BIRLA VISHVAKARMA MAHAVIDYALAYA
COURSE CODE : 4CP33

Present By 19CP205- Mehul Kurkute

MENTOR INFORMATION

INTERNAL GUIDE

PROF. KIRTI SHARMA

Mr. Parimal Solanki

CONVENER

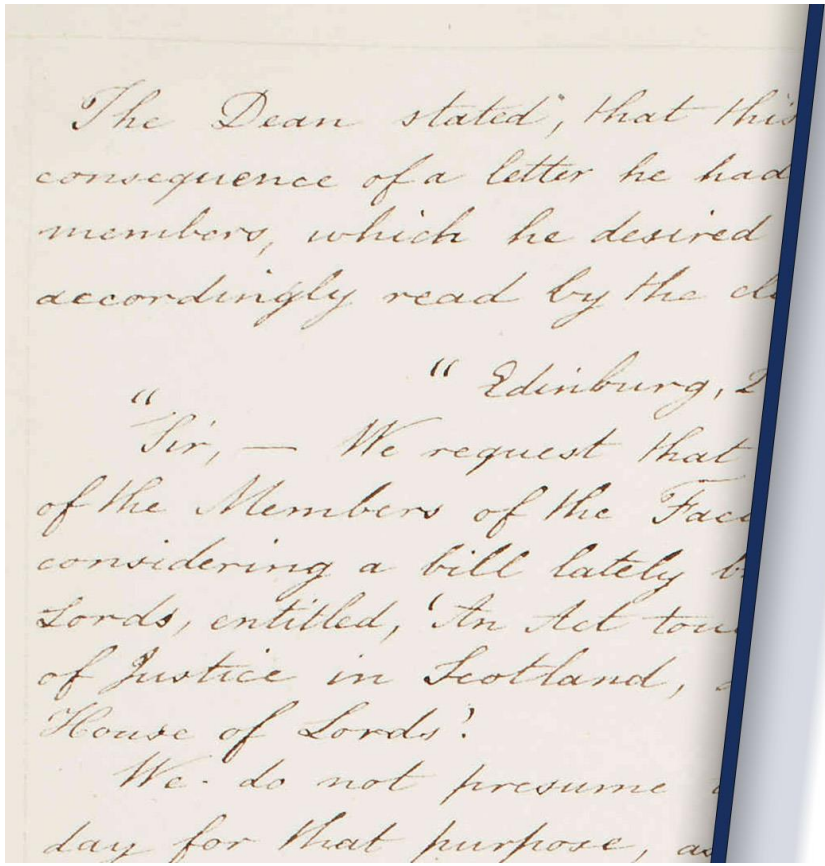
PROF. MAHASWETA J. JOSHI

INDUSTRY GUIDE

MR. HARSH KIRATSATA

Mr. Sidharth Patel

HANDWRITTEN TEXT RECOGNITION



The Dean stated, that this meeting was called in
consequence of a letter he had received from sev
members, which he desired might be read. It wa
accordingly read by the clerh and is as follows.
"Edinburg, 25th Nov. 1807
We request that you will call a mecting
of the Members of the Faculty, for the purpose
considering a bill lately brouglt into the House
Lords, entitled, An Act touching the Administr
of Justice in Scotland, and touching Appeals to
House of Lords.
W do not presume to suggest any particular-

INDEX

INTRODUCTION

Handwritten Text Recognition (HTR) systems transcribe text contained in scanned images into digital text

Handwriting recognition has been one of the most fascinating and challenging research areas in field of image processing and pattern recognition in the recent years

It contributes immensely to the advancement of automation process and improves the interface between man and machine in numerous applications.

PROBLEM STATEMENTS

- Fail to covert handwritten text to machine-readable text
- Fail to translate handwritten text
- Fail to digitalize handwritten notes and handwritten attendance.
- Time consume to check handwritten MCQ answer sheet.

EXISTING SOLUTION

ABBYY FindReader

MyScript

OBJECTIVE

- ✓Recognize handwritten documents, which include characters, words, lines, paragraphs, etc.
- ✓Translating one Language to Another
- ✓Digitizing Handwritten Notes in pdf format
- ✓Digitizing Handwritten Attendance Sheet
- ✓Make a system that can evaluate handwritten MCQ Answer sheet with the correct answer

APPLICATIONS OF PROJECT

- ✓Convert Handwritten to Machine readable code.
- ✓Translate one Language to Another.
- ✓Digitizing Handwritten Notes.
- ✓Digitizing Handwritten Attendance Sheet.
- ✓Handwritten MCQ Test Checker.

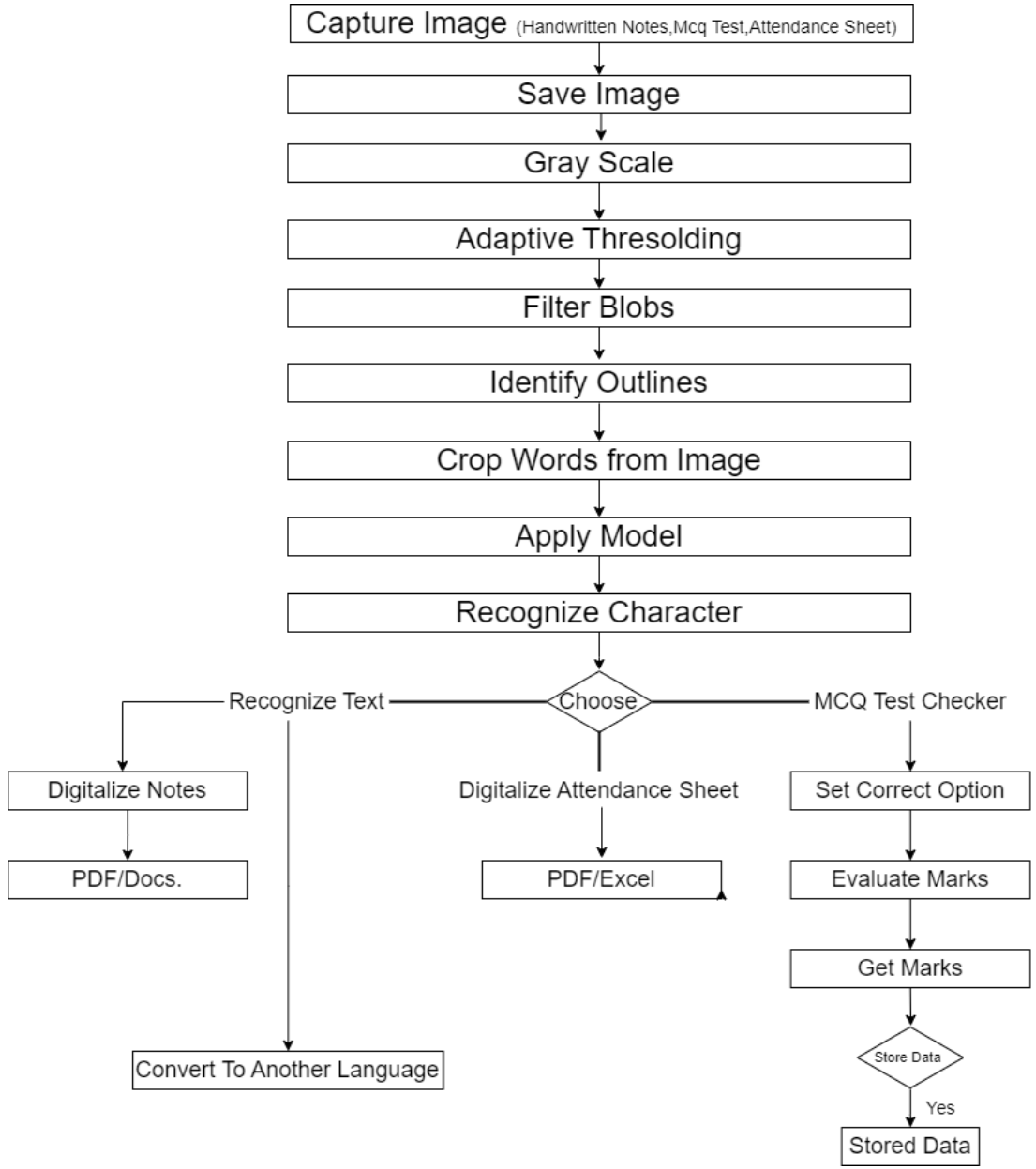
FUNCTIONAL REQUIREMENTS

- ✓ Image pre-processing
- ✓ Character segmentation
- ✓ Feature extraction
- ✓ Machine learning model
- ✓ Converting Handwritten Text
- ✓ User interface
- ✓ Live Handwritten Text Recognition
- ✓ Convert Text into speech
- ✓ Digitizing Handwritten Notes
- ✓ Digitizing Attendance
- ✓ Handwritten MCQ test checker
- ✓ Storing data into database
- ✓ Filter and visualize results
- ✓ Generate Result
- ✓ History

TECHNOLOGY USED

- **Technologies**
  - Python, TensorFlow, Keras
  - OpenCV, Object detection, Image-processing
  - Optical Character Recognition (OCR)
  - Tkinter For GUI
- **Tools**
  - Jupyter Notebook
  - Visual Studio Code

FLOW-CHART


Capture Image (Handwritten Notes,Mcq Test,Attendance Sheet)
Save Image
Gray Scale
Adaptive Thresolding
Filter Blobs
Identify Outlines
Crop Words from Image
Apply Model
Recognize Character
Choose
Recognize Text
MCQ Test Checker
Digitalize Notes
PDF/Docs.
Digitalize Attendance Sheet
PDF/Excel
Set Correct Option
Evaluate Marks
Get Marks
Store Data
Yes
Stored Data
Convert To Another Language

INDEX OF IMPLEMENTATION

- ✓Dataset
- ✓Model Implementation
- ✓Image-processing
- ✓Handwritten MCQ Test Checker
- ✓Storing marks into database
- ✓Filter and visualize marks
- ✓Digitalize Attendance Sheet

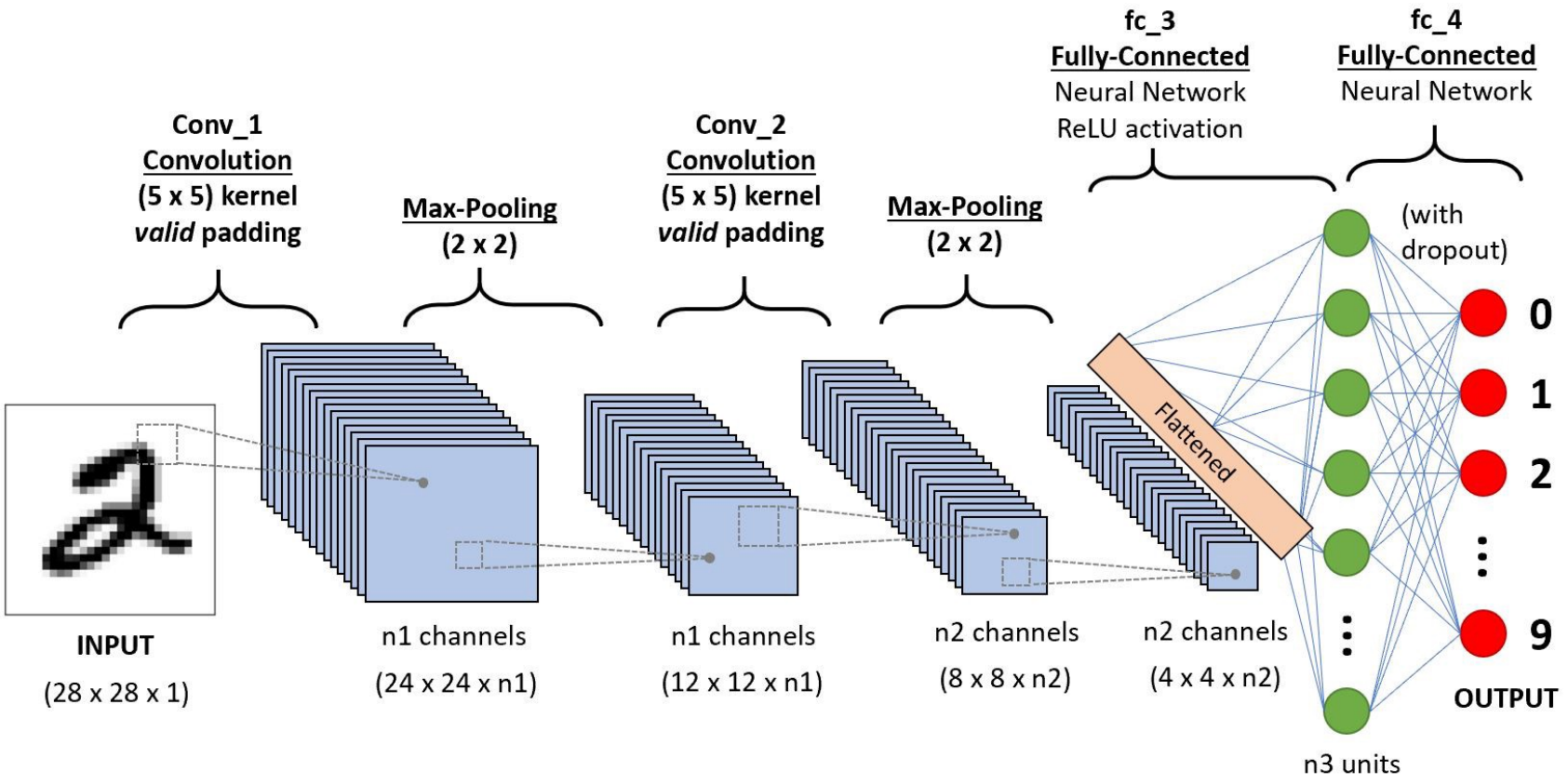
IMPLEMENTATION

✓ **Dataset**

1. MNIST Dataset

2. IAM Dataset

MODEL OVERVIEW



CNN-MODEL ON MNIST DATASET FOR HANDWRITTEN DIGIT RECOGNITION

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 25, 25, 32)        544

 max_pooling2d (MaxPooling2D  (None, 12, 12, 32)       0
 )

 flatten (Flatten)           (None, 4608)              0

 dense (Dense)               (None, 128)               589952

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 591,786
Trainable params: 591,786
Non-trainable params: 0
_________________________________________________________________
```

Fig 4.3 Model Summary

```
Epoch 1/10
1875/1875 [==============================] - 56s 28ms/step - loss: 0.1410 - accuracy: 0.9577 - val_loss: 0.0548 - val_accuracy:
0.9808
Epoch 2/10
1875/1875 [==============================] - 51s 27ms/step - loss: 0.0511 - accuracy: 0.9840 - val_loss: 0.0409 - val_accuracy:
0.9868
Epoch 3/10
1875/1875 [==============================] - 51s 27ms/step - loss: 0.0338 - accuracy: 0.9897 - val_loss: 0.0495 - val_accuracy:
0.9829
Epoch 4/10
1875/1875 [==============================] - 52s 28ms/step - loss: 0.0229 - accuracy: 0.9930 - val_loss: 0.0466 - val_accuracy:
0.9854
```

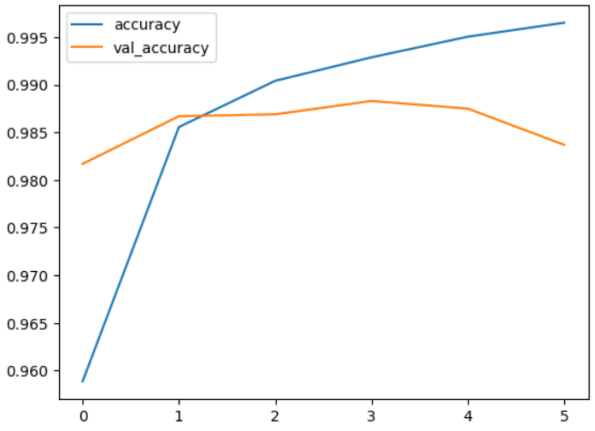
Fig 4.4 Model Training

CNN-MODEL ON MNIST DATASET

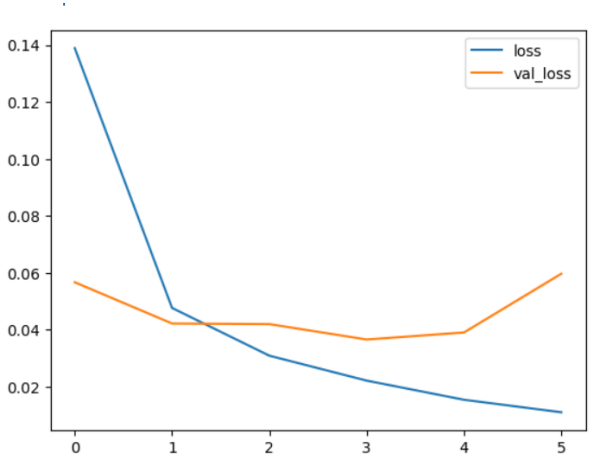
39]:

	loss	accuracy	val_loss	val_accuracy
0	0.138913	0.958867	0.056669	0.9817
1	0.047644	0.985567	0.042181	0.9867
2	0.030867	0.990417	0.041966	0.9869
3	0.022113	0.992883	0.036557	0.9883
4	0.015391	0.995050	0.039029	0.9875

Model Training



Training Accuracy Vs Validation Accuracy



Training Loss Vs Validation Loss

CRNN-MODEL ON IAM DATASET

```
423/423 [==============================] - 612s 1s/step - loss: 15.6534
Epoch 2/10
423/423 [==============================] - 615s 1s/step - loss: 13.9153
Epoch 3/10
423/423 [==============================] - 610s 1s/step - loss: 13.1496
Epoch 4/10
423/423 [==============================] - 582s 1s/step - loss: 12.5125
Epoch 5/10
423/423 [==============================] - 606s 1s/step - loss: 12.0853
Epoch 6/10
423/423 [==============================] - 747s 2s/step - loss: 11.6634
Epoch 7/10
423/423 [==============================] - 562s 1s/step - loss: 11.2682
Epoch 8/10
423/423 [==============================] - 568s 1s/step - loss: 10.9288
Epoch 9/10
423/423 [==============================] - 574s 1s/step - loss: 10.4279
Epoch 10/10
423/423 [==============================] - 595s 1s/step - loss: 10.0384
Epoch 1/10
428/428 [==============================] - 642s 1s/step - loss: 10.6078
Epoch 2/10
428/428 [==============================] - 649s 2s/step - loss: 9.8129
Epoch 3/10
428/428 [==============================] - 652s 2s/step - loss: 9.2717
Epoch 4/10
428/428 [==============================] - 807s 2s/step - loss: 8.7884
Epoch 5/10
428/428 [==============================] - 907s 2s/step - loss: 8.2483
Epoch 6/10
428/428 [==============================] - 583s 1s/step - loss: 7.7563
Epoch 7/10
428/428 [==============================] - 586s 1s/step - loss: 7.2454
Epoch 8/10
428/428 [==============================] - 592s 1s/step - loss: 6.6551
Epoch 9/10
428/428 [==============================] - 603s 1s/step - loss: 6.0978
Epoch 10/10
428/428 [==============================] - 609s 1s/step - loss: 5.5847
Epoch 1/10
422/422 [==============================] - 600s 1s/step - loss: 9.5774
Epoch 2/10
422/422 [==============================] - 590s 1s/step - loss: 8.1945
Epoch 3/10
422/422 [==============================] - 675s 2s/step - loss: 7.4257
Epoch 4/10
 33/422 [=>............................] - ETA: 10:12 - loss: 6.5487
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input (InputLayer)          [(None, 256, 64, 1)]      0

 conv1 (Conv2D)              (None, 256, 64, 32)       320

 batch_normalization (BatchN  (None, 256, 64, 32)      128
 ormalization)

 activation (Activation)     (None, 256, 64, 32)       0

 max1 (MaxPooling2D)         (None, 128, 32, 32)       0

 conv2 (Conv2D)              (None, 128, 32, 64)       18496

 batch_normalization_1 (Batc  (None, 128, 32, 64)      256
 hNormalization)

 activation_1 (Activation)   (None, 128, 32, 64)       0

 max2 (MaxPooling2D)         (None, 64, 16, 64)        0

 conv3 (Conv2D)              (None, 64, 16, 128)       73856

 batch_normalization_2 (Batc  (None, 64, 16, 128)      512
 hNormalization)

 activation_2 (Activation)   (None, 64, 16, 128)       0

 max3 (MaxPooling2D)         (None, 32, 16, 128)       0

 dropout (Dropout)           (None, 32, 16, 128)       0

 conv4 (Conv2D)              (None, 32, 16, 256)       295168

 batch_normalization_3 (Batc  (None, 32, 16, 256)      1024
 hNormalization)

 activation_3 (Activation)   (None, 32, 16, 256)       0

 max4 (MaxPooling2D)         (None, 16, 16, 256)       0

 reshape (Reshape)           (None, 64, 1024)          0

 dense1 (Dense)              (None, 64, 64)            65600

 lstm1 (Bidirectional)       (None, 64, 512)           657408

 lstm2 (Bidirectional)       (None, 64, 512)           1574912

 dense2 (Dense)              (None, 64, 30)            15390

 softmax (Activation)        (None, 64, 30)            0

=================================================================
Total params: 2,703,070
Trainable params: 2,702,110
```

CNN-MODEL ON IAM DATASET

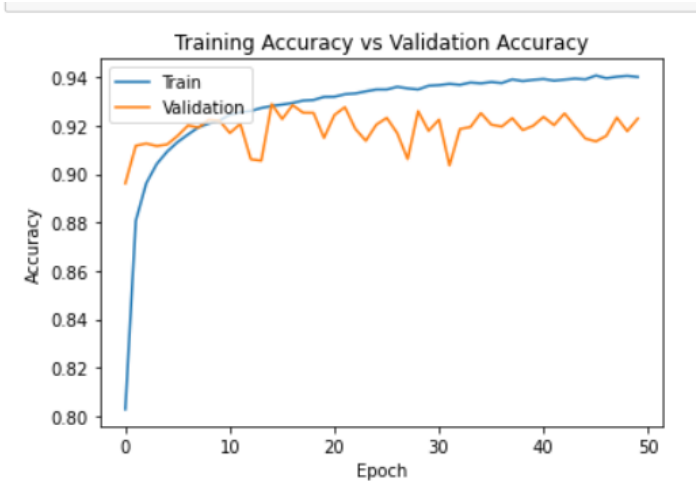
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 5, 5, 128)         73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
dropout (Dropout)            (None, 2, 2, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 128)               65664
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 35)                4515
=================================================================
Total params: 162,851
Trainable params: 162,851
Non-trainable params: 0
_________________________________________________________________
```

Training Loss Vs Validation Loss

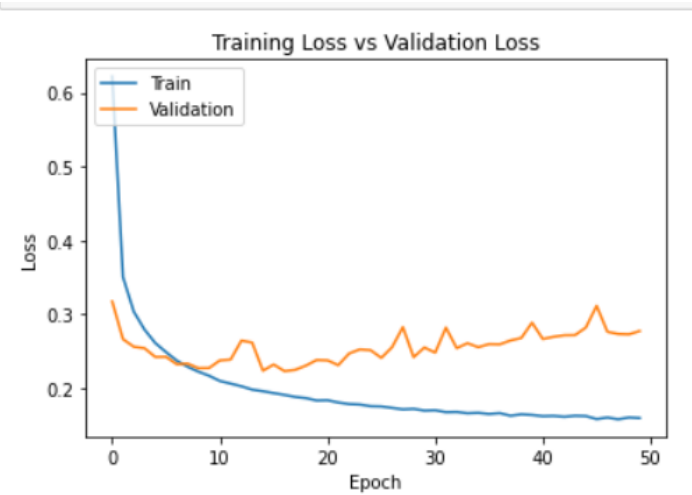
CNN-MODEL ON IAM DATASET

```
In [18]: history = model.fit(train_X,train_Y, epochs=50, batch_size=32, validation_data = (val_X, val_Y),  verbose=1)
y: 0.9197
Epoch 45/50
4375/4375 [==============================] - 13s 3ms/step - loss: 0.1625 - accuracy: 0.9390 - val_loss: 0.2823 - val_accurac
y: 0.9146
Epoch 46/50
4375/4375 [==============================] - 13s 3ms/step - loss: 0.1584 - accuracy: 0.9406 - val_loss: 0.3117 - val_accurac
y: 0.9134
Epoch 47/50
4375/4375 [==============================] - 14s 3ms/step - loss: 0.1605 - accuracy: 0.9395 - val_loss: 0.2765 - val_accurac
y: 0.9157
Epoch 48/50
4375/4375 [==============================] - 14s 3ms/step - loss: 0.1582 - accuracy: 0.9401 - val_loss: 0.2736 - val_accurac
y: 0.9233
Epoch 49/50
4375/4375 [==============================] - 13s 3ms/step - loss: 0.1606 - accuracy: 0.9405 - val_loss: 0.2731 - val_accurac
y: 0.9176
Epoch 50/50
4375/4375 [==============================] - 14s 3ms/step - loss: 0.1598 - accuracy: 0.9400 - val_loss: 0.2775 - val_accurac
y: 0.9228
```

Model Training



Training Accuracy Vs Validation Accuracy

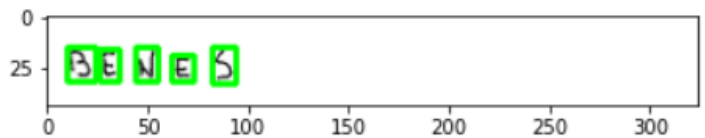


Training Loss Vs Validation Loss

In [25]:
```
letter,image = get_letters("../input/handwriting-recognition/train_v2/train/TRAIN_00003.jpg")
word = get_word(letter)
print(word)
plt.imshow(image)
```

```
BZNES
```

Out[25]:
```
<matplotlib.image.AxesImage at 0x7f8640051a10>
```



In [26]:
```
letter,image = get_letters("../input/handwriting-recognition/train_v2/train/TRAIN_00023.jpg")
word = get_word(letter)
print(word)
plt.imshow(image)
```

```
LIL0U
```

Out[26]:
```
<matplotlib.image.AxesImage at 0x7f86384f8750>
```

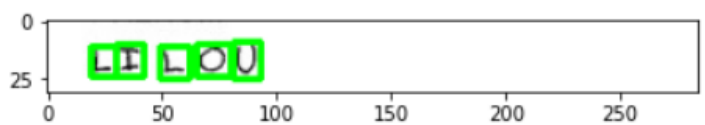
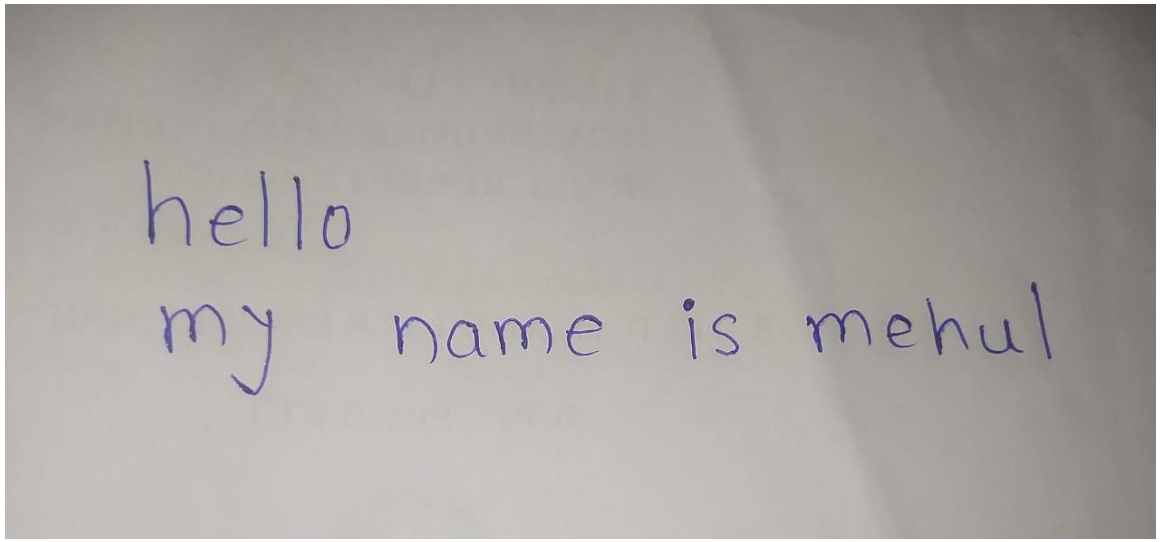
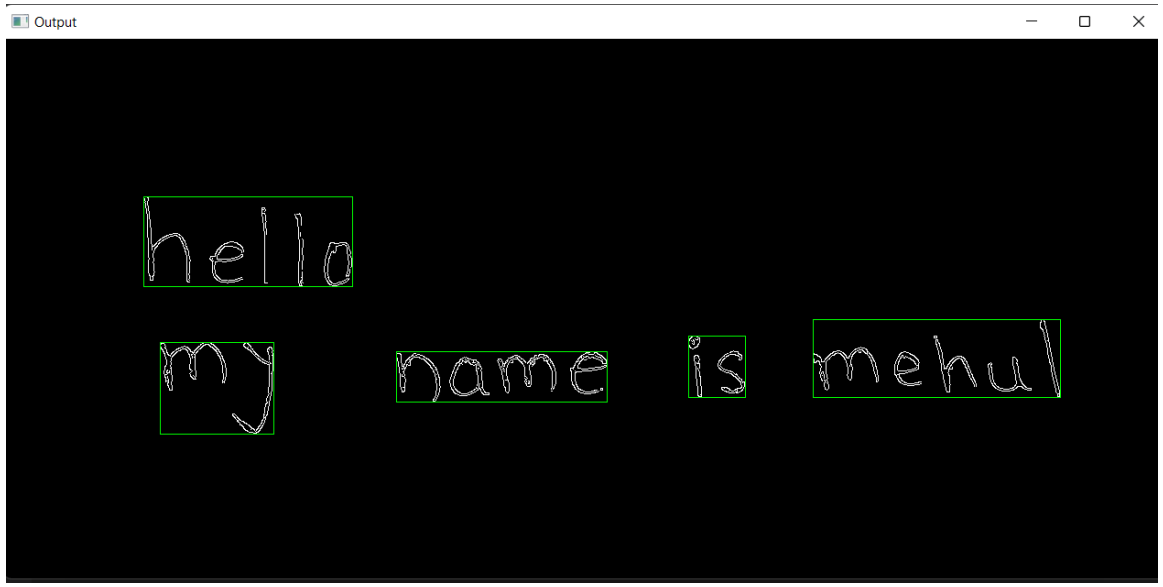


IMAGE-PROCESSING

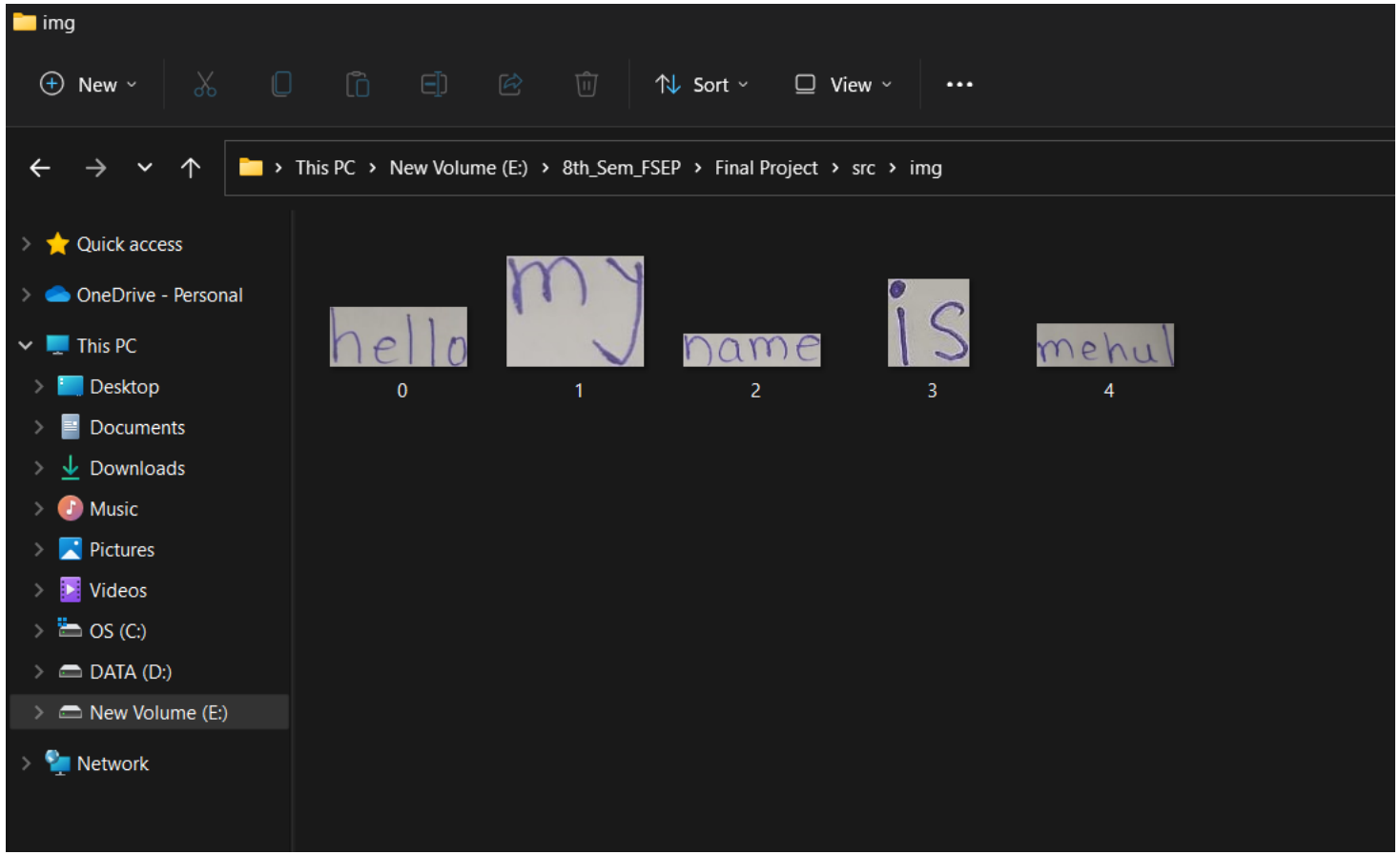


Original Image

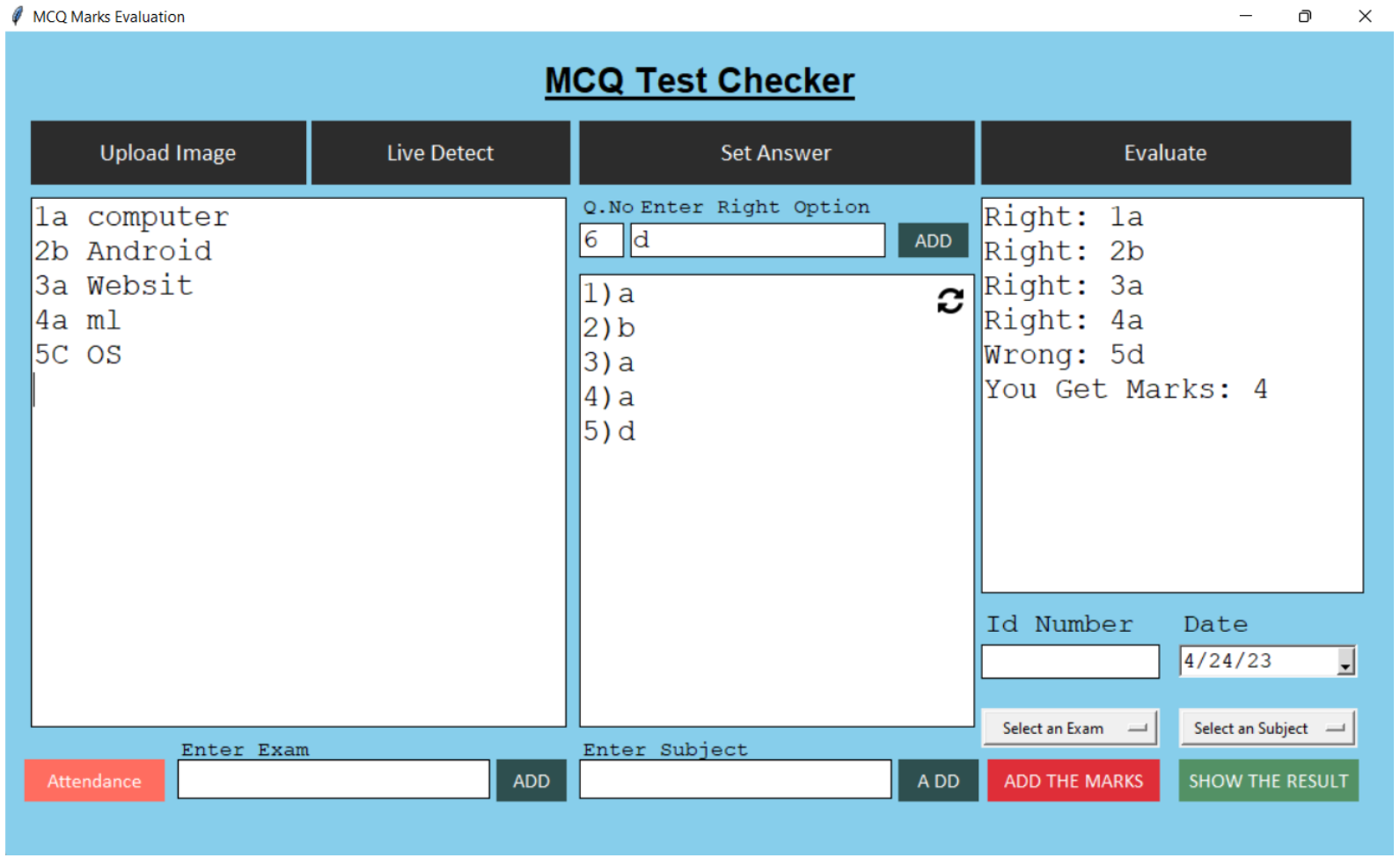


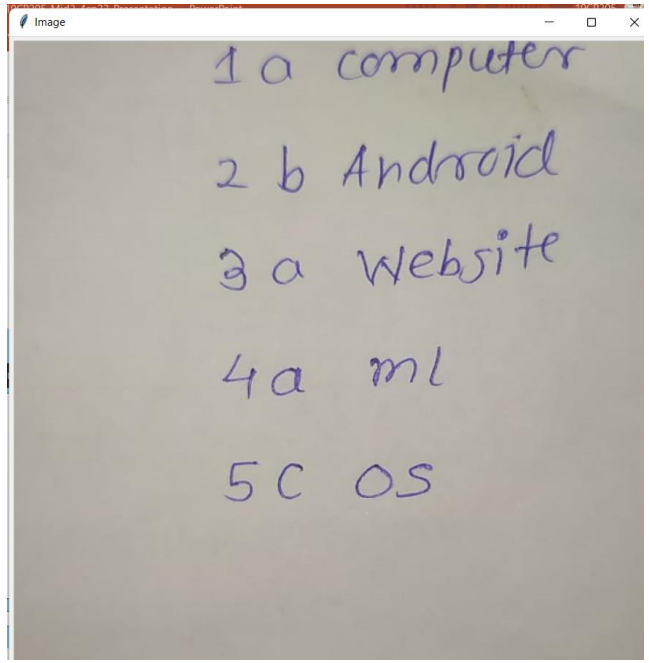
Detect Word Image

CROPPED IMAGE

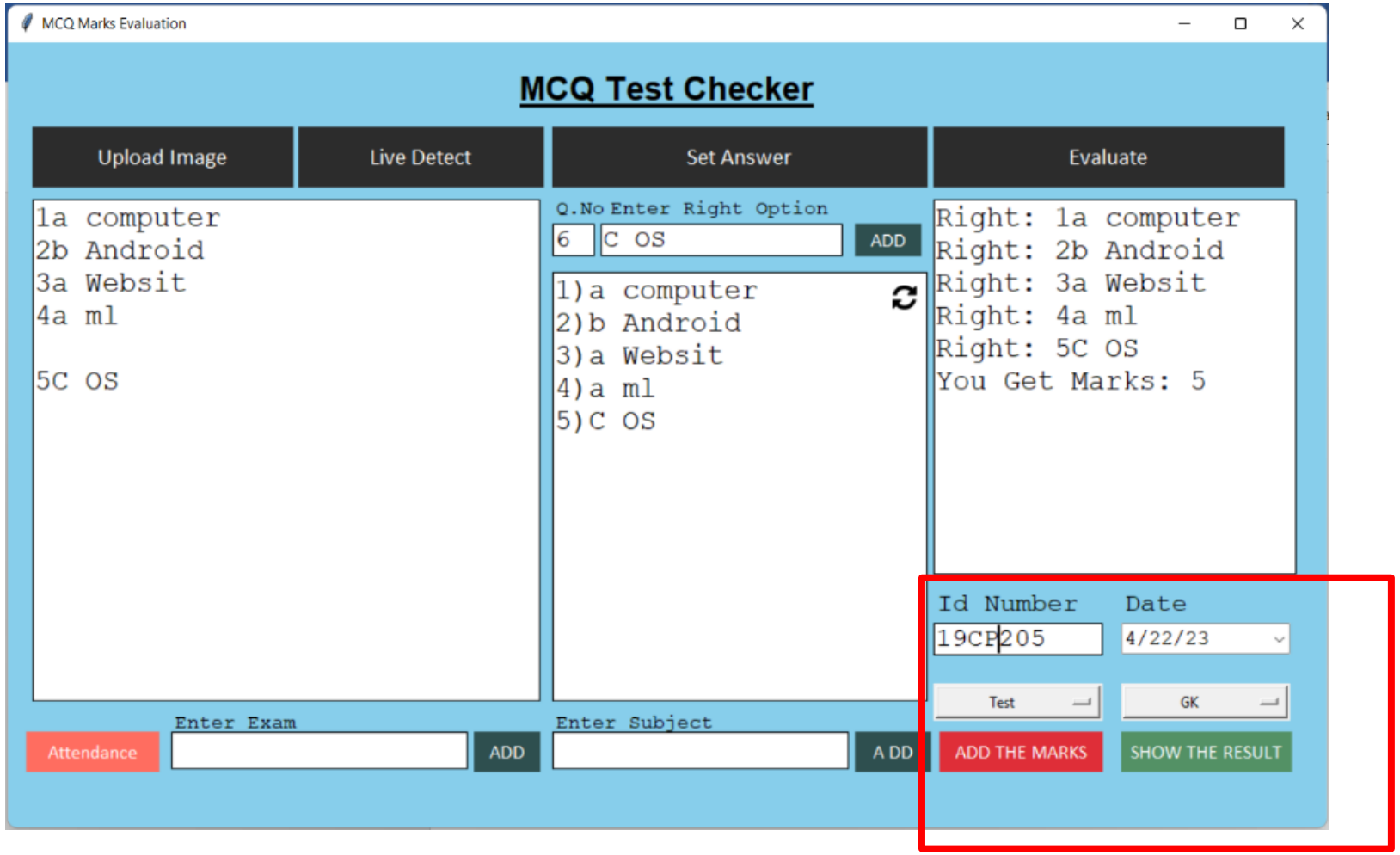


MCQ TEST CHECKER

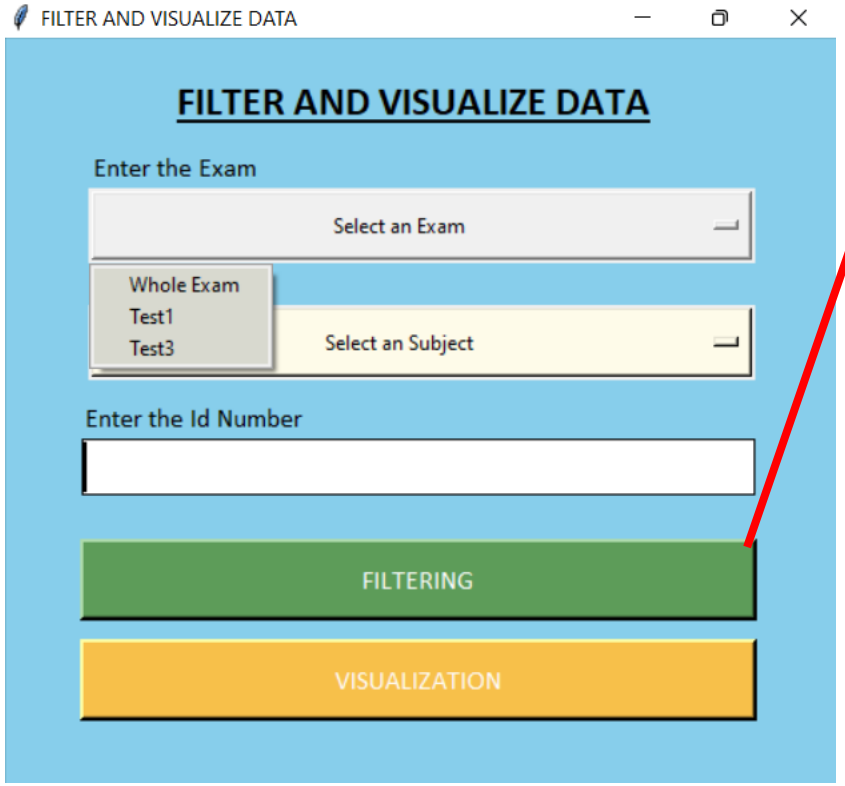


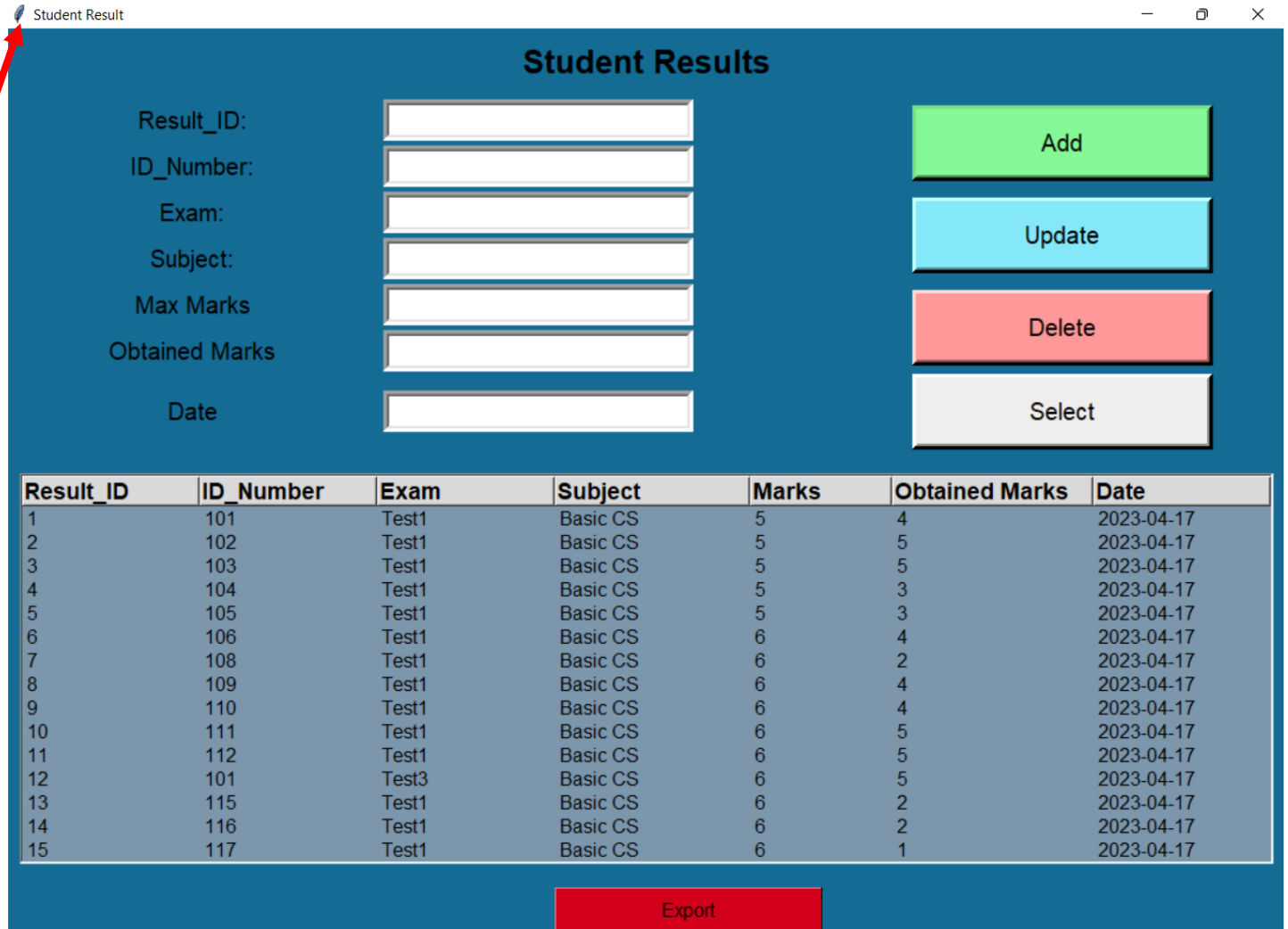


STORING MARKS INTO DATABASE



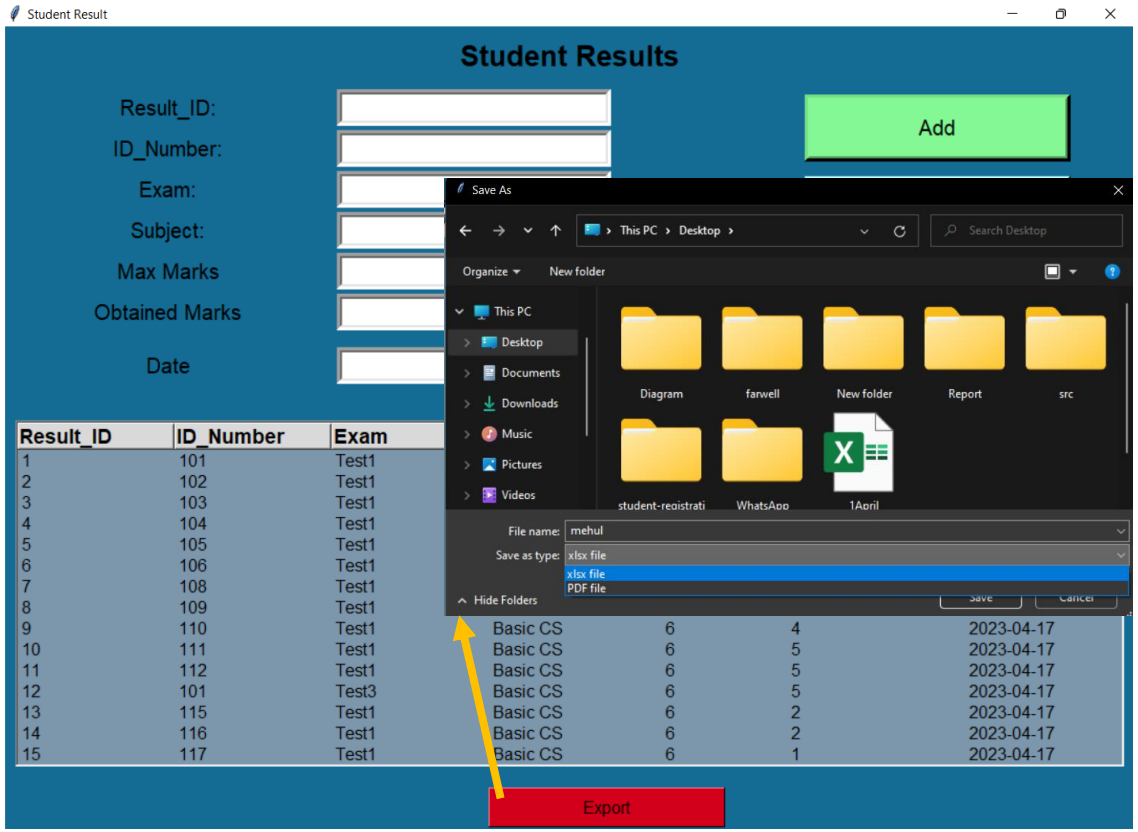
FILTER RESULT





Result_ID	ID_Number	Exam	Subject	Marks	Obtained Marks	Date
1	101	Test1	Basic CS	5	4	2023-04-17
2	102	Test1	Basic CS	5	5	2023-04-17
3	103	Test1	Basic CS	5	5	2023-04-17
4	104	Test1	Basic CS	5	3	2023-04-17
5	105	Test1	Basic CS	5	3	2023-04-17
6	106	Test1	Basic CS	6	4	2023-04-17
7	108	Test1	Basic CS	6	2	2023-04-17
8	109	Test1	Basic CS	6	4	2023-04-17
9	110	Test1	Basic CS	6	4	2023-04-17
10	111	Test1	Basic CS	6	5	2023-04-17
11	112	Test1	Basic CS	6	5	2023-04-17
12	101	Test3	Basic CS	6	5	2023-04-17
13	115	Test1	Basic CS	6	2	2023-04-17
14	116	Test1	Basic CS	6	2	2023-04-17
15	117	Test1	Basic CS	6	1	2023-04-17

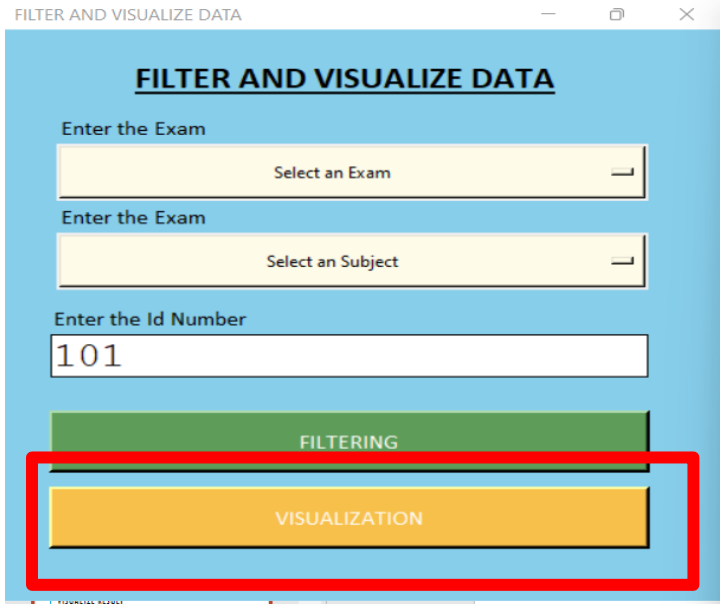
CONVERT RESULT INTO EXCEL SHEET OR PDF

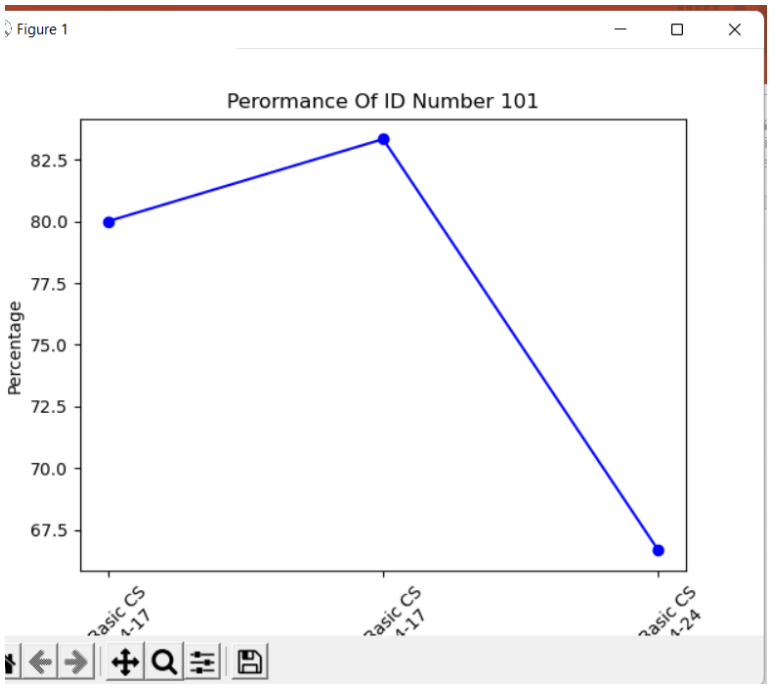


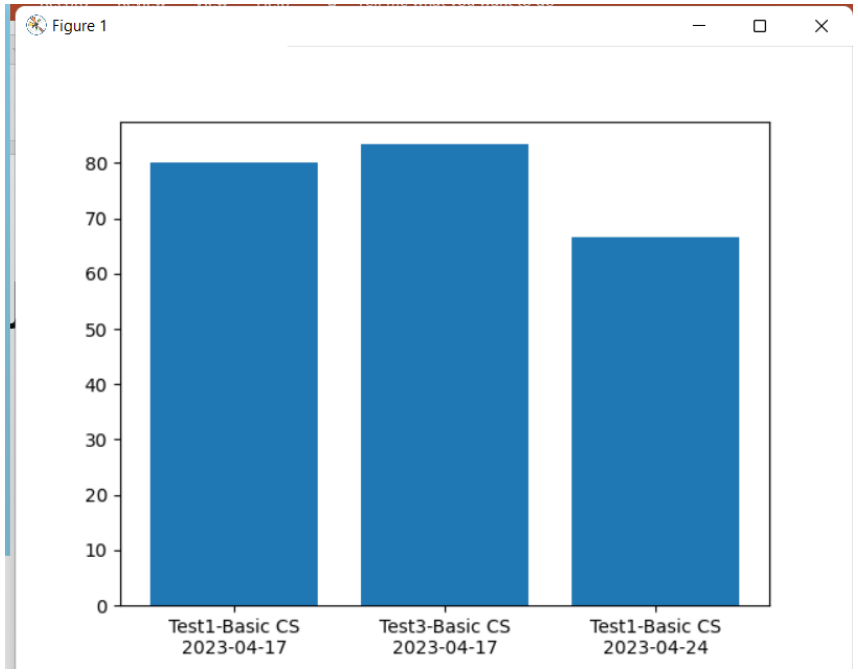
	A	B	C	D	E	F	G
1	Result Num	Id Num	Exam	Sub	Max Marks	Ob marks	Date
2	1	2	mid1	CP	10	10	2023-04-04
3	2	44	mid111	ec	5	10	2023-04-05
4	3	33	Test	GK	10	9	2023-04-05
5	4	35	Test3	GK	10	9	2023-04-05
6	5	55	Test	GK	8	5	2023-04-06
7	11	205	Test	GK	8	8	2023-04-06
8	12	205	Test	GK	8	8	2023-04-06
9	13	205	Test	GK	8	8	2023-04-06
10	14	205	Test	GK	8	8	2023-04-06
11	15	205	Test	GK	8	8	2023-04-06
12	18	205	Test	GK	8	8	2023-04-06
13	19	205	Test	GK	8	8	2023-04-06
14	20	205	Test	GK	8	8	2023-04-06
15	21	205	Test	GK	8	8	2023-04-06
16	22	205	Test	GK	8	8	2023-04-06
17	23	205	Test	GK	8	8	2023-04-06
18	8	33	Test	GK	10	9	2023-04-07
19	25		Select an E	Select an S	10	8	2023-04-17
20	27	33	Test	GK	5	3	2023-04-19
21	29	22	Test	Select an S	9	9	2023-04-19
22	30	1	Test	you	7	7	2023-04-19
23							
24							

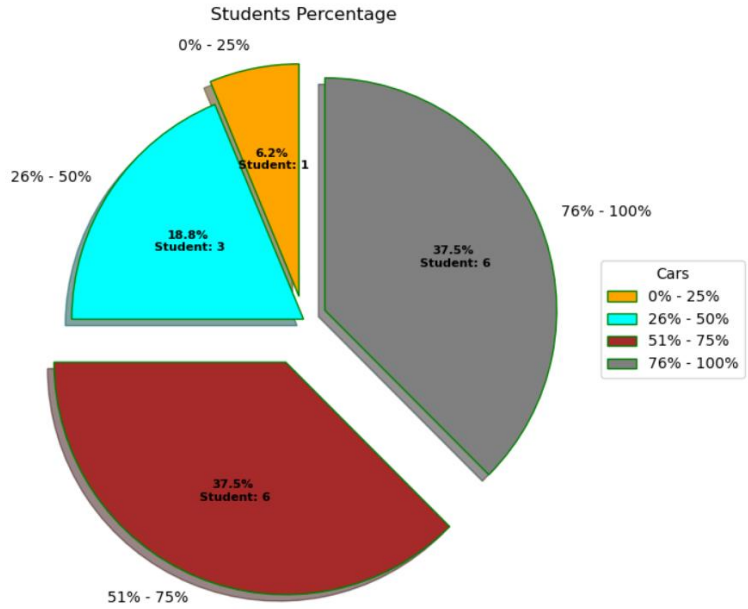
Result Num	Id Num	Exam	Sub	Max Marks	Ob marks	Date
1	2	mid1	CP	10	10	2023-04-04
2	44	mid111	ec	5	10	2023-04-05
3	33	Test	GK	10	9	2023-04-05
4	35	Test3	GK	10	9	2023-04-05
5	55	Test	GK	8	5	2023-04-06
11	205	Test	GK	8	8	2023-04-06
12	205	Test	GK	8	8	2023-04-06
13	205	Test	GK	8	8	2023-04-06
14	205	Test	GK	8	8	2023-04-06
15	205	Test	GK	8	8	2023-04-06
18	205	Test	GK	8	8	2023-04-06
19	205	Test	GK	8	8	2023-04-06
20	205	Test	GK	8	8	2023-04-06
21	205	Test	GK	8	8	2023-04-06
22	205	Test	GK	8	8	2023-04-06
23	205	Test	GK	8	8	2023-04-06
8	33	Test	GK	10	9	2023-04-07
27	33	Test	GK	5	3	2023-04-19
29	22	Test	Select an Subject	9	9	2023-04-19
30	1	Test	you	7	7	2023-04-19

VISUALIZE RESULT

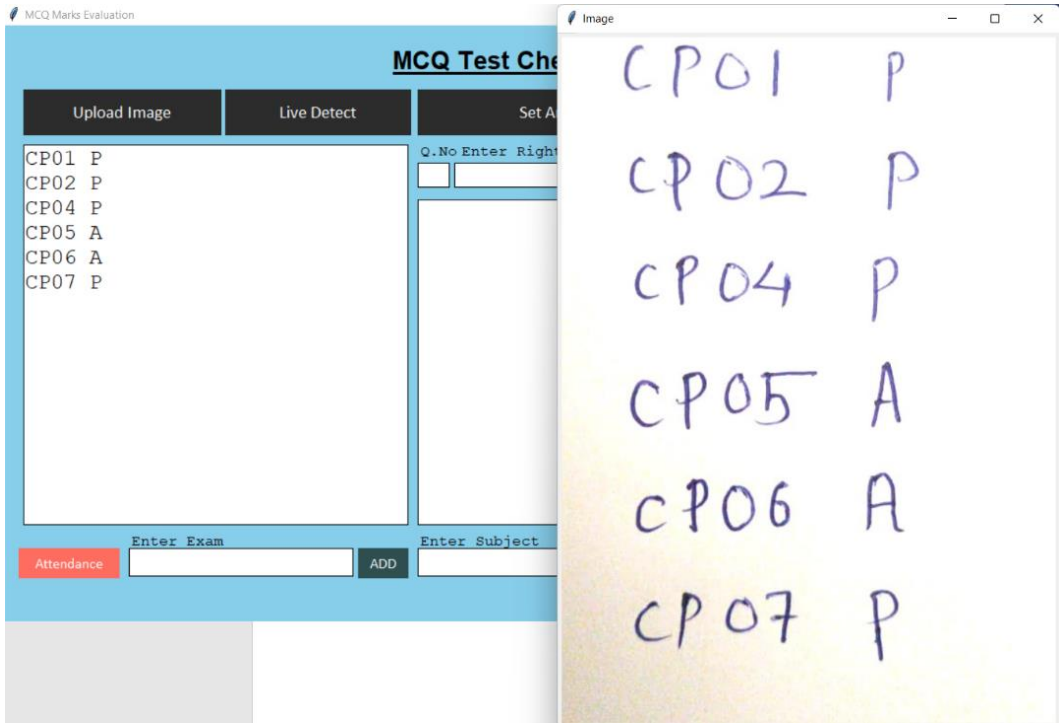


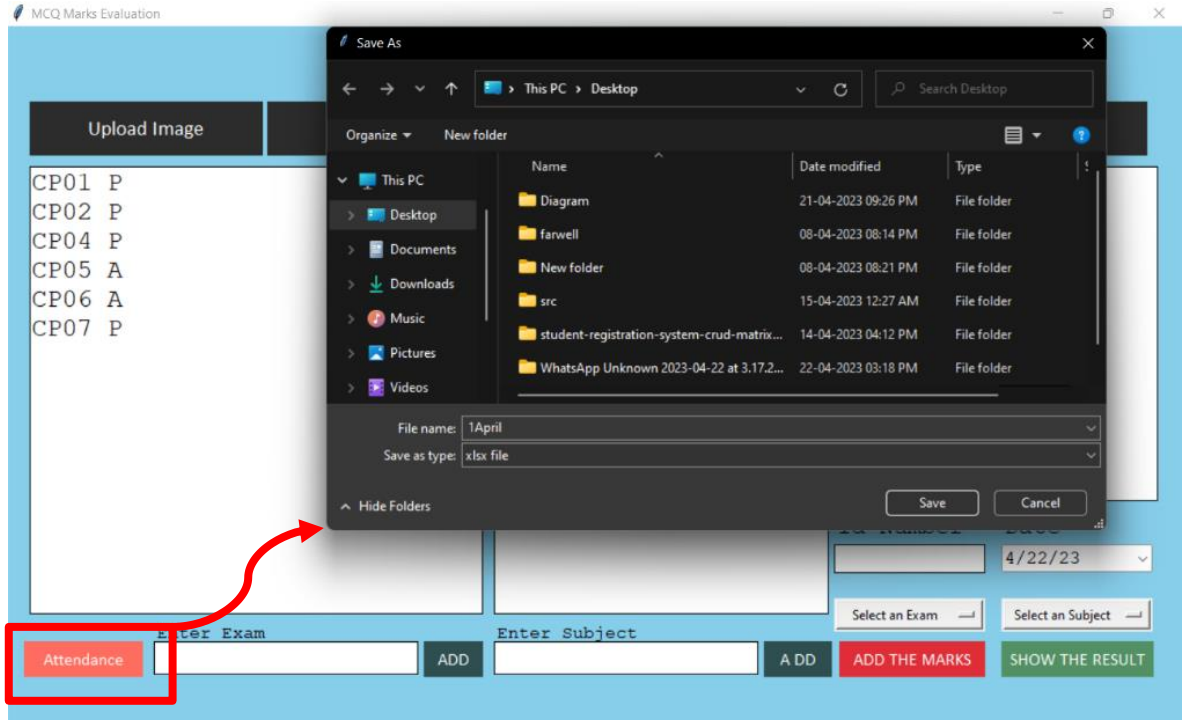






DIGITALIZE ATTENDANCE





DIGITALIZE ATTENDANCE INTO EXCEL FORMAT

	A	B	C	D
1	**ID Number**	**P/A**		
2	CP01	P		
3	cp02	P		
4	cP04	P		
5	cP05	A		
6	CP06	A		
7	cP07	P		
8				
9				
10				

FUTURE SCOPE

- ✓In the future we can increase the accuracy of the model by using larger datasets and more numbers epoch.
- ✓We can implement more applications with the help of handwritten text recognition.
- ✓We can work on different languages of handwritten text like Hindi, Gujarati Marathi.

REFERENCE

- https://www.youtube.com/playlist?list=PLeo1K3hjS3uu7CxAacxVndI4bE_o3BDtO
- https://www.tensorflow.org/api_docs/python/tf
- https://www.tensorflow.org/tutorials/images/cnn
- https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939
- https://docs.opencv.org/4.x/d9/df8/tutorial_root.html
- http://www.fki.inf.unibe.ch/databases/iam-handwriting-database
- https://core.ac.uk/download/pdf/327266589.pdf

THANK YOU