#	Ans	#	Ans	#	Ans	#	Ans
1	C, D	26	A	51	B	76	A
2	B	27	B	52	C	77	B
3	B	28	C	53	A	78	B
4	D	29	B	54	A	79	D
5	A	30	A	55	D	80	C
6	C	31	A	56	C	81	B
7	C	32	D	57	A	82	B
8	C	33	B	58	B	83	D
9	B	34	D	59	C	84	A
10	A	35	B	60	A	85	C
11	B	36	B	61	C	86	D
12	B	37	C	62	B	87	D
13	C	38	A	63	B	88	D
14	D	39	A	64	C	89	C
15	D	40	A	65	C	90	C
16	A	41	D	66	A	91	A
17	C	42	C	67	C	92	B
18	B	43	A	68	C	93	C
19	C	44	D	69	C	94	B
20	C	45	B	70	A	95	B
21	B	46	B	71	C	96	B
22	B	47	C	72	C	97	C
23	D	48	D	73	B	98	D
24	B	49	C	74	C	99	A
25	C	50	B	75	A	100	B

Bubble Sheet Checker

Hamza Iftikhar i192003

Hamza Khalid i192011

DIP | Mr. Shoaib Mehboob

Contents

OMR is used to recognize marks on paper. It is mostly used for checking MCQs. There are a ton of ways you can recognize marks e.g., contour detection, bounding boxes.

Contour Detection:

For this project we used contour detection to first recognize the blocks where actual MCQ markings are.
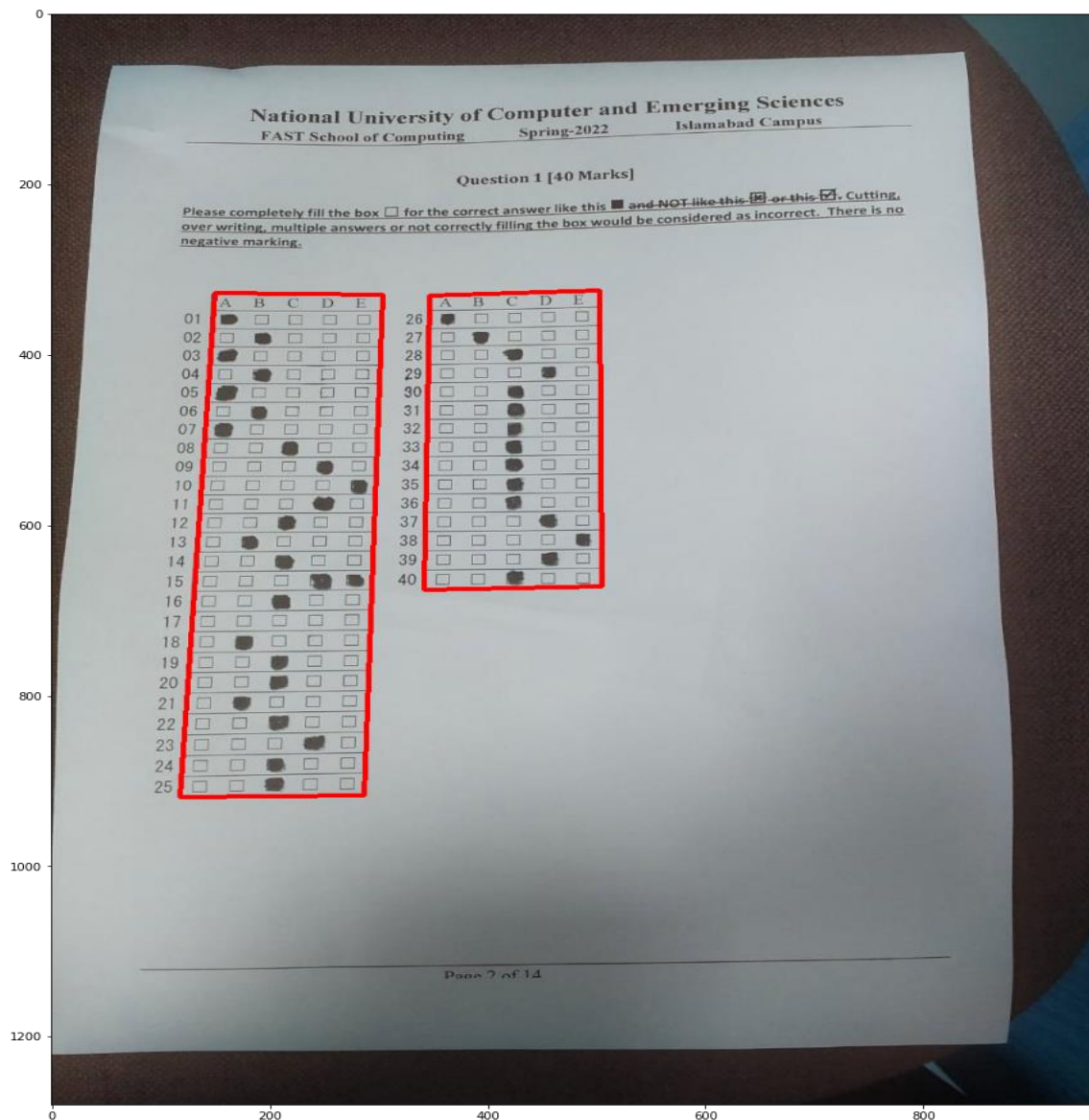
Perspective Transformation:

Then we applied four-point transformation to get the top-down perspective of the exam.

OMR

OMR
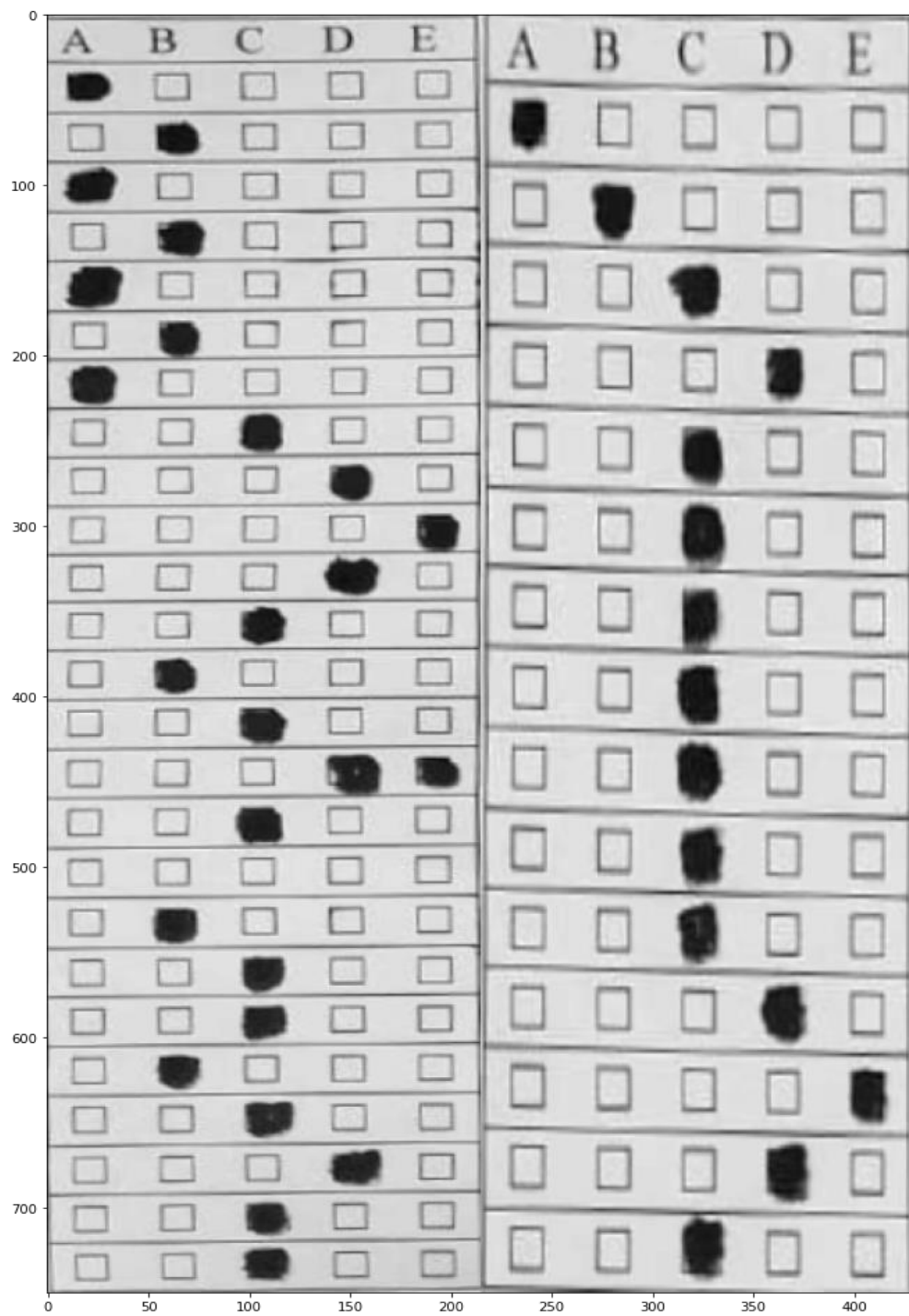
Image Binarization:

Then we first applied Adaptive Gaussian thresholding to get the binary image and then opened (Erosion followed by Dilation) that binary image.
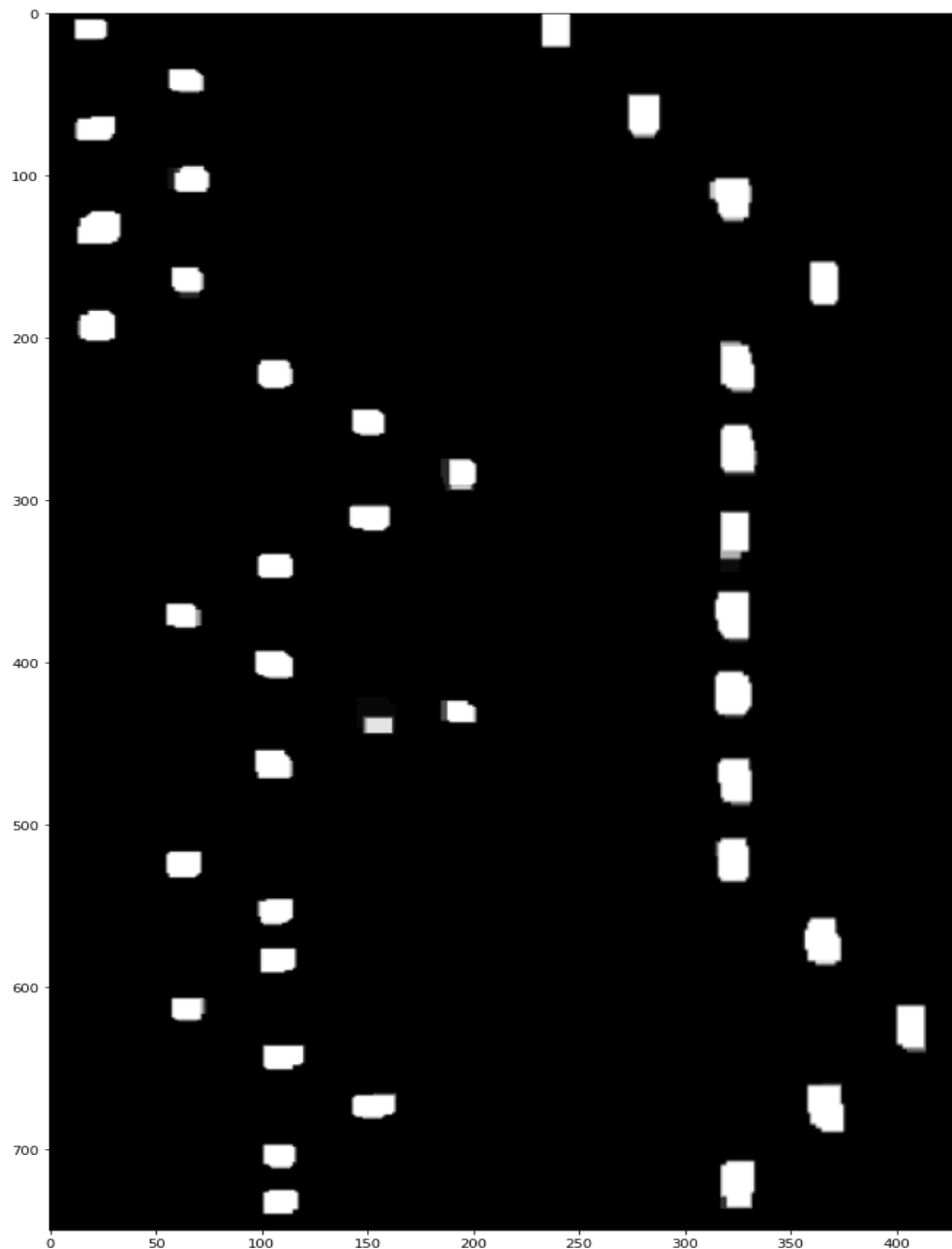
Remember that these threshold-ed marks are for each block of MCQs stacked side by side just like how warped blocks are stacked.

Detecting Choices:

After we got the binary image of square marks, we split the image into a table of sorts where rows were the number of questions and columns were the number of choices for an MCQ. Then we counted the number of non-zero pixels in each square. We chose a threshold value for the purpose of considering a square marked. If the pixels were greater than threshold value then that square was marked as the corresponding option i.e. A, B, C, D, E. If there was more than one square that had greater pixels than threshold it meant that the test taker marked more than one options for an MCQ therefore it was considered as Unmarked. The MCQ was also considered Unmarked if there were no squares containing pixels greater than the threshold value to cater for the missed MCQs.

Detecting Solution:

The solution image was taken by the user and then performed the same steps to that image to get the solution of the exam. Lastly the options of the test taker were compared to that of the solution and the corresponding score was computed.

GUI:

The GUI for System is made with Next.js with FASTAPI as Backend. We have used tailwind for Frontend CSS with Typescript for strict type checking.

Features:

1. Create Multiple Exam Folders
2. Generate Reports
3. Visual view of each exam
4. Generate Report
5. Download Report in CSV
6. Upload a New Solution
7. Upload more images to an existing exam

Creating an Exam:

On the Dashboard, select "Create New Exam"
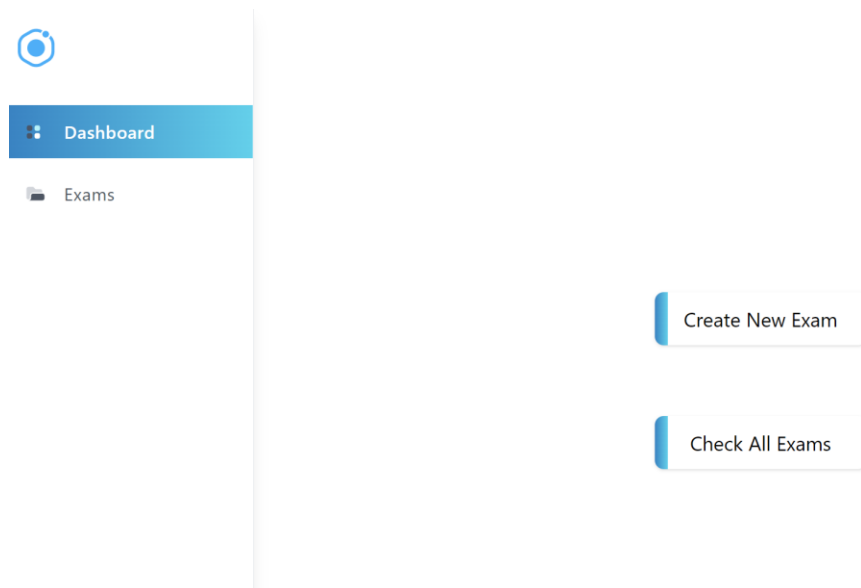
1. Enter Exam name

Enter
Exam Name

[   ]

[Create]

2. Upload Exam Solution

Upload
Exam Solution

Drag and drop your file here
or
browse to upload

Upload

3. Upload all the Students answers

Upload
Exam Images

Drag and drop your file here
or
browse to upload
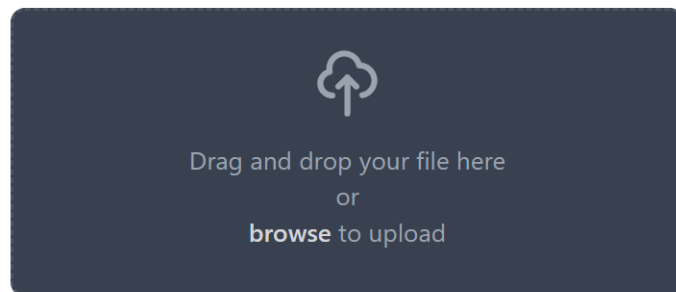
Upload

Now when you click Upload. It will take you to the Dashboard again.

View All Exams:

From the Dashboard select "View all Exam"

Here you will get a list of all the Exam folders. Let's click on A$

All Exams

A3	A4

Individual Exams:

We can now see the Exams content. The Report is automatically generated on the fly.

A4

Actions

Generate Report | Upload Solution | Upload an Image | Download CSV

Report
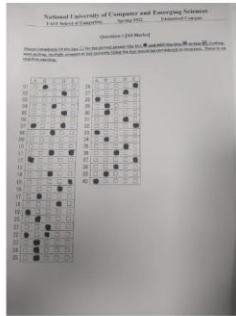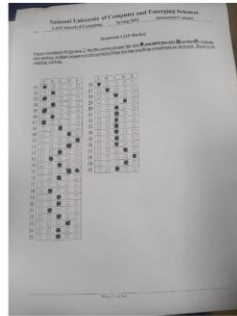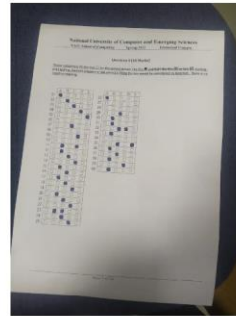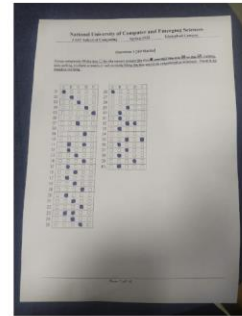

Score: 7
1.jpeg

Score: 7
2.jpeg

Score: 40
3.jpeg

Score: 40
4.jpeg

Individual Exams:

We can now perform the following actions

1. **Upload a New Solution:** Click on Upload Solution to provide a new Solution
2. **Generate Report:** If you uploaded a new solution, you can click generate report and it will generate an uploaded report
3. **Upload New Image to Score:** You can upload more images by clicking on the Upload an Image Button and Then generate a new Report to Score them.
4. **Download CSV:** You can download the CSV File of the report.