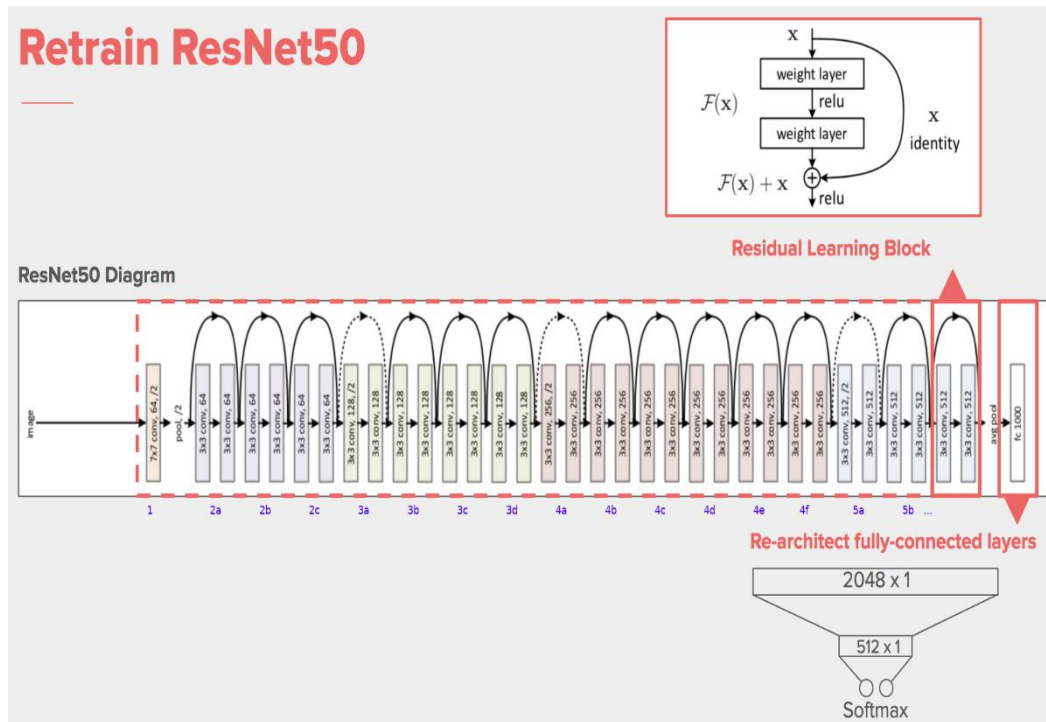


Figure 2.3.1: Image representing the Neural Network inside ResNet Model

DATASET EXPLANATION

3. Dataset Explanation

3.1 Dataset Description

For this project, the dataset we need is the images of students in our college with respect to their year of study. We gathered our data from various sources like College Administrators, College website and some live images of students. Then we categorized the images based on their year of study and the registration number of students is taken as the file name for that particular student.

All the images that we have taken are of different formats like JPEG, PNG and possessing different resolutions with file sizes ranging from 1MB - 8MB. Apart from this, the images that we have used are of the same dimension. Some images are redundant and others in landscape mode. We created a custom dataset that contains unique encodings of each person mapped with their registration number and stored these encodings in a JSON file.

```
"17PA1A0501.JPG" : [ -0.17033266, 0.03377763, 0.04389345, -0.0191634, -0.0970946, -0.03743669,
-0.01167169, -0.183294, 0.10846838, -0.07526951, 0.18952605, -0.02477362,
-0.15392999, -0.08188977, -0.01554761, 0.08585906, -0.12595017, -0.15294687,
-0.02087514, -0.05726422, 0.03869738, -0.01908768, 0.05470384, 0.1009514,
-0.16071466, -0.33012244, -0.07218581, -0.17036639, -0.01086591, 0.02297334,
-0.0443004, 0.05252742, -0.22908254, -0.0211346, 0.02150698, 0.03007019,
0.03049274, -0.03754643, 0.1578431, -0.01883491, -0.26013038, -0.10222366,
0.10321003, 0.2155305, 0.13528927, 0.04583996, -0.02767656, -0.09124197,
0.07718439, -0.18695846, 0.10373522, 0.16702053, 0.02293461, -0.04273614,
0.04282254, -0.16534293, 0.01497499, 0.01282747, -0.17351563, 0.03188762,
0.02521833, -0.02664701, -0.05889768, -0.10862891, 0.25101924, 0.18781576,
-0.07064091, -0.1013041, 0.21103449, -0.13888793, -0.02000797, 0.06550181,
-0.15454136, -0.22336255, -0.21905382, -0.0234061, 0.47396412, 0.15307298,
-0.17503856, 0.04618587, -0.19939417, -0.03724056, 0.11838096, 0.10093502,
0.00505611, 0.03524293, -0.06510487, 0.00598165, 0.2113688, -0.03106946,
-0.02179696, 0.22251265, -0.02595844, -0.04985695, 0.04235845, 0.01230709,
-0.12533522, 0.01630762, -0.17728224, -0.10424995, 0.01138198, -0.02450003,
0.02716848, 0.13256538, -0.25623778, 0.06202086, 0.07723416, -0.05274283,
0.06316276, 0.01108676, -0.07391231, -0.13380925, 0.09679175, -0.19479018,
0.19608338, 0.09897007, 0.06455779, 0.19421047, 0.04674074, 0.09151028,
-0.00167877, -0.07385932, -0.15604396, -0.01696332, 0.09328782, -0.05878304,
0.03604128, 0.02456564 ]
```

Figure 3.1.1 An image illustrating the encoding of a student image with registered no. 17PA1A0501.



Figure 3.1.2 An image illustrating the dataset of a student images with their registered numbers.

DATA PREPROCESSING

4. Data Preprocessing

4.1 Data Analysis

To feed data into our model and for generating results, we need image data of students which are quite clear with each frame in perfect visualization so that we don't miss even a single feature. Then we analyzed all the images we have gathered carefully and found that many images are of different resolutions with unclear backgrounds and different file formats. There are also images that are highly distorted and facial orientation is not straight. This analysis phase played a key role for us as images are the primary source for making predictions on student images.

4.2 Data Cleaning

Since we have collected data from various sources, there are images of students with different resolutions, distorted images and some are in portrait mode, others in landscape mode. We have got images in different file formats like PNG, JPEG, etc. So we thought of unifying data by following a specific criterion of converting images of higher file size (say above 4 MB) to a range of 1MB - 4MB, eliminated the images which are distorted and also in landscape mode so that our face recognition algorithm can work fine on these images. We also pruned images of different formats like PNG so that all images fall under the same format.

We are unable to gather some images of students from college administrators and therefore we filled those missing student's images by recapturing their images after taking their consent. While generating encodings for students in form of key value pair, the encoding key is obtained as rollnumber.JPG, so a python script is written inorder to format the dictionary keys and another python script is used inorder to format values for every roll number which are obtained as space separated values initially.

DESIGN AND DATA FLOW DIAGRAM

5. Design And Data Flow Diagram

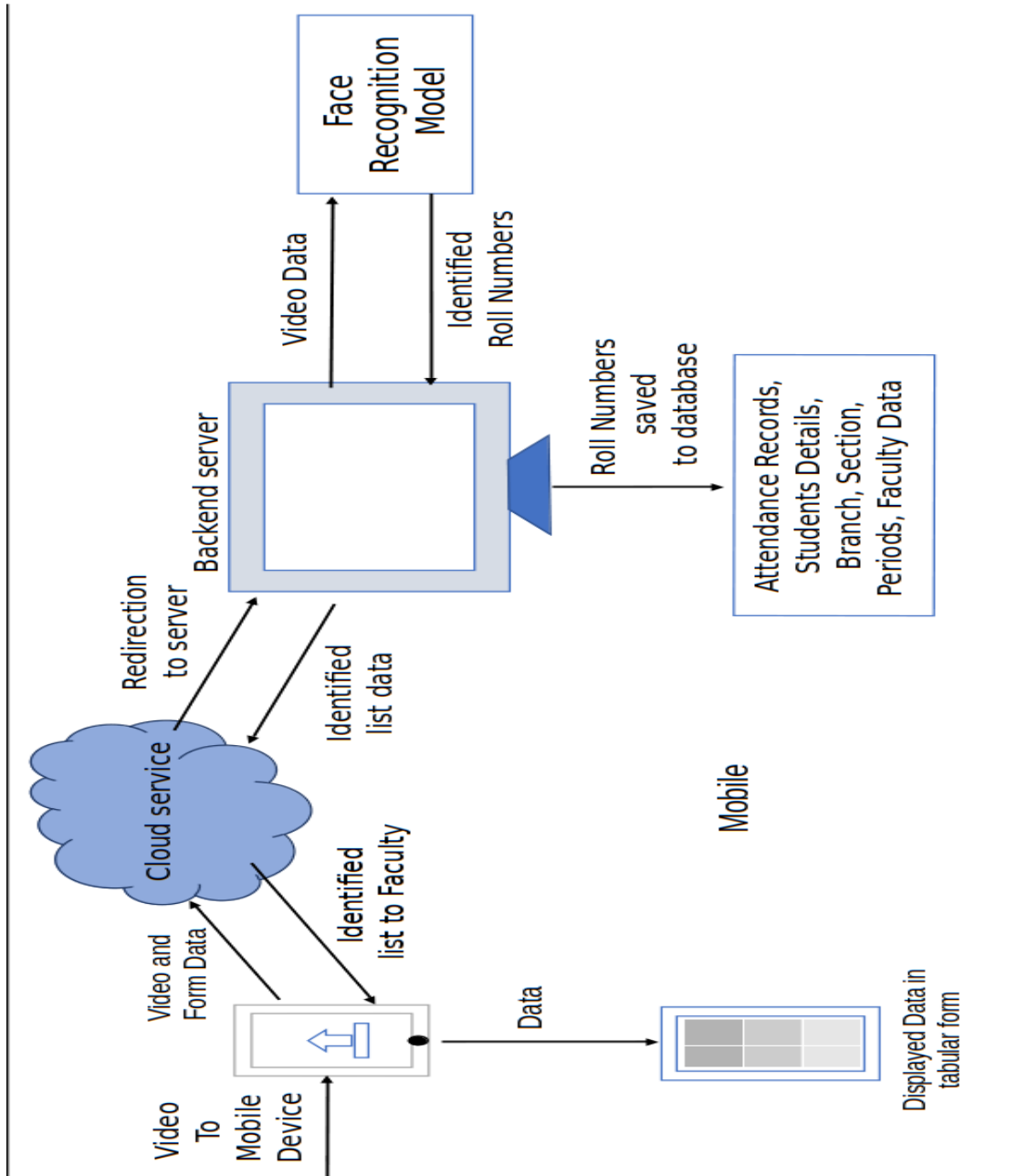


Figure 5.0.1 Design Diagram

5.1 Design

The design of this application consists of a frontend and backend where both are developed using two different application frameworks. The use of two different frameworks enabled us to deal with development easily and also this led to the creation of API's since both frameworks are different and to connect them and share data among these we need to make use of Application Programming Interface (API).

The frontend interface is what the faculty will see and upload attendance or perform any other functionalities provided with respect to application. After uploading video or performing any task, the data will be sent to backend if it is necessary and performs required computations and eventually generating responses based on the request received by server.

Here the application will be hosted on a server computer by making use of a cloud service so that the application can be accessed from anywhere rather than a single point of use and the faculty is supposed to access it through a browser portal. Upon entering the application URL in the browser, the user is displayed with a login page to sign in and then it gets redirected to a dashboard page with functionality to take attendance. Then the user will be prompted to enter some details representing a particular class and then finally capturing a video which eventually ends with a upload button. Then the backend will receive the video, details and process the video by redirecting it to the face recognition model which splits the video into frames and processes each frame, thus identifying students and storing the list of students appeared in the video in database along with timestamp and class details.

This list of students will be sent back to frontend application where the teacher can see the identified students and confirm the attendance if it's accurate or the teacher can manually add students if anyone is not identified.

5.2 Data Flow

The data flow begins with the teacher signing in where he/she is requested to enter their Faculty ID and Password which will be sent to an API running on server side as it captures these details and verifies it against the already existing records or faculty accounts. Upon successful verification and if the user is found, then a unique 256 bit token will be generated which will be sent to the frontend application where that token and Faculty ID will be stored in the browser's local storage. On sending any request to the backend application, this token will be retrieved from the browser's local storage and sends the token along with the request to the server.

Then the server will verify the token and if it's legitimate, then it takes the request forward or else the request will not be processed.

The next area of data flow is seen when the faculty uploads the attendance data to the server. Initially the faculty is prompted to select values from a few dropdown menus and then to upload video. After submitting, the video along with some details namely, branch, semester, studying year, section and period, faculty id will be sent to an api on server and then video data will be redirected to face recognition model for processing and rest of the details are stored in database.

When the face recognition model finishes its process, it sends the data of students identified to the server function call and then it is stored in the database. These list of students will also be sent back to faculty for any cross checking. The respective API will be called and data is sent by front end for every request made by faculty with respect to functionality provided in application. There also exists another website which is only for administrators to visualize all the data and since the admin site is developed using backend framework itself, there is only normal data flow rather than API involvement.

DATABASE ARCHITECTURE

6. Database Architecture

6.1 Database Explanation

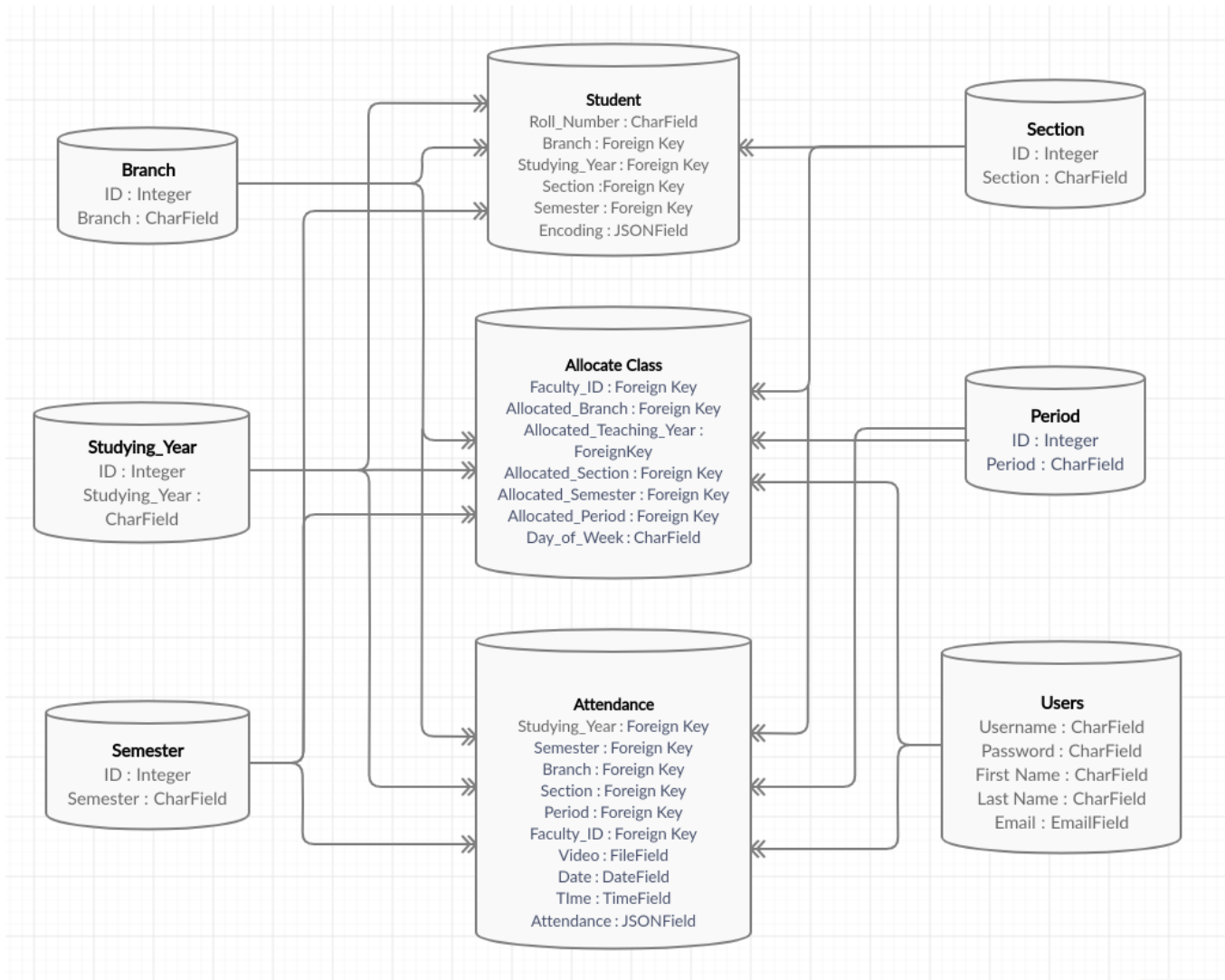


Figure 6.1.1 Image representing the structure of database

The Database used for the purpose of this application development is SQLite3 which is a built in database that comes with django framework. Inorder to manage the complexity of maintaining student details along with faculty details and classes allocated to them, we divided every large table into small tables and maintained references among tables so that redundancy can be eliminated and also it is efficient to manage. To dig deeper into this and illustrate, there are nine data tables each dedicated for Branch, Studying Year of students, Semester, Section, Period, Users.

Attendance, Student and Allocated_classes. So there are six tables like Branch, Studying Year, Semester, Section, Period, Faculty which have only data fields without any references to other tables. And then there are tables like Student, Allocate_class and Attendance which have references from other tables. For example in the case of Students, the data table stores Roll_Number of student, his/her unique encoding and the other data fields like Branch, Studying_Year, Section, Semester are stored as references from other dedicated small tables.

In the scenario of they are having the Allocate_class, Allocated_Branch, Allocated_Year, Allocated_Section, Allocated_Semester, Allocated_Period, Faculty_ID are taken as references and Days of Week is only field without reference. Finally, the Attendance contains Studying_Year, Semester, Branch, Section, Period, Faculty_ID as references from other tables and Video, Date, Time, Attendance list are the only reference less attributes.

Apart from this, there is an AUTH table which stores only the authentication tokens of users in order to verify for every request made by the user. The validity for every authentication token is set to 24 hours and after that time, faculty needs to re login again. There is also another data table for storing group permissions set to faculty in order to perform any group level operations or activities. Every record of faculty, branch, study year, semester, section, period is assigned with a unique id in their respective table which serves as a primary key and eliminates any form of redundancy.

FACE RECOGNITION MODEL & DATABASE MODELS UTILIZED

7. Face Recognition and Database Models Used

7.1 Face Recognition Model

This FaceRecognition Model is built using dlib's state-of-the-art face recognition which is developed with deep learning. The model has an accuracy of 99.38% on the labeled Faces. Dlib is a general purpose cross platform software library written in the programming language C++. Its design is heavily influenced by ideas from design by contract and component based software engineering. Thus it is, first and foremost, a set of independent software components. In our application, when the video sent by the user is derived in the backend and ready for processing, then we will be sending it to another where the program or script exists for dividing video into frames at our requested frame rate and the image that are now obtained will be used by face recognition model for processing. Initially after considering each photo, the model should be able to identify the faces in the picture and generate facial encodings for the same picture. This is done by,

```
face_landmarks_list = face_recognition.face_landmarks(image)
```

The above two lines of code illustrates how image is being loaded into script and then how we can get facial features from the image. Then after that, we need to generate encodings for all the people that are identified in the image. In this way, all the frames which are generated should have to be run in order to get all the present people's encodings. The code below illustrates an example of a person in generation of encodings.

```
biden_encoding = face_recognition.face_encodings(known_image)[0]
```

Now we need to compare the encodings which are generated from all those images with the images of people in our database. Now we start iterating over the faces with each and every encoding and then find if there is any similarity that exists between people using the comparison functionality that work on comma separated list of numpy array values. The code below illustrates comparison.

```
results = face_recognition.compare_faces([biden_encoding], unknown_encoding)
```

7.2 Django Models













Django administration		
Site administration		
APPFR1		
Allocate_classss	+ Add	Change
Attendances	+ Add	Change
Branches	+ Add	Change
Periods	+ Add	Change
Sections	+ Add	Change
Semesters	+ Add	Change
Students	+ Add	Change
Studying years	+ Add	Change
AUTH TOKEN		
Tokens	+ Add	Change
AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

Figure 7.2.1 Image illustrating our Database Models for efficiently storing all the necessary details.

The above picture provides an insight into the number of database models(Tables) used for cleanly storing data without any complexity. All the models are grouped into three categories where APPFR1 stores the models.

Home › Appfr1 › Allocate_classs › Bhargav_Kiran

Change allocate_class

Faculty ID:	Bhargav_Kiran ▼	 
Allocated Branch:	CSE ▼	 
Allocate Studying Year:	II ▼	 
Allocated Section:	B ▼	 
Allocated Semester:	II ▼	 
Allocated Period:	VI ▼	 
Day of Week:	Wednesday ▼	






Figure 7.2.2 An image describing the class allocated to a faculty in CSE 2nd Year 2nd Semester with Section B on 4th Period of every Wednesday.



- **Allocate_class:** Which keeps track of all the classes allocated to a particular faculty by Branch, Semester, Period etc.
- **Attendance:** Model is responsible for storing the records of student attendance categorized by date, time, branch, section, period, faculty.
- **Branch:** Is responsible for storing all the branches available in a particular institution.
- **Period , Section , Semester:** are the models used for storing period data including start time and end time.
- **Studying Year:** refers to the academic years available in institution.



Home › Appfr1 › Students › 16PA1A05E9



Change student

Roll Number:

Branch: CSE  

Studying Year: IV  

Section: C  

Semester: II  

Encoding:

```
{
  "16PA1A05E9":[
    -0.2015482,
    0.01025733,
    -0.00896542,
    -0.04674792,
    -0.02378061,
    -0.10259741,
    0.03629379,
    -0.09330913,
    0.06175518,
    0.01175518
  ]
}
```

Enter valid JSON

Delete

Figure 7.2.3 Image describing how each student is stored.

- **Students:** Model is used for storing necessary student details like Roll Number, Section, Branch, Semester, Studying Year.

The next category is the AUTH TOKEN,

- **Token** stores all the 256 bit hash values which are unique for users signed in, so that it provides authentication by verifying that token when a user generates requests from the front end.

The Final Category is AUTHENTICATION AND AUTHORIZATION, where there are two models termed as,

Change user

Username:	<input type="text" value="Bhargav_Kiran"/>
<small>Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.</small>	
Password:	algorithm: pbkdf2_sha256 iterations: 180000 salt: YotRSm***** hash: NGLT1A*****
<small>Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.</small>	

Personal info

First name:	<input type="text" value="K Bhargav"/>
Last name:	<input type="text" value="Kiran"/>
Email address:	<input type="text"/>

Permissions

☒ Active
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

☐ Staff status
Designates whether the user can log into this admin site.

☐ Superuser status
Designates that this user has all permissions without explicitly assigning them.

Figure 7.2.4 Image illustrating how each user is stored.

- **Users:** Store all the faculty details such as username, encrypted password, first name, last name, email and permissions set for each teacher. This model also stores the administrator data.
- **Groups:** Are used for grouping faculty or users into groups so that necessary permissions can be set to a group of faculty and other permissions which are only need to be set to a group of people.

EXPERIMENTAL SETUP

8. Experimental Setup

8.1 Postman

Postman is an API Development tool which comes handy while developing applications where frontend and backend are two different components led by two different frameworks or more than one technology. In this project, postman is used while developing the backend part of the application since it involved development of API's. As it is not possible to simultaneously work on both frontend and backend at the same time, postman came to our rescue. Starting with the development of backend using technology which is Django framework, API's were created and they were tested by using postman tool before integrating with the front end part.

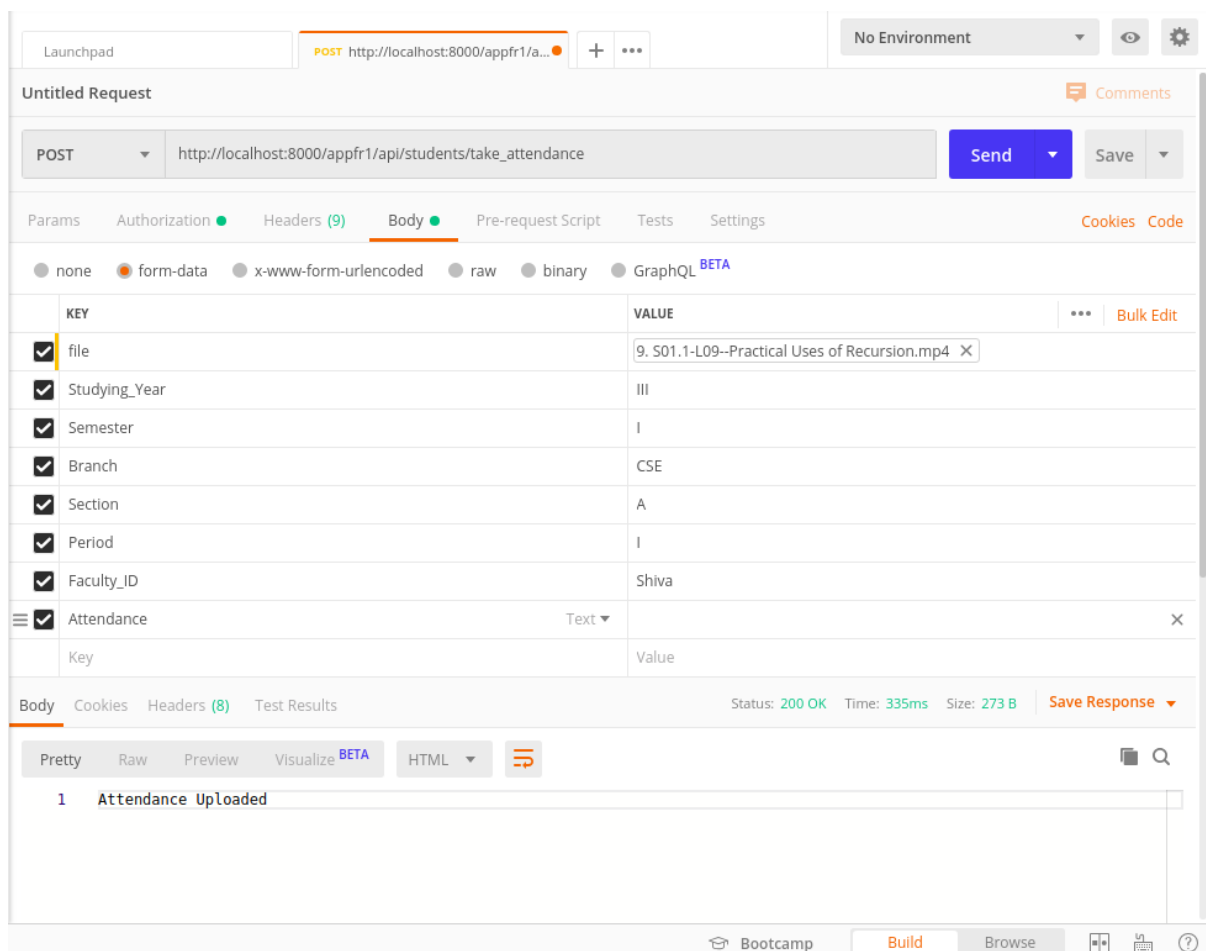


Figure 8.1.1 An image to illustrate the interface and working of postman

8.2 Localhost environment

Localhost is a server environment which is inbuilt designed into a browser system to experimentally test the application and its interface before it can provide functionality through a cloud service or a remote server environment. Experimentally, to develop applications and test the features and functionality of it, we used a localhost environment. Since we have two different application components, we are running localhost server on two different ports where Django application (Backend) runs on localhost:8000 and Angular application (Front end) runs on localhost:4200. The 8000 and 4200 are the port numbers which act as channels for communication to a single server. The localhost domain name is resolved to 127.0.0.1 when a request is sent by user.

8.3 Jupyter Notebook

To develop the facial recognition part of the application, we used Jupyter Notebook which is for testing our model on sample data like a live video taken from a class, splitting the video into frames and it gave the results we expected.

The jupyter notebook environment is also used for writing code to format student encodings like converting each student encoding into a key-value pair to store in separate dictionary, keeping all these encodings into a separate list dedicated for each branch and year and dividing them section wise so that the data can be fed easily into the student database.

EVALUATION RESULTS

9. Evaluating Results

9.1 Adding Students

The below image illustrates the working of a backend API through postman and how the data is being stored under the Students table of database successfully and returns the result back to the postman tool.

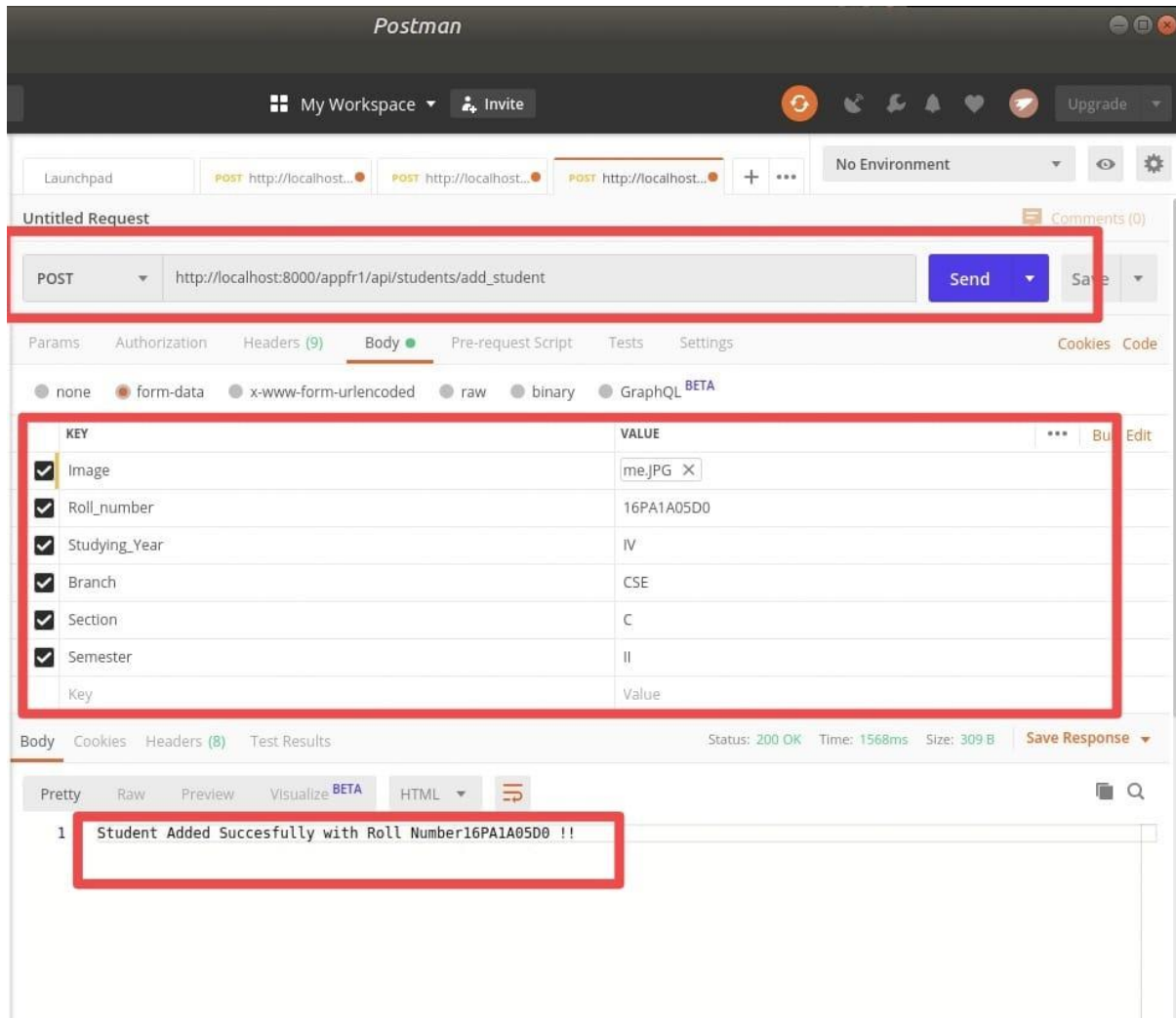


Figure 9.1.1 Image illustrating the request sent to API

Here in the above, the top red box indicates the URL to which the POST request is sent and the next red box indicates the parameters that we are sending along with request. This form data is sent in the form of JSON. The last box displays the output that a student with that particular roll number is added.

Home > Appfr1 > Students > 16PA1A05D0

Change student



Roll Number:

16PA1A05D0

Branch:

CSE



▼



Studying Year:

IV



▼



Section:

C



▼



Semester:

II

▼



Encoding:

{
"16PA1A05D0": [
-0.1879992038011551,
0.04612002894282341,
0.02659745328128338,
-0.05236280336976051,
0.022424528375267982,
-0.06797502189874649,
-0.008463971316814423,
-0.120054692029953,
0.10000000000000000
]}

Enter valid JSON

Delete

Figure 9.1.2 16PA1A05D0 in the database and his unique Encoding

9.2 Allocating class for a Faculty

Allocating class for a faculty works by sending parameters such as branch, section, teaching year etc along with the post request to API URL. The image below shows a request is being sent to backend and reverted back with a confirmation response.

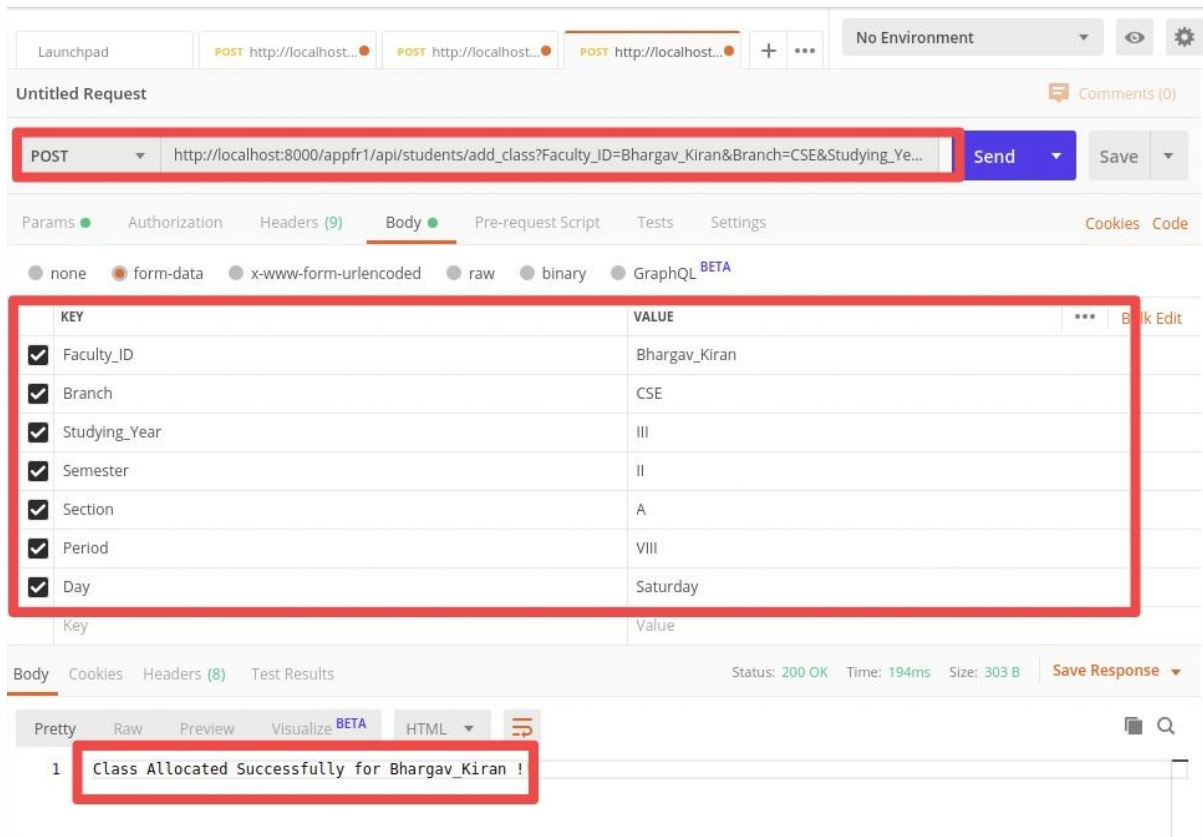


Figure 9.2.1 Request for allocating classes to Faculty

The top box indicates URL to which a POST request is sent and the next box indicates the parameters provided like ID, branch, year of teaching, semester, section, final box visualizes the output with that particular faculty name as an acknowledgement.

9.3 Listing the Attendance

On uploading the data and video to backend server through API, the application will process the request and then sends the video to face recognition model which identifies faces in frames of video and compares them with the encodings of students based on their branch, section and studying year. The below image provides an illustration of how the data is sent through postman to backend. The top red colored box indicates the URL to which data is sent and the second box is list of parameters that we are sending along with video and the third box is the list of students identified in the video which are returned in the form of dictionary as roll number and boolean value.

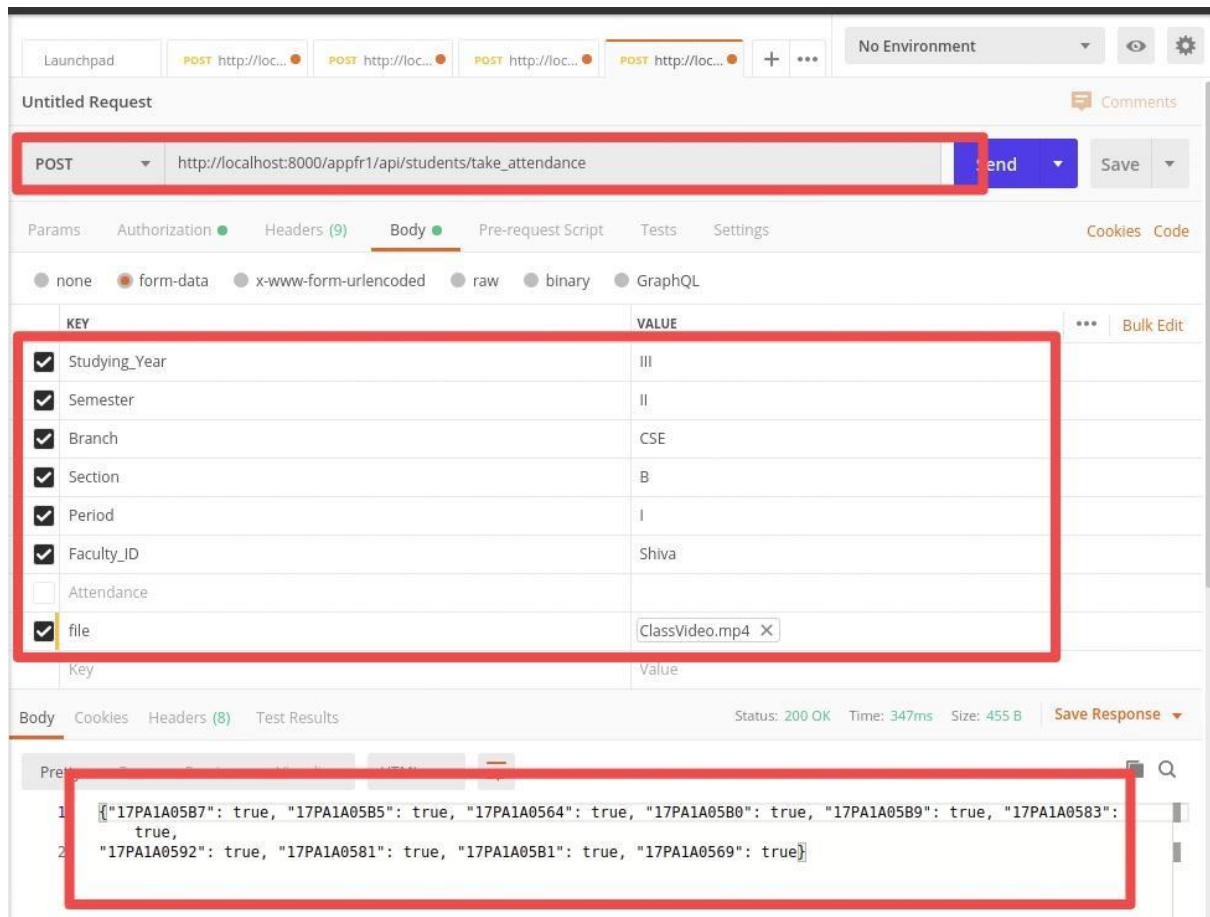


Figure 9.3.1 Image indicating the list of students identified in the video

FRONT SCREENS & EXPLANATION

10. Front Screens And Explanation

10.1 Login System

Initially, on opening the web application in the browser, the teacher is presented with a login page where the teacher is supposed to log in. The teacher is expected to enter their faculty id and password which are given to every one by the admin itself. On logging in, this user name and password will be sent to an application running on a server and then Json Web Token authentication will come into picture. Here the faculty credentials are validated and a unique 256 bit token will be generated which is sent back to frontend application. There is also a refresh token generated which is used in the case of unconditional token errors or token expirations. These tokens along with username are stored in localstorage of the user's browser and this access token is used for authentication in every subsequent request made by the user. Whenever the token expires, the user needs to login again to continue using the application.

When the token expires, then a new token will be generated by the backend application. By default, the validity of the token is 5 minutes but we had extended the limit to 1 day. After that time, the user will be logged out and needs to re login again. The fields of username and password are set to required so that user can't miss any of two fields and he/she can't login without that. Apart from this, an error text will also be generated if the user tries to login into the application without entering both the details. Eventually, after verifying credentials, generating authentication token and storing them in the browser's localstorage, user will be redirected to a dashboard component, where he/she can perform any further operations like log out, upload attendance etc. Since Angular Framework follows an approach of dividing application into modules, we have dedicated an entire component for the purpose of building login functionality.

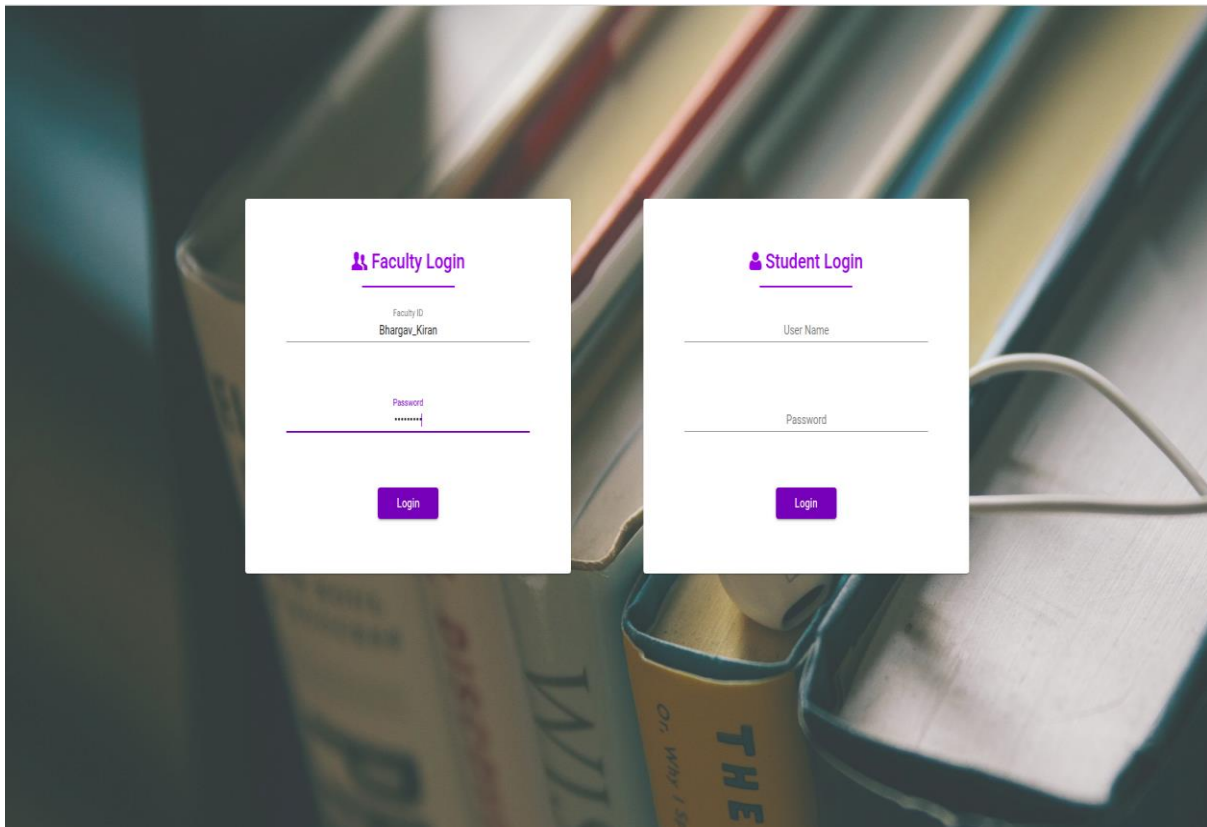


Figure 10.1.1 Common Login Screen for Teacher and Student

10.2 Dashboard Component

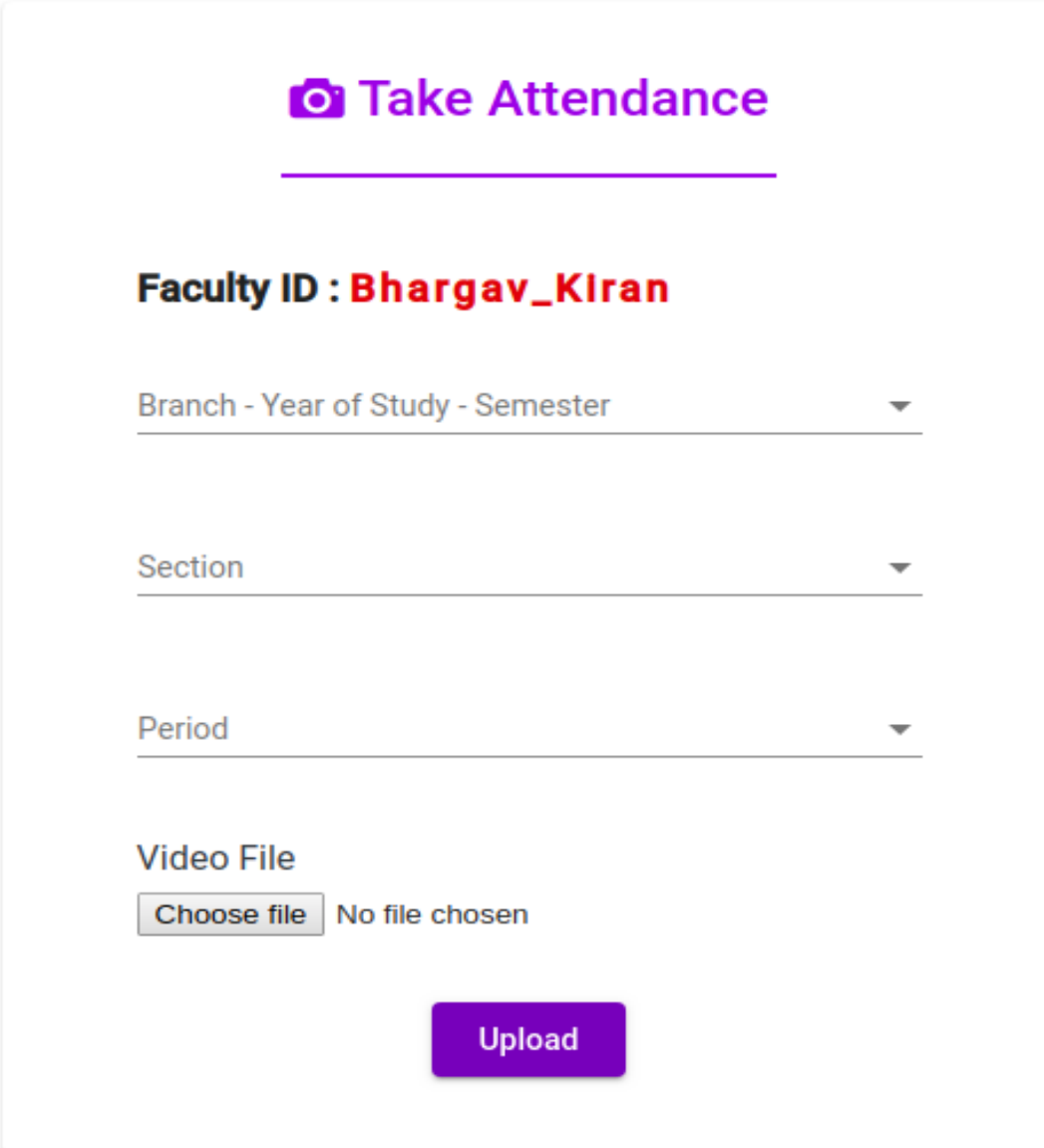
The dashboard component is an entry point to the application which welcomes with a take attendance and other set of features. The dashboard component initially has a header with a main title of application on the left corner and two buttons to the right side which are menu icon and person icon. The menu icon contains items like View my profile, View allocated classes, display a particular student's attendance, check lectures taken for a particular class, display student's profile and add a student. The person icon has a button for username and other for log out. On clicking the log out option, the user will be redirected to the login page and the authentication, username stored till now will be deleted from browser's local storage.

The view profile button display's faculty's profile, students attendance allows to see a particular student's attendance based on roll number and student's

profile allows to see a candidate's details and finally allocated classes option displays classes allocated for a faculty. Then coming to the rest of the component is a simple card component to take attendance where faculty has to select details from a drop down menu. The first dropdown is a combination of three data fields united into one containing branch, year and semester.

These values are again split into three parts when they are uploaded to the backend server. The next dropdown is the section name and the last one is the period. On selecting the details and then comes the final option to upload a video taken by the teacher. Then after uploading the video, the front end part of the application divides the information contained in first drop down menu into three components and then sends all these small pieces of information along with video to backend application. Now these details filled by faculty are used for filtering students and video is sent to face recognition model. Face Recognition model starts running on a selected list of students by faculty and then after processing is completed, the identified list of students is sent back to the front end part of the application to display elegantly.

The attendance which is once for a particular session can't be taken again once it is saved into the database. This is done in order to prevent multiple attendances if any. The database also stores date and time on which the particular attendance is taken so that it can act as timestamp and also for querying results if needed. Since the attendance video once taken is not used anytime after storing the data of students into a database, the video data will be deleted everyday after the classes were completed and even a feature is under development to delete the video instantly after attendance is saved.



Take Attendance

Faculty ID : Bhargav_Kiran

Branch - Year of Study - Semester ▼

Section ▼

Period ▼

Video File

No file chosen

Figure 10.2.1 Front screen of attendance upload with ID of a faculty

10.3 Display Data

Now, on submitting attendance data successfully and validation by face recognition model is completed in the backend, the list of present students is sent back again to the front end application for cross checking by the teacher. The data by default is generated in the form of a dictionary with a key value as student roll number and it's values as either True or False. For frontend application to parse this data.

It has to be converted into JSON format. So, we are converting this dictionary object data into json data and then this json data will be stored into a local array in angular application and then after storing process is completed, then this data will be displayed in an aesthetic way in tabular form. Since it takes a couple of seconds to process video data and receive response from backend, a spinner animation will be displayed till results are received and when results are received, spinner animation will go off. The table will be displayed in a green header with two columns namely Roll Number and Status.

Roll Number	Status
17PA1A05B7	True
17PA1A05B5	True
17PA1A0564	True
17PA1A05B0	True
17PA1A05B9	True
17PA1A0583	True
17PA1A0592	True
17PA1A0581	True
17PA1A05B1	True
17PA1A0569	True

Figure 10.3.1 Image of identified students as output in frontend application

FUTURE WORK

11. Future Goal

11.1 Short Term Goal

Our work on this project hasn't been completed by what we have done till now. Since it's a big project and everything can't be done in a couple of months, maintenance is quite necessary to know about the bugs that can be popped up at runtime of the project when deployed into production. So maintenance errors need to be detected and should be resolved within no time. Apart from maintenance things, there's a set of work that has been reserved for future purpose which is like inducing updates into existing project and giving our best to users. Right now, we're in a state where we have developed a complete full stack application for taking attendance and to process the attendance, a machine learning model is also used. As they are two different components, we have carefully integrated them so that they will work flawlessly.

Update 1:

The existing version works by taking necessary details of class which faculty needs to take attendance and then uploads images of the class of students which are captured by faculty himself. As a future work, we have decided to add a live streaming video capture option that works by taking a live video of students and uploading it to database which will be processed by dividing the entire video into several frames and then processing each frame against the section or class selected by the faculty who is taking attendance. This benefits that can be seen in this model is that it reduces the effort taken by faculty to upload several images which can be replaced with a single live video. Also, another advantage which can be seen is the time taken to upload several images as compared to uploading a video. The other beneficial thing which can be observed on the backend part of the application is the memory taken to store several images against the memory taken to store a single compressed video.

11.2 Long Term Goal

The existing version right now is completely dedicated exclusively for use of faculty. As a further update, we would like to add a student login system where students with their respective registration numbers and password can login to the portal and check out their attendance and their self details, number of backlogs, class wise attendance etc. This addition makes the application useful for both teachers and students. A plan we have made to do this is by having a separate database table for students to store their details, username and password and then linking registration number of student with all other tables by using foreign key attribute.

Through the development of necessary APIs, the communication between frontend and backend parts can be supported. Eventually, the goal is to develop an attendance checking system for students and a profile page for them just like faculty.

Update 2:

In this present version of application, the database that we are using is sqlite3 which comes inbuilt with Django framework. But the problem that arises with this database is that it is a lightweight database and can't handle high level functionality that many databases can support these days. For development purposes this database fits well but when it comes to production environment or in big data environment, this database doesn't handle well. So in further advancement of this application, we would like to update the database from sqlite3 to mongo dB which is a non-relational database and has the ability to handle huge data and query processing. As mongo dB is a NoSQL database, data will be stored in json format inside the database and converted to json format for communication between server and database. This will improve the efficiency of the application backend and ultimately improves user experience.

BIBLIOGRAPHY

12. Bibliography

12.1 Resources and Citations Listing

- angular.io/docs

The information that existed in this website paved the way to develop the front end part of the application by utilising the latest features of angular in a possible way.

- material.angular.io/

Complex applications often need immersive UI to make users utilize every feature and try it out. Angular Material combined with Angular Framework made our application to be more aesthetic.

- metaltoad.com/blog/angular-api-calls-django-authentication-jwt

Setting up an authentication system in our application would have become a nightmare for us if this article hadn't come to our rescue. Provided very valuable information on developing a secure authentication system where angular is the primary case on frontend.

- docs.djangoproject.com/en/3.0/

To develop a backend application that is quite satisfying and reducing complexity at every instinct possible, the documentation of django had played a keen role in our project.

- www.django-rest-framework.org/

As we had frontend and backend as two different components developed using two different frameworks, developing API's is obsolete and django rest framework provided a quicker way for us to develop secure and simple Application Programming Interfaces.

- medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

This article played a major role for us in learning about how facial recognition works and also to know more about deep Learning.

- [**www.techiediaries.com/django-rest-image-file-upload-tutorial/**](http://www.techiediaries.com/django-rest-image-file-upload-tutorial/)
Provided a clear insight into uploading files through angular when there are two different frameworks and APIs involved.
- [**www.ijeat.org/wp-content/uploads/papers/v8i3S/C11230283S19.pdf**](http://www.ijeat.org/wp-content/uploads/papers/v8i3S/C11230283S19.pdf)
The research paper that inspired us in the selection of this particular project.
- [**www.researchgate.net/publication/327671423_Automated_Student_Attendance_Management_System_Using_Face_Recognition**](http://www.researchgate.net/publication/327671423_Automated_Student_Attendance_Management_System_Using_Face_Recognition)
A research paper that provided more insight into how a well organized attendance system for students can be developed.
- [**stackoverflow.com/questions/35388993/how-to-design-a-schema-structure-for-database-for-attendance-management-system**](http://stackoverflow.com/questions/35388993/how-to-design-a-schema-structure-for-database-for-attendance-management-system)
Designing a database system for attendance management involves a lot of complexity like managing students, faculty, academic years, branches etc. This query on stackoverflow helped us to achieve a non redundant database system.

APPENDIX

13. Appendix

13.1 Login Component Angular Code Html

```

<div class="Login">
  <div class="justlayer">
    <div class="only-width">
      <div class="FLogin">
        <mat-card>
          <div style="margin-bottom: 10%;"></div>
          <mat-card-title class="Tophead" style="color: blueviolet;">
            <span class="entypo-
users" style="color: blueviolet;"></span> Faculty Login
          </mat-card-title>
          <div style="width: 30%;border: 1px solid blueviolet;margin-
left: auto;margin-right: auto;;margin-bottom: 5%;"></div>
          <mat-card-content class="content">
            <form class="example-
form" [formGroup] = "loginForm" (ngSubmit) = "onSubmit()">
              <mat-form-field class="form-style">
                <input matInput placeholder="Faculty ID" type="text" id = "usern
ame" formControlName="username" [ngClass]="{ 'is-
invalid': submitted && f.username.errors }">
                <div *ngIf="submitted && f.username.errors">
                  <div *ngIf="f.username.errors.required" style="color: red;">Fa
culty ID is required</div>
                </div>
              </mat-form-field>

              <mat-form-field class="form-style">
                <input matInput placeholder="Password" type="password" id = "pas
sword" formControlName="password" [ngClass]="{ 'is-
invalid': submitted && f.username.errors }">
                <div *ngIf="submitted && f.password.errors">
                  <div *ngIf="f.password.errors.required" style="color: red;">Pa
ssword is required</div>
                </div>
              </mat-form-field>

              <button [disabled] = "loading" mat-raised-
button color="primary" class = "button-style">Login</button>
            </form>
            <div style="margin-bottom: 10%;"></div>
          </mat-card-content>
        </mat-card>
      </div>

      <div class="SLogin">
        <mat-card>
          <div style="margin-bottom: 10%;"></div>
          <mat-card-title class="Tophead" style="color: blueviolet;">
            <span class="fontawesome-
user" style="color: blueviolet;"></span> Student Login
          </mat-card-title>

```

```

    <div style="width: 30%;border: 1px solid blueviolet;margin-
left: auto;margin-right: auto;margin-bottom: 5%;"></div>
    <mat-card-content class="content">
      <form class="example-form">
        <mat-form-field class="form-style">
          <input matInput placeholder="User Name">
        </mat-form-field>

        <mat-form-field class="form-style">
          <input matInput placeholder="Password">
        </mat-form-field>
        <button mat-raised-button color="primary" class = "button-
style">Login</button>
      </form>
      <div style="margin-bottom: 10%;"></div>
    </mat-card-content>
  </mat-card>
</div>
</div>
</div>
</div>

```

13.2 Login Component Angular Code TypeScript

```

import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { ActivatedRoute, Router } from '@angular/router';
import { UserService } from '../user.service';

```

```

@Component({
  selector: 'app-login-component',
  templateUrl: './login-component.component.html',
  styleUrls: ['./login-component.component.css'],
})
export class LoginComponentComponent implements OnInit {

```

```

  loginForm: FormGroup;
  loading = false;
  submitted = false;
  returnUrl: string;

```

```

  constructor(
    private formBuilder: FormBuilder,
    private route: ActivatedRoute,
    private router: Router,
    private user: UserService
  ) {}

```

```

  ngOnInit() {
    this.loginForm = this.formBuilder.group({
      username: ['', Validators.required],
      password: ['', Validators.required]
    });
  }

```

```

    });

    this.returnUrl = this.route.snapshot.queryParams['returnUrl'] || '/';
  }

  get f() {
    return this.loginForm.controls;
  }

  onSubmit() {

    this.submitted = true;
    if (this.loginForm.invalid) {
      return;
    }

    const username = this.loginForm.value.username;
    const password = this.loginForm.value.password;
    this.user.authenticate(username, password).subscribe(
      result => {
        localStorage.setItem('token', result['access']);
        localStorage.setItem('refresh', result['refresh']);
        this.router.navigate(['/dashboard']);
        if (!this.user.userset) {
          this.user.userset = true;
          localStorage.setItem('username', username);
        }
      },
      error => {
        console.log('error');
      }
    );
  }
}

```

13.3 Django Models Code In Models.py

```

from django.db import models
from jsonfield import JSONField
from django.contrib.auth.models import User

class Branche(models.Model):
    Branch = models.CharField(max_length = 256, primary_key = True)
    def __str__(self):
        return self.Branch

class StudyingYear(models.Model):
    Studying_Year = models.CharField(max_length = 256, primary_key = True)

```

```

def __str__(self):
    return self.Studying_Year

class Semester(models.Model):
    Semester = models.CharField(max_length = 5, primary_key = True)
    def __str__(self):
        return self.Semester

class Section(models.Model):
    Section = models.CharField(max_length = 256, primary_key = True)
    def __str__(self):
        return self.Section

class Period(models.Model):
    Period = models.CharField(max_length = 5, primary_key = True)
    StartTime = models.TimeField()
    EndTime = models.TimeField()
    def __str__(self):
        return self.Period

class Student(models.Model):
    Roll_Number = models.CharField(max_length=10, default = "")
    Branch = models.ForeignKey(Branche, on_delete=models.CASCADE)
    Studying_Year = models.ForeignKey(StudyingYear, on_delete=models.CASCADE)
    Section = models.ForeignKey(Section, on_delete=models.CASCADE)
    Semester = models.ForeignKey(Semester, on_delete=models.CASCADE)
    Encoding = JSONField(default = "")
    def __str__(self):
        return self.Roll_Number

class Allocate_class(models.Model):
    Faculty_ID = models.ForeignKey(User, on_delete=models.CASCADE)
    Allocated_Branch = models.ForeignKey(Branche, on_delete=models.CASCADE)
    Allocate_Studying_Year = models.ForeignKey(StudyingYear, on_delete=models.C
ASCASE)
    Allocated_Section = models.ForeignKey(Section, on_delete=models.CASCADE)
    Allocated_Semester = models.ForeignKey(Semester, on_delete=models.CASCADE)
    Allocated_Period = models.ForeignKey(Period, on_delete=models.CASCADE)
    Day_of_Week = models.CharField(max_length = 10, choices = DAYS_OF_WEEK)

    class Meta:
        unique_together = (("Allocated_Branch", "Allocate_Studying_Year", "Alloc
ated_Section", "Allocated_Period", "Day_of_Week"),)

    def __str__(self):
        return str(self.Faculty_ID)

class Video(models.Model):

```



```
Video = models.FileField()
def __str__(self):
    return str(self.Video)

class Attendance(models.Model):
    Studying_Year = models.ForeignKey(StudyingYear,on_delete=models.CASCADE)
    Semester = models.ForeignKey(Semester,on_delete=models.CASCADE)
    Branch = models.ForeignKey(Branche,on_delete=models.CASCADE)
    Section = models.ForeignKey(Section, on_delete=models.CASCADE)
    Period = models.ForeignKey(Period, on_delete=models.CASCADE)
    Faculty_ID = models.ForeignKey(User,on_delete=models.CASCADE)
    Date = models.DateField(auto_now_add=True)
    Time = models.TimeField(auto_now_add=True)
    Attendance = JSONField(default = "")

    class Meta:
        unique_together = (("Studying_Year", "Semester", "Branch", "Section", "Per
iod", "Faculty_ID", "Date"),)

    def __str__(self):
        return str(self.Date)
```