Project Name	Project - A Novel Method for Handwritten Digit Recognition System
Team ID	PNT2022TMID50703
Team Members	Gomathi N, Rajeshwari C, Sakthi Manisha M, Tamilselvi P

1. INTRODUCTION

Project Overview

Our project is “A Novel Method for Handwritten Digit Recognition System.” This is a three- step process and it has user friendly interface. The three-step process are

1. Login
2. Upload
3. Result

Login - This is the first page; in this page you have to enter your email id and password. If you entered the correct credentials you will redirect to the next page.

Upload – This is the second page; in this page you can upload the image in your local system. In this page you could not upload the files except the jpeg, png and jpg files. It will also provide the facility to preview the image that you have uploaded.

Result – This is the third page; in this page the predicted value will be shown in the graph format. You can also download the page.

Purpose

The human handwritten digits are not perfect and it can be made with different sizes and shapes. To overcome this problem, it is needed some system that is faster than humans. The attractive solution for this problem is “Handwritten Digit Recognition System.” It is difficult to identify someone’s handwritten digits to recognize. It will make people stressed. They could not complete their work on time. To reduce these complications, it will be useful. Through this people can easily upload their handwritten digit image and they can get the predicted value. This handwritten digit recognition system can be useful in business perspective as well. Industries and organization can use this system as their part of work. Banks, Postal service can use this to recognize the digit code written by peoples. Our model is going to deploy in a web. So anyone on the internet can access the service provide by the system.

2. LITERATURE SURVEY

Existing problem

Handwritten recognition system has problems when it comes to accuracy. The issue is that there is a wide range of handwritings good and bad. This makes it tricky for programmers to provide enough examples of how every character might look. Sometimes, characters look very similar, making it hard for a computer to recognize accurately. If the system does not provide accurate prediction means it makes confusion to the users. It takes more time to predict the value it makes people anxious. These are all the problem in existing system.

References

1. 1. Saqip Ali, Zeeshan Shaukat, Muhammad Azeem, Zareen Sakhawat, Tariq Mahmood and Khalil ur Rehman, "An efficient and improved scheme for handwritten digit recognition based on convolutional neural network", Springer Nature Applied Sciences, pp: 1-9, 2019.

2. Hui-huang Zhao and Han Liu, "Multiple classifiers fusion and CNN feature extraction for handwritten digits recognition", Granular Computing, pp: 411-418, 2019.

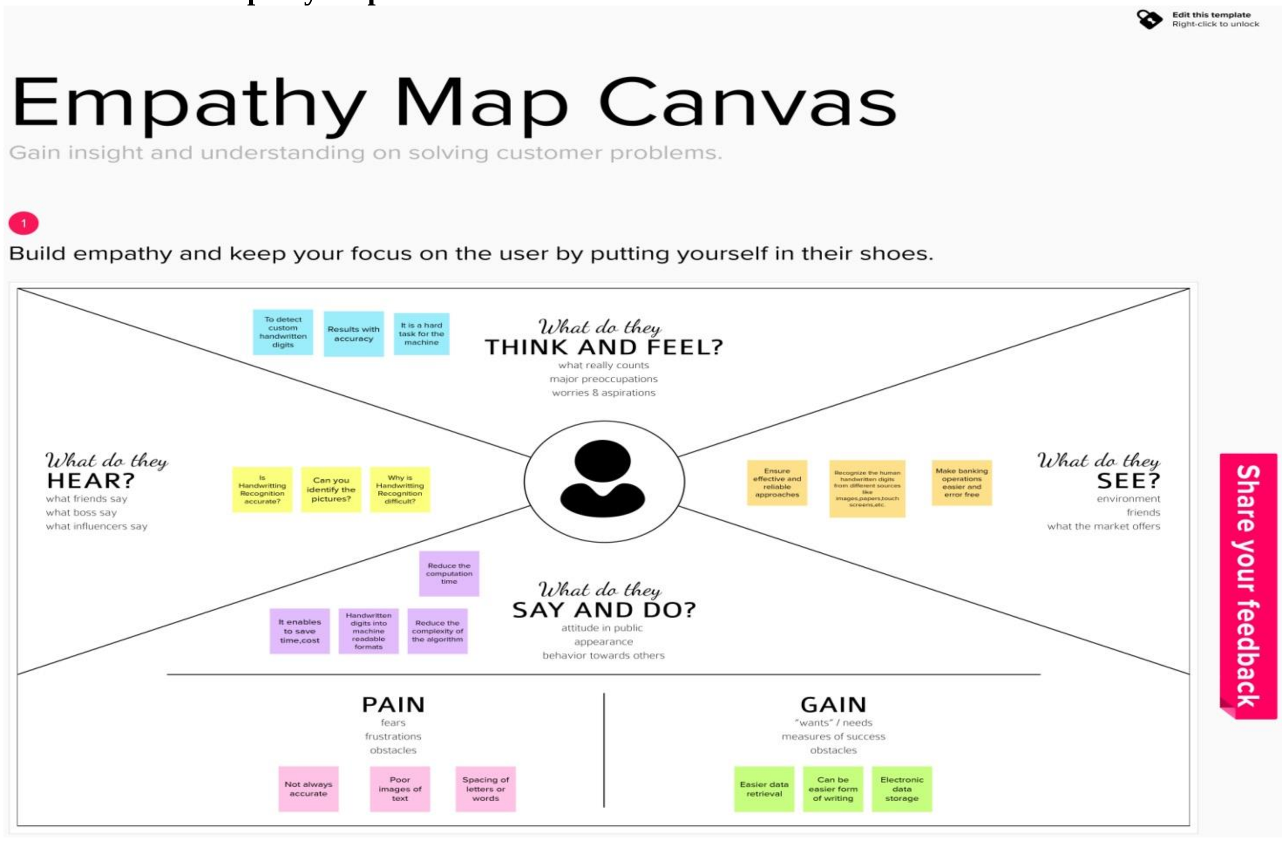
3. Ali Abdullah Yahya, Jieqing Tan and Min Hu, "A Novel Handwritten Digit Classification System Based on Convolutional Neural Network Approach", pp: 1-26, Sensors, 2021.

Problem statement definition

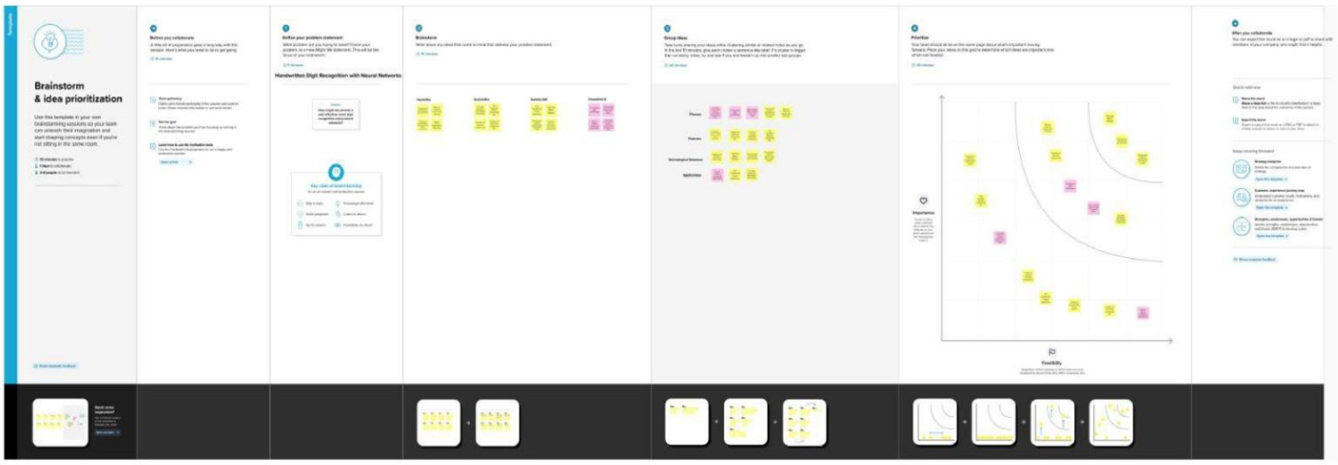
Everyone have different type of handwriting and it is also difficult to recognize the digit. The delay in recognition makes people anxious. It also makes delay in work completion. To reduce these types of problems we bring the solution that is digit recognizer.

3. IDEATION & PROPOSED SOLUTION

Empathy Map Canvas


Edit this template
Right-click to unlock
Empathy Map Canvas
Gain insight and understanding on solving customer problems.
1
Build empathy and keep your focus on the user by putting yourself in their shoes.
To detect custom handwritten digits
Results with accuracy
It is a hard task for the machine
What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
What do they HEAR?
what friends say
what boss say
what influencers say
Is Handwritting Recognition accurate?
Can you identify the pictures?
Why is Handwritting Recognition difficult?
Ensure effective and reliable approaches
Recognize the human handwritten digits from different sources like images,papers,touch screens,etc
Make banking operations easier and error free
What do they SEE?
environment
friends
what the market offers
Reduce the computation time
It enables to save time,cost
Handwritten digits into machine readable formats
Reduce the complexity of the algorithm
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
PAIN
fears
frustrations
obstacles
Not always accurate
Poor images of text
Spacing of letters or words
GAIN
"wants" / needs
measures of success
obstacles
Easier data retrieval
Can be easier form of writing
Electronic data storage
Share your feedback

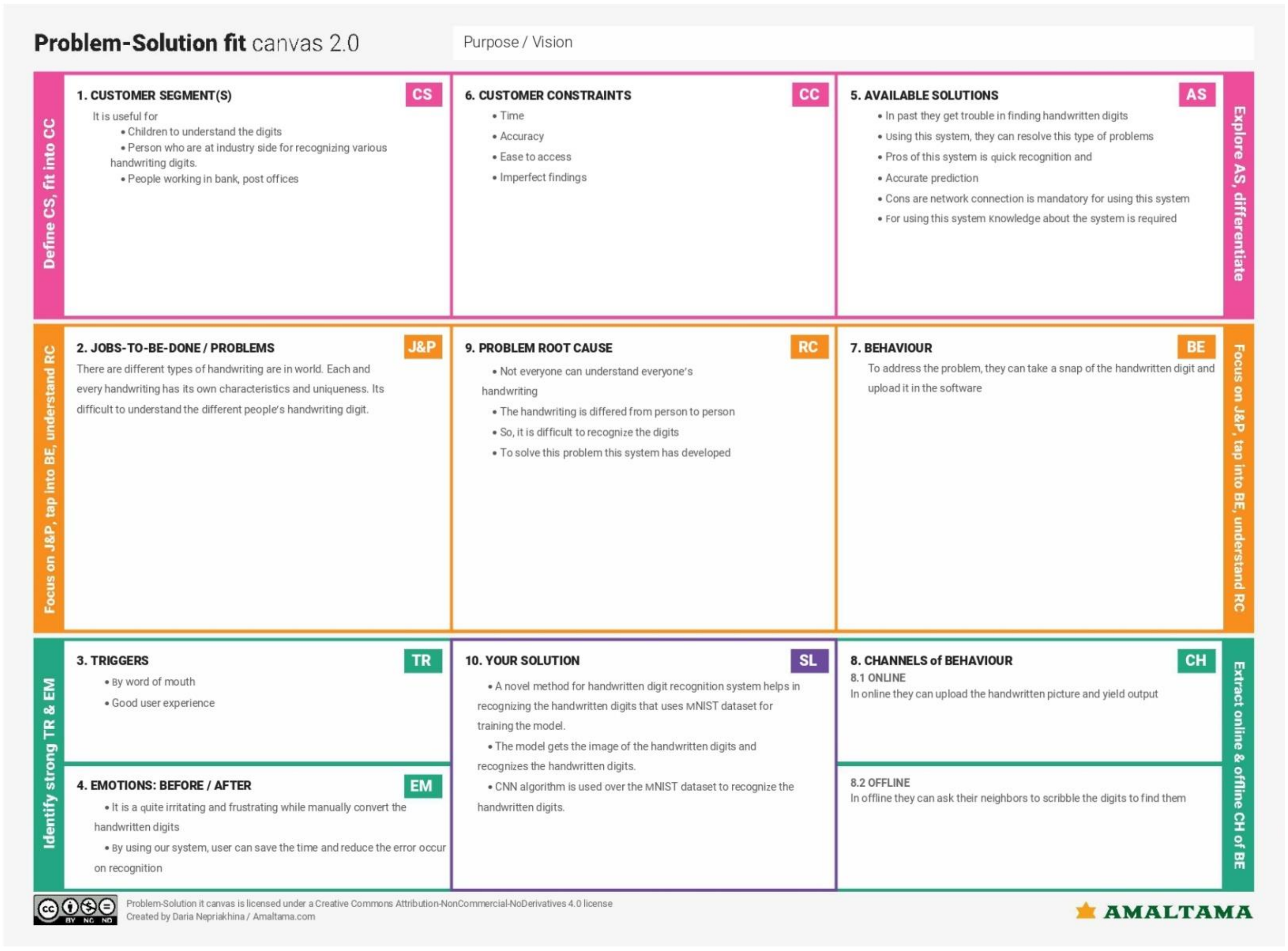
Ideation & Brainstorming


Template
Brainstorm & idea prioritization
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.
Before you collaborate
Define your problem statement
Handwritten Digit Recognition with Neural Networks
Key rules of brainstorming
Brainstorm
Group ideas
Prioritize
Importance
Feasibility
After you collaborate
Quick add-ons
Keep moving forward

Proposed Solution

S.No.	Parameters	Description
1.	Problem Statement (problem to be solved)	● It is very difficult to recognize the handwritten digits because every individual in this world has their own style of writing. ● Recognition accuracy and computation time still require further improvement.
2.	Idea / Solution Description	● Handwritten digit recognized system to increase accuracy with minimum computation time.
3.	Novelty / Uniqueness	To develop effective and reliable approach for recognition of handwritten digits using Deep Learning based Convolutional Neural Network (D-CNN).
4.	Social Impact / Customer satisfaction	● Handwritten digit recognition is one of the practically important issues in pattern recognition applications. ● This system is used for everything is being digitalized to reduce human effort.
5.	Business Model (Revenue Model)	● We are proposing to develop an automatic banking deposit number recognition system which is able to recognize the handwritten account number and amount number on the case deposit slip.
6.	Scalability of the solution	● Handwritten recognition is one of the most challenging areas of the pattern recognition. ● My project is effective until put a even better solution depends on accuracy and computation time.

Problem Solution fit



Problem-Solution fit canvas 2.0

Purpose / Vision

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

It is useful for
- Children to understand the digits
- Person who are at industry side for recognizing various handwriting digits.
- People working in bank, post offices

6. CUSTOMER CONSTRAINTS CC

- Time
- Accuracy
- Ease to access
- Imperfect findings

5. AVAILABLE SOLUTIONS AS

- In past they get trouble in finding handwritten digits
- Using this system, they can resolve this type of problems
- Pros of this system is quick recognition and
- Accurate prediction
- Cons are network connection is mandatory for using this system
- For using this system knowledge about the system is required

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

There are different types of handwriting are in world. Each and every handwriting has its own characteristics and uniqueness. Its difficult to understand the different people's handwriting digit.

9. PROBLEM ROOT CAUSE RC

- Not everyone can understand everyone's handwriting
- The handwriting is differed from person to person
- So, it is difficult to recognize the digits
- To solve this problem this system has developed

7. BEHAVIOUR BE

To address the problem, they can take a snap of the handwritten digit and upload it in the software

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

- By word of mouth
- Good user experience

4. EMOTIONS: BEFORE / AFTER EM

- It is a quite irritating and frustrating while manually convert the handwritten digits
- By using our system, user can save the time and reduce the error occur on recognition

10. YOUR SOLUTION SL

- A novel method for handwritten digit recognition system helps in recognizing the handwritten digits that uses MNIST dataset for training the model.
- The model gets the image of the handwritten digits and recognizes the handwritten digits.
- CNN algorithm is used over the MNIST dataset to recognize the handwritten digits.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE
In online they can upload the handwritten picture and yield output

8.2 OFFLINE
In offline they can ask their neighbors to scribble the digits to find them

Extract online & offline CH of BE

AMALTAMA

4. REQUIREMENT ANALYSIS

Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
1	FR-1	Image Data: Handwritten digit recognition refers to a computer's capacity to identify human handwritten digits from a variety of sources, such as photographs, documents, touch screens, etc., and categorize them into ten established classifications (0-9). In the realm of deep learning, this has been the subject of countless studies.
2	FR-2	Digit Classifier Model: To train a convolutional network to predict the digit from an image, use the MNIST database of handwritten digits. get the training and validation data first.
3	FR-3	Modified National Institute of Standards and Technology dataset: The abbreviation MNIST stands for the MNIST dataset. It is a collection of 60,000 tiny square grayscale photographs, each measuring 28 by 28, comprising handwritten single digits between 0 and 9.

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	One of the very significant problems in pattern recognition applications is the recognition of handwritten characters. Applications for digit recognition include filling out forms, processing bank checks, and sorting mail.
NFR-2	Security	1) The system generates a thorough description of the instantiation parameters, which might reveal information like the writing style,in addition to a categorization of the digit. 2) The generative models are capable of segmentation driven by recognition. 3) The procedure uses a relatively.
NFR-3	Reliability	The samples are used by the neural network to automatically deduce rules for reading handwritten digits. Furthermore, the network may learn more about handwriting and hence enhance its accuracy by increasing the quantity of training instances. Numerous techniques and algorithms, such as Deep Learning/CNN, SVM, Gaussian Naive Bayes, KNN, Decision Trees, Random Forests, etc., can be used to

		recognize handwritten numbers.
NFR-4	**Performance**	With typed text in high-quality photos, optical character recognition (OCR) technology offers accuracy rates of greater than 99%. However, variances in spacing, abnormalities in handwriting, and the variety of human writing styles result in less precise character identification.
NFR-5	**Availability**	To develop an automatic banking deposit number recognition system which is able to recognize the handwritten account number and amount number on the cash deposit slip.

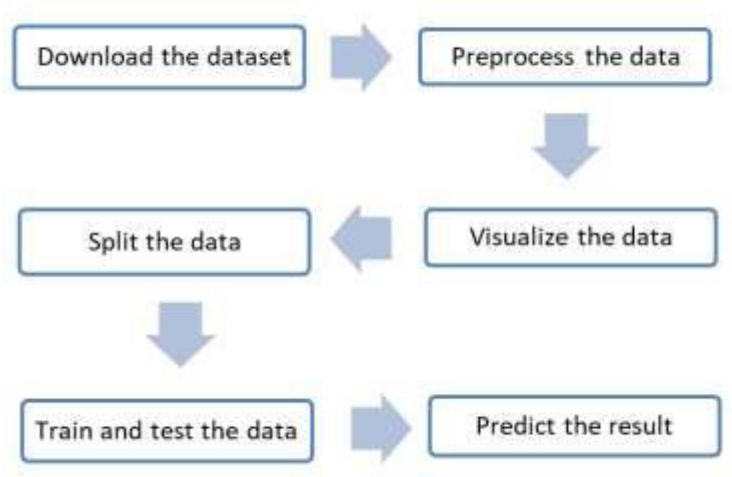
5. PROJECT DESIGN

Data Flow Diagrams

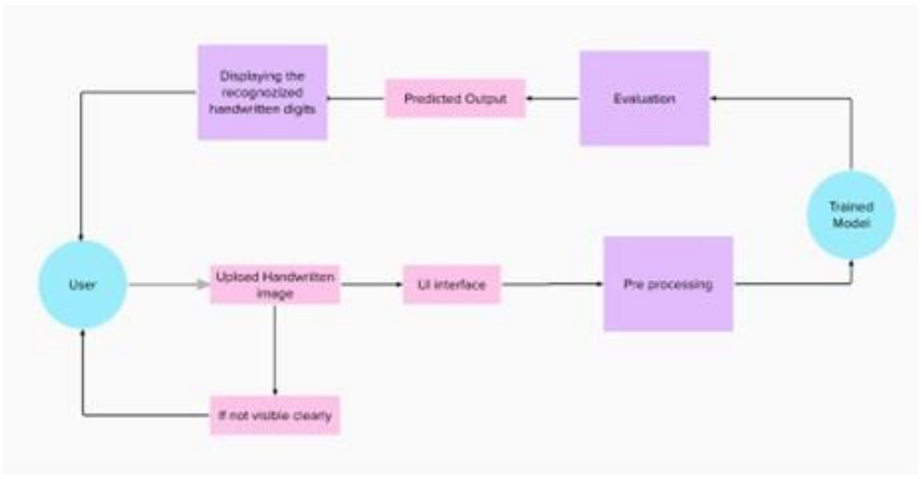
Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

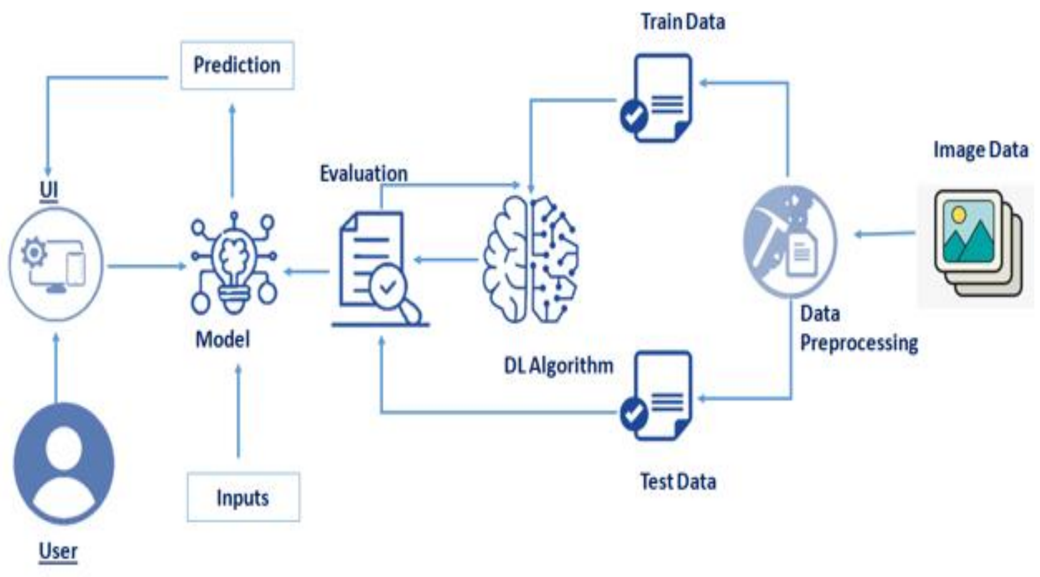
Flow Diagram:



DFD :



Solution & Technical Architecture



User stories

Use the below template to list all the user stories for the product

User Type	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Acceptance criteria**	**Priority**	**Release**
Customer (Mobile user)	Details	USN-1	As a user, I can know the deta fundamental usage of digit rec	I can access information of details page	Low	Sprint-2
	Image upload	USN-2	As a user, I will upload the handwritten digit image to the digit recognition system.	I can upload the image from the local system	High	Sprint-2
	Recog nized result	USN-3	As a user, I can see the As a user, I can see the predic digits in the digit recogni	I can see the output of the recognized digit	High	Sprint-3
Customer (Web user)	Details	USN-1	As a user, I can know the detai details of the fundamental usage of di recognition system.	I can access information of details page	Low	Sprint-2
	Image upload	USN-2	As a user, I will upload the image to the digit recognition s	I can upload the image from the local system	High	Sprint-2
	Recog nized result	USN-3	As a user, I will upload the ha digit image to the application b upload button.	I can see the output of the recognized digit	High	Sprint-3

6. PROJECT PLANNING & SCHEDULING

Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Pre processing	USN-1	As a user, I can upload any kind of image with the pre-processing step is involved in it.	3	High	Gomathi, Rajeshwari, Sakthi manisha ,Tamil selvi
Sprint-1		USN-2	As a user, I can upload the image in any resolution.	1	Low	Gomathi, Rajeshwari, Sakthi manisha, Tamil selvi
Sprint-2	Model	USN-3	As a user, I will get a application with ML model which provides high accuracy of recognized handwritten digit.	2	Medium	Gomathi ,Rajeshwari, Sakthi manisha, Tamilselvi
Sprint-2		USN-4	As a user, I can pass the handwritten digit image for recognizing the digit.	2	Medium	Gomathi, Rajeshwari, Sakthi manisha, Tamil selvi
Sprint-2		USN-5	As a user, I can get the most suitable recognized digit.	3	High	Gomathi, Rajeshwari, Sakthi manisha, Tamil selvi
Sprint-3	User Interface	USN-6	As a user, I can login and I will upload the handwritten digit image to the application by clicking a upload button.	5	High	Gomathi, Rajeshwari, Sakthi

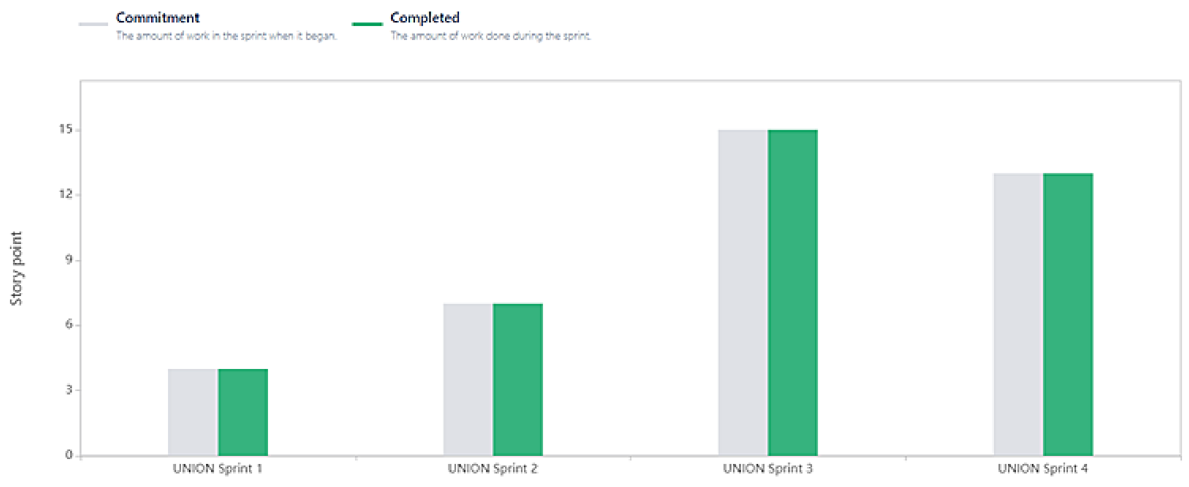
						manisha , Tamil selvi
Sprint-3		USN-7	As a user, I can know the details of the fundamental usage of the application.	2	Low	Gomathi, Rajeshwari, Sakthi manisha, Tamil selvi
Sprint-3		USN-8	As a user, I can see the predicted / recognized digits in the application	8	High	Gomathi, Rajeshwari, Sakthi manisha, Tamil selvi
Sprint-4	Cloud Deployment	USN-9	As a user, I can access the web application and make the use of the product from anywhere	13	High	Gomathi, Rajeshwari, Sakthi manisha, Tamil selvi

Sprint Delivery Schedule

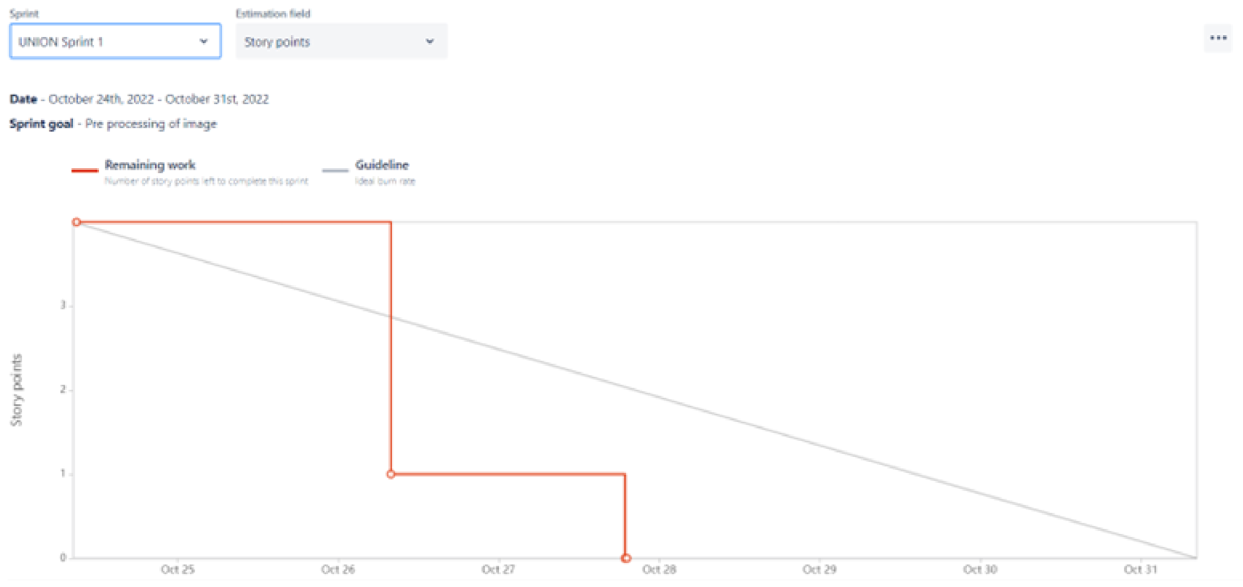
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	4	6 Days	24 Oct 2022	29 Oct 2022	4	29 Oct 2022
Sprint-2	7	6 Days	31 Oct 2022	05 Nov 2022	7	05 Nov 2022
Sprint-3	15	6 Days	07 Nov 2022	12 Nov 2022	15	12 Nov 2022
Sprint-4	13	6 Days	14 Nov 2022	19 Nov 2022	13	19 Nov 2022

Reports from JIRA

Velocity Report



Sprint 1 – Burndown Chart



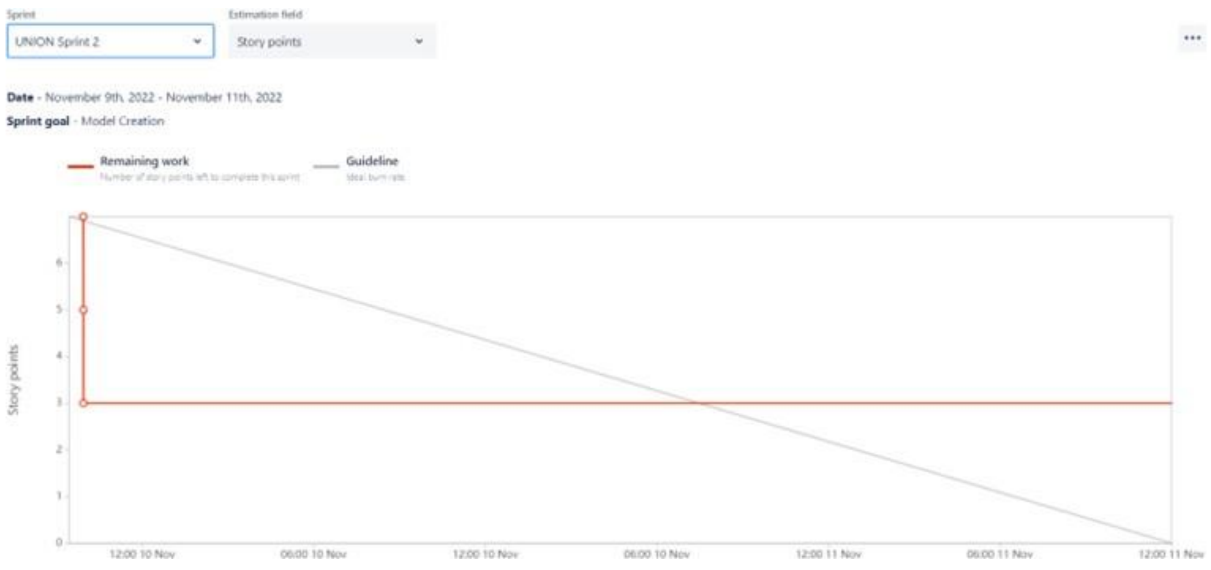
Sprint: UNION Sprint 1

Estimation field: Story points

Date - October 24th, 2022 - October 31st, 2022

Sprint goal - Pre processing of image

Sprint 2 – Burndown Chart



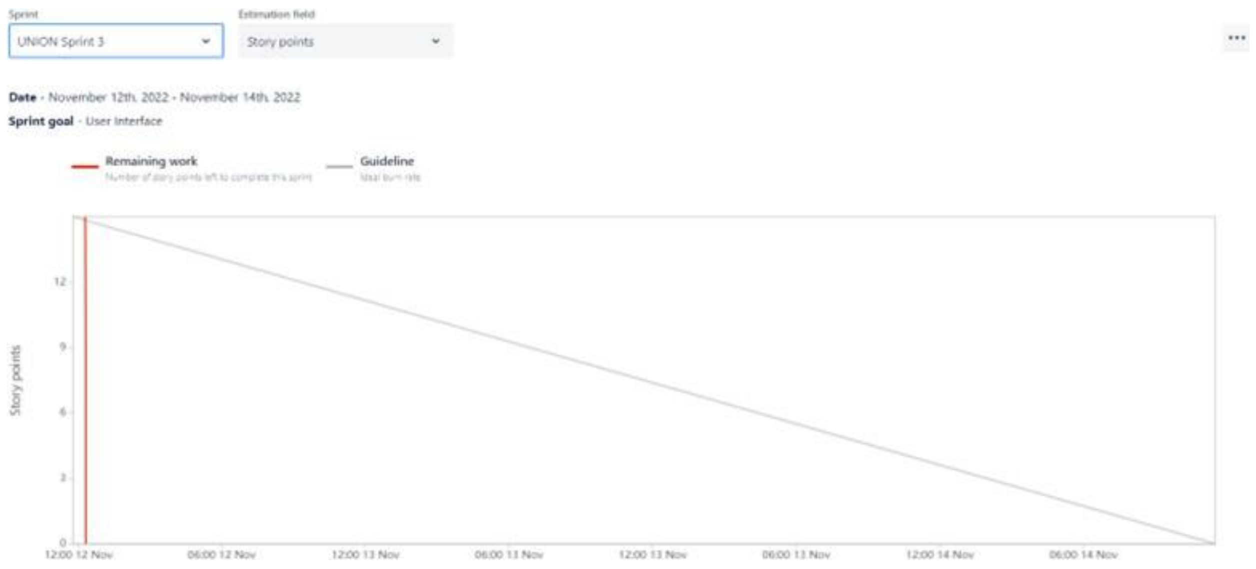
Sprint: UNION Sprint 2

Estimation field: Story points

Date - November 9th, 2022 - November 11th, 2022

Sprint goal - Model Creation

Sprint 3 – Burndown Chart

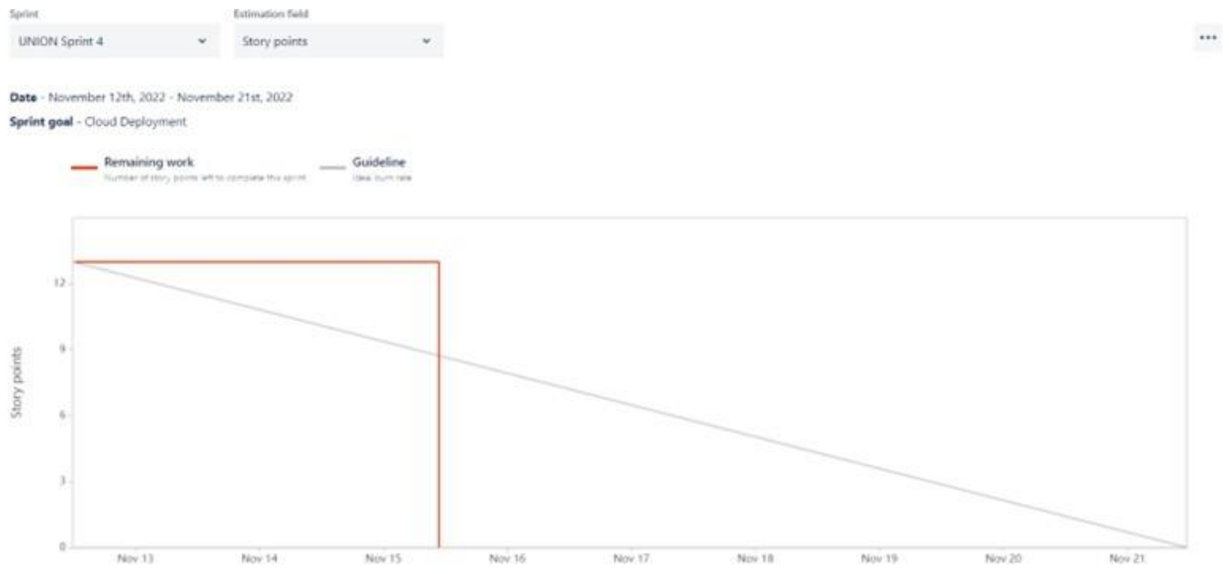

Sprint: UNION Sprint 3

Estimation field: Story points

Date - November 12th, 2022 - November 14th, 2022

Sprint goal - User Interface

Sprint 4 – Burndown Chart


Sprint: UNION Sprint 4

Estimation field: Story points

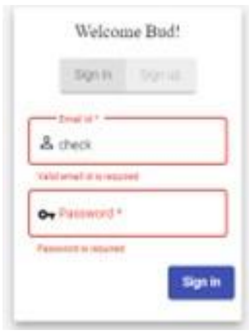
Date - November 12th, 2022 - November 21st, 2022

Sprint goal - Cloud Deployment

7. CODING & SOLUTIONING

Features

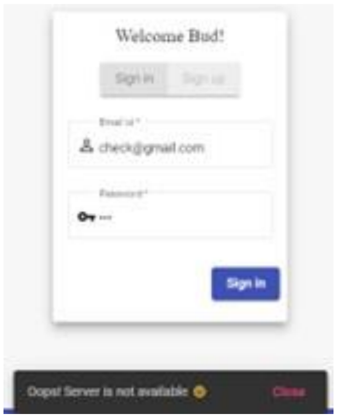
- Basic validations are verified, when user enters a credentials.



Valid email id is required

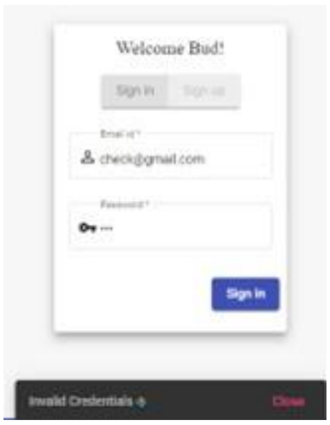
Password is required

- If the server is not available, we should restrict the user without redirecting to the next page and let them wait in the same login page with an indication message “Oops server is not available”.



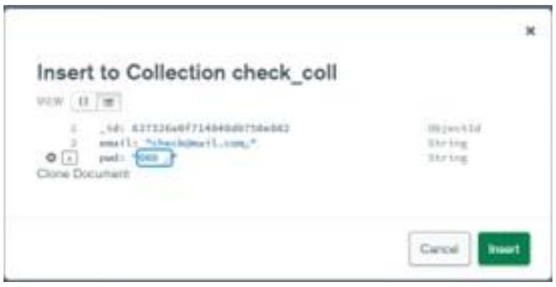
Oops! Server is not available

- When the user enters the invalid credential the snackbar will appears at the bottom and shows the corresponding message.



Invalid Credentials

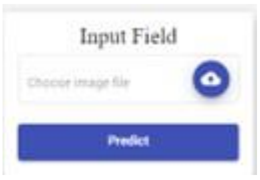
- If the user is available in mongodb collections, he/she is redirected to the next page. If the new user credentials are to be entered, it will done by creating a new document with necessary details in the mongodb.



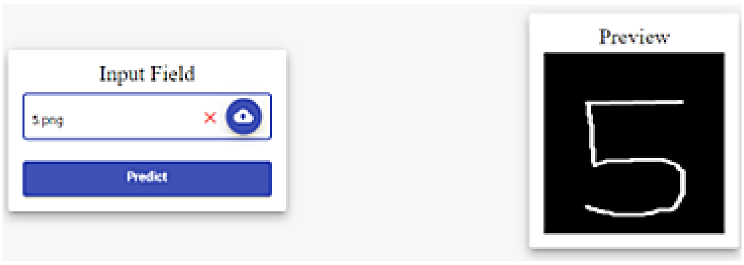
Insert to Collection check_coll

```
_id: [illegible]        ObjectId
email: "check@mail.com,"     String
pwd: "[illegible]"            String
```

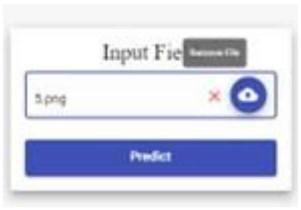
- When correct email is entered, user should redirect to the upload page by showing the indicating message that user is verified.



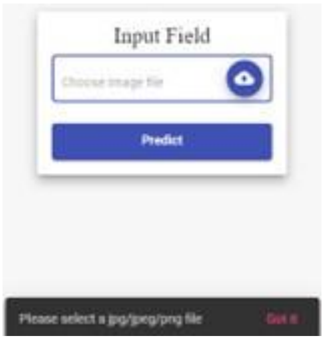
- Preview is useful in most of the places, here also we implement the preview card. When the user upload the pic, it stores in the database and show the preview immediately.



- Tip tool is useful to know about what we are doing. There is remove file button, to remove the file in form data. Immediately after removing the file, the preview card isdisappeared to maintain the flow.

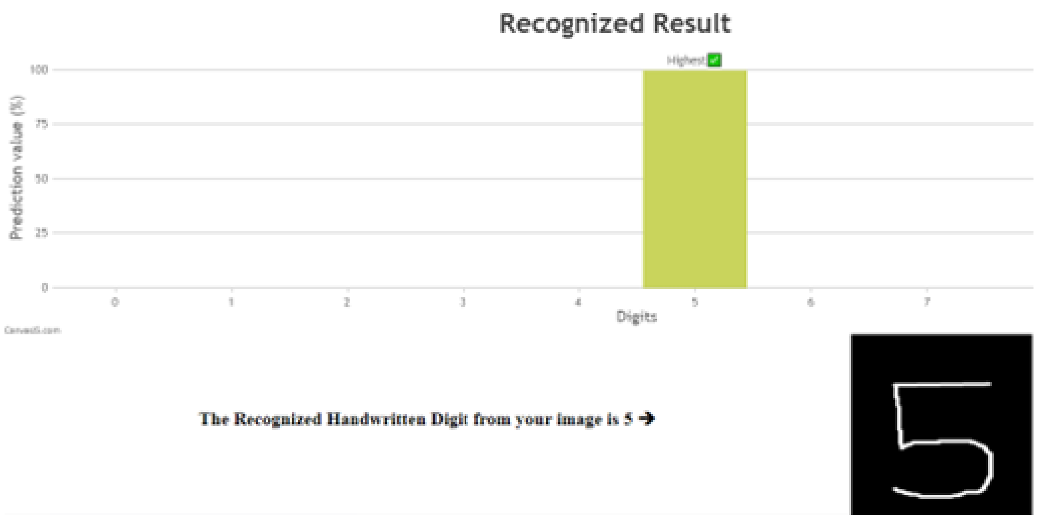


- Uploading the files is restricted only to image files. In special case, user enters a other format file, it will collapse the system. For overcoming that, new snackbar is created and shows the corresponding message if user enters the other format files.



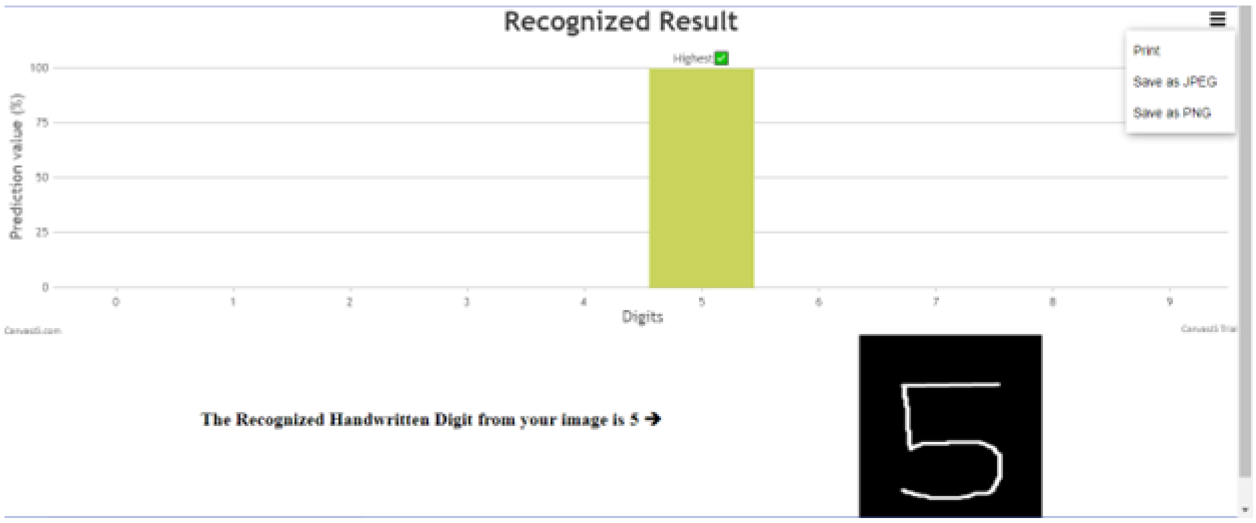
Please select a jpg/jpeg/png file

- Main feature of our project is showing the predicted results as a column chart, it makes the user to easily understand the results. Chart is created by canvas js library.



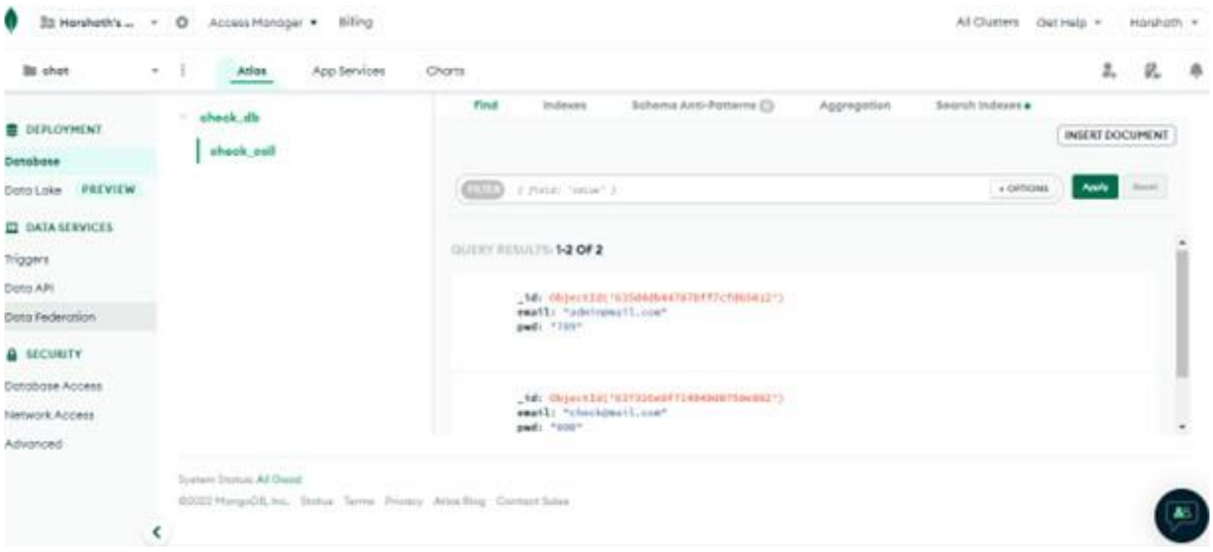
The Recognized Handwritten Digit from your image is 5 ➔

- Chart can be downloaded if it is needed.



The Recognized Handwritten Digit from your image is 5 ➔

Database Schema



QUERY RESULTS: 1-2 OF 2

```
_id: ObjectId('[illegible]')
email: "admin@mail.com"
pwd: "789"

_id: ObjectId('[illegible]')
email: "check@mail.com"
pwd: "000"
```

8. User Acceptance Testing

1.Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	17	3	4	3	27
Duplicate	1	3	2	1	7
External	2	5	3	1	11
Fixed	9	4	7	28	48
Not Reproduced	0	0	1	0	1
Skipped	1	0	1	1	3
Won't Fix	0	0	0	1	1
Totals	29	15	18	35	98

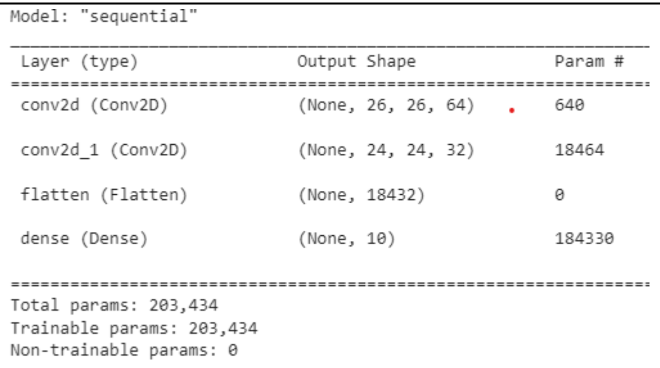
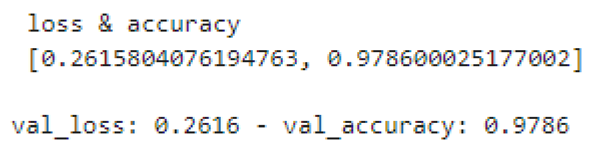
2.Test Case Analysis

This reportshows the numberof test cases that have passed, failed,and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	33	0	0	33
Security	10	0	0	10
Outsource Shipping	6	0	0	6
Exception Reporting	17	0	0	17
Final Report Output	8	0	0	8
Version Control	5	0	0	5

9.RESULTS

Performance Metrics

S.No.	Parameter	Values	Screenshot
1.	Model Summary	Layers- Conv2d (Conv2D) Conv2d_1 (Conv2D) Flatten (Flatten) Dense (Dense)	
2.	Accuracy	Training Accuracy – 0.978 Validation Accuracy - 0.9786	

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 conv2d_1 (Conv2D)           (None, 24, 24, 32)        18464

 flatten (Flatten)           (None, 18432)             0

 dense (Dense)               (None, 10)                184330

=================================================================
Total params: 203,434
Trainable params: 203,434
Non-trainable params: 0
```

```
 loss & accuracy
 [0.2615804076194763, 0.978600025177002]

val_loss: 0.2616 - val_accuracy: 0.9786
```

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES

1. User friendly interface makes the user to navigate easily to other pages.
2. It provides high accuracy.
3. The quick prediction will save the time of the users.
4. It will show the preview of the image which the user uploaded. It will help the user to check whether he/she uploaded the correct image.
5. It provides results in graphical representation for easy understanding.
6. Handwritten Digit Recognizer is an angular js application and it is also deployed in github pages for easy access.
7. Login credentials are not static it will be fetched from mongodb atlas collections.
8. Users can download the prediction result as chart.

DISADVANTAGES

1. This is only for single digit recognition.

2. The persons who have the knowledge about this only can use this.

3. For now, the flask API is only run in the local host. So it is only used in offline.

11. CONCLUSION

An implementation of handwritten recognition using deep learning has been implemented. In this handwritten recognition system high accuracy is achieved. We have used the Machine Learning algorithm CNN for accuracy. Here mongodb is used to store the information like email id and password. During login, verification will be done by fetching the information which is stored in mongodb. It has many features and one of the best features is it will show the preview of the image after it is uploaded and showing the predicted results in graphical representation. Preview helps the user to check whether the correct image is uploaded. Column chart (Canvas js) makes the result page more attractive. The accuracy rate of this handwritten recognizer is 97.86%.

12. FUTURE SCOPE

Artificial Intelligence have more scope in these days. It plays a vital role in every places such as schools, colleges, offices, etc. Like that the Handwritten Recognition system will be more helpful in many fields. In post office it is used to recognize the digits of the postal codes. In medical coding it will be more useful to recognize the digits. The task of handwritten digit recognition, using a classifier, has great importance and use such as online handwriting recognition on computer tablets, recognize zip codes on mail for postal mail sorting, processing bank cheque amounts, numeric entries in forms filled up by hand and so on.

13. APPENDIX

Source Code

DigitAPI.py

```python
from flask import Flask,Blueprint
from flask_cors import CORS
from flask_pymongo import PyMongo
from endpoints import api endpoints

def create_app():
    webapp = Flask(__name__)
    CORS(webapp)

    api_blueprint = Blueprint('api_blueprint',__name__)
    api_blueprint = api_endpoints(api_blueprint)
    webapp.register_blueprint(api_blueprint, url_prefix= '/api')
    return webapp

app=create_app()
if('__main__'==__name__):
    app.run(host='0.0.0.0')
```

endpoints.py

```python
from flask_pymongo import pymongo
from flask import request,send_file
from keras.models import load_model
from PIL import Image
import numpy as np

model = load_model("digit-recognition.h5")
uri = 'mongodb+srv://harsh:harsh@cluster0.rxvjk.mongodb.net/?retryWrites=true&w=majority'
client = pymongo.MongoClient(uri)
db = client.check_db
coll = db.check_coll
print('connection has made')

def api_endpoints(endpoints):
    @endpoints.route('/verify', methods=['POST'])
```

```
def verify():
    try:
        email = request.form.get('email')
        pwd = request.form.get("pwd")
        flag = coll.find_one({"email":email, "pwd":pwd})
        status={
        'statuscode' : 200,
        }
        if(flag!=None):
            status['statusmessage'] = "true"
        else:
            status['statusmessage'] = "false"
    except Exception as e:
        status={
            'statuscode' : 400,
        'statusmessage' : str(e)
        }
    return status

@endpoints.route('/upload', methods=['POST'])
def upload():
    input = request.files.get("image")
    global format
    format = request.form.get("format")
    img= Image.open(input)
    img = img.resize((200,200))
    img.save("files/input."+format)
    return send_file(path_or_file = "files/input."+format)

@endpoints.route('/predict', methods=['GET'])
def predict():
    result = {};
    img=Image.open("files/input."+format).convert("L")
    img = img.resize((28,28))
    im2arr=np.array(img)
    im2arr = im2arr.reshape(1,28,28,1)
    y_pred = model.predict(im2arr)
    result["value"] = int(np.argmax(y_pred))
    print("Predicted value is",result)
    return result

@endpoints.route('/image', methods=['GET'])
def image():
    return send_file(path_or_file = "files/input."+format)
return endpoints
```

<u>Digit Recognizer (AngularJS files) :</u>

Login Component

login.component.html

```
<div class="entire-login">
    <mat-card class="mat-elevation-z8">
        <mat-card-header class="flex-center">
            <mat-card-title >
                Welcome Bud!
            </mat-card-title>
        </mat-card-header>
        <mat-card-actions class="flex-center">
                    <mat-button-toggle-group style="margin: auto;" appearance="legacy">
                                <mat-button-toggle  value="sign_in" checked="true">Sign in</mat-button-toggle>
                                <mat-button-toggle  value="sign_up" disabled="true">Sign up</mat-button-toggle>
            </mat-button-toggle-group>
        </mat-card-actions>
        <mat-card-content class="card-content">
            <mat-form-field appearance="outline">
                <mat-label>Email id</mat-label>
                <mat-icon matPrefix>perm_identity</mat-icon>
                        <input  id="email"  name="email"  matInput type='email' [formControl]="emailFC" placeholder="" #email/>
                <mat-error *ngIf="emailFC.hasError('required')">
                    Email id is required
                </mat-error>
                  <mat-error *ngIf="emailFC.hasError('email') && !emailFC.hasError('required')">
                    Valid email id is required
                </mat-error>
            </mat-form-field>
            <mat-form-field appearance="outline">
                <mat-label >Password</mat-label>
```

```html
                <mat-icon matPrefix> vpn_key</mat-icon>
                   <input id="password" name="password" matInput type="password" [formControl]='passwordFC' #password>
                <mat-error *ngIf="passwordFC.hasError('required')">
                    Password is required
                </mat-error>
            </mat-form-field>
            <mat-card-actions align="end">
                    <button  mat-raised-button  color="primary" (click)="val_credentials(email.value,password.value)">Sign in</button>
            </mat-card-actions>
        </mat-card-content>
    </mat-card>
</div>
```

login.component.css

```css
.entire-login{
    display: flex;
    justify-content: center;
    align-items: center;
    height:85vh;
    background: #f7f7f7;
}

.card-content{
    display: flex;
    flex-direction: column;
}
.flex-center{
    display: flex;
    justify-content: center;
}
.mat-card{
    /* background-color: aliceblue; */
    box-shadow: 50px;
    font-family: 'Times New Roman', Times, serif;
}
```

login.component.spec.ts

```
// Done by Harshath.M

import      {      ComponentFixture,      TestBed      }      from
'@angular/core/testing';

import { LoginComponent } from './login.component';

describe('LoginComponent', () => {
  let component: LoginComponent;
  let fixture: ComponentFixture<LoginComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ LoginComponent ]
    })
    .compileComponents();

    fixture = TestBed.createComponent(LoginComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

login.component.ts

```
// Done by Harshath.M

import { HttpClient } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { FormControl, Validators } from '@angular/forms';
import { MatSnackBar } from '@angular/material/snack-bar';
import { Router } from '@angular/router';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
```

```
export class LoginComponent implements OnInit {
  email = "";
  pwd = "";
  invalid= true;
  showbutton = true;

    constructor(private  route:Router,  private  http:HttpClient, private snackbar:MatSnackBar) {

  }

  ngOnInit(): void {}
  emailFC = new FormControl('',[Validators.email,Validators.required]);
  passwordFC = new FormControl('',[Validators.required]);

  val_credentials(email:string,pwd:string){
    let formdata = new FormData();
    formdata.append("email",email);
    formdata.append("pwd",pwd);
    let api_url = "http://127.0.0.1:5000/api/";
    this.http.post(api_url+"verify",formdata).subscribe({
      next:((res:any)=>{
        if(res.statusmessage=='true'){
          this.snackbar.open("Email  and  Password  is verified□","Welcome", {duration:2000});
          this.route.navigate(['/upload']);
        }
        else if(res.statusmessage=="false"){
          this.snackbar.open("Invalid Credentials ♂","Close", {duration:4000});
        }
        else{
          this.snackbar.open("Oops! Something went wrong□","Close", {duration:4000});
        }
      }),
      error:(()=>{
        this.showbutton=true;
        this.snackbar.open("Oops!  Server  is  not  available □","Close", {duration:4000});
      })
    });
  }
}
```

Page not found Component

page-not-found.component.html

```html
<div>
    <p>Page not found!!!</p>
    <p>Please enter the correct URL...</p>
</div>
```

page-not-found.component.css

```css
div{
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    height: 80%;
}
```

page-not-found.component.spec.ts

```typescript
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { PageNotFoundComponent } from './page-not-found.component';
describe('PageNotFoundComponent', () => {
  let component: PageNotFoundComponent;
  let fixture: ComponentFixture<PageNotFoundComponent>;
  beforeEach(async () => {
    await TestBed.configureTestingModule({
    declarations: [ PageNotFoundComponent ]
    })
    .compileComponents();

    fixture = TestBed.createComponent(PageNotFoundComponent);
    component = fixture.componentInstance;
  fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

page-not-found.component.ts

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-page-not-found',
  templateUrl: './page-not-found.component.html',
  styleUrls: ['./page-not-found.component.css']
})
export class PageNotFoundComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

Result Component

result.component.html

```
<div class="middle" *ngIf="!load_graph">
    <mat-progress-spinner mode="indeterminate"  diameter="60" strokeWidth="7" ></mat-progress-spinner>
</div>
<canvasjs-chart class="chart" *ngIf="load_graph" [options]="chartOptions" [styles]="{width: '100%', height:'360px'}"></canvasjs-chart>

<div *ngIf="load_graph" class="description">
  <h2>The Recognized Handwritten Digit from your image is {{pred_value}} □</h2>
  <img [src]="preview">
</div>
```

result.component.css

```
.middle{
    display: flex;
    justify-content: center;
    align-items: center;
    height:80%;
}
.chart{
    margin-top:20px;
}
.metrics-table{
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
    margin: 50px 0;
}
table{
    min-width: 350px;
    margin-bottom: 20px;
}
.description{
  display: flex;
  justify-content: space-evenly;
  flex-wrap: wrap;
  align-items: center;
}
.description h2{
  font-family: 'Times New Roman', Times, serif;
  font-weight: bold;
}
```

result.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { ResultComponent } from './result.component';

describe('ResultComponent', () => {
  let component: ResultComponent;
  let fixture: ComponentFixture<ResultComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ ResultComponent ]
    })
    .compileComponents();

    fixture = TestBed.createComponent(ResultComponent);
    component = fixture.componentInstance;
  fixture.detectChanges();18
  });

  it('should create', () => {
  expect(component).toBeTruthy();22
  });
});
```

result.component.ts

```
import { HttpClient } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { DomSanitizer } from '@angular/platform-browser';
import { saveAs } from 'file-saver';


// Done by Harshath.M

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

  constructor(private http:HttpClient,
              private domsanitizer:DomSanitizer) { }
   chartOptions:any;
  pred_value = 0 ;
  load_graph = false;
  preview: any;
  api_url = "http://127.0.0.1:5000/api/";

  ngOnInit(): void {

this.http.get(this.api_url+'image',{responseType:'blob'}).subscr
ibe({
      next:((res:any)=>{
        let objecturl = URL.createObjectURL(res);
        this.preview =
this.domsanitizer.bypassSecurityTrustUrl(objecturl);
      })
    });

this.http.get(this.api_url+"predict").subscribe((res:any)=>{
      this.pred_value = res.value;
              this.open_page();
        this.load_graph = true;
        });
  }
  getDataPoints() {
    let dataPoints =[];
    for (var i = 0; i <= 9 ; i++)
      dataPoints.push({
        x: i,
        y: 0
      });
    dataPoints[this.pred_value]= { x : this.pred_value ,y:100,
indexLabel: "Highest\u2705"};
    console.log(dataPoints);
   return dataPoints;
   }

   open_page(){
        this.chartOptions = {
//https://canvasjs.com/angular-charts/chart-index-data-label/
      animationEnabled: true,
              exportEnabled: true,
              theme: "light2",
              title: {
              text: "Recognized Result"
 },
      axisX: {
        title: "Digits",
      interval: 1
      },
      axisY:{
        title: "Prediction value (%)",
        maximum: 110,
        interval:25
      },
             data: [{
                   type: "column",
            dataPoints: this.getDataPoints()
              }]
        }
   }
}
```

Upload Component

upload.component.html

```html
<div class="entire">
    <mat-card class="card mat-elevation-z8">
        <mat-card-title style="text-align: center;">Input Field</mat-card-title>
        <mat-form-field appearance="outline">
            <input hidden type='file' accept="image/*" #fileclick (change)="select_file($event)" >
            <input readonly matInput value="{{this.fname}}" placeholder="Choose image file" >
            <button *ngIf="this.file" matSuffix (click)="deletefile()" matTooltip="Remove File" matTooltipPosition = "above" color="warn" mat-icon-button>
                <mat-icon>close</mat-icon>
            </button>
            <button matSuffix mat-mini-fab color="primary" (click)="fileclick.click()" matTooltip="Select a file" matTooltipPosition="right">
                <mat-icon>backup</mat-icon>
            </button>
        </mat-form-field>
        <button (click)="predict()" mat-raised-button color="primary" style="min-height: 40px;">
            <span>Predict</span>
        </button>
    </mat-card>

    <mat-card class="card mat-elevation-z8" *ngIf="enable_preview">
        <mat-card-title style="text-align: center;">Preview</mat-card-title>
        <img [src]="preview">
    </mat-card>
</div>
```

upload.component.css

```

.entire{
    display: flex;
    justify-content: space-evenly;
    flex-wrap: wrap;
    align-items: center;
    height:85vh;
    background: #f7f7f7;
}
.card{
    display: flex;
    flex-direction: column;
    justify-content: center;
}
.mat-card{
    box-shadow: 50px;
    font-family: 'Times New Roman', Times, serif;
}
```

upload.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { UploadComponent } from './upload.component';

describe('UploadComponent', () => {
  let component: UploadComponent;
  let fixture: ComponentFixture<UploadComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ UploadComponent ]
    })
    .compileComponentseateComponent(UploadComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
```

upload.component.ts

```
import { HttpClient } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { MatSnackBar } from '@angular/material/snack-bar';
import { Router } from '@angular/router';
import { DomSanitizer } from '@angular/platform-browser';

// Done by Harshath.M

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

  all_formats=['png', 'jpg', 'jpeg']
  file :any;
  fname ='';
  fformat='';
  formdata:any;
  enable_preview =false;
  preview : any;

  constructor(private snackbar:MatSnackBar,
              private http:HttpClient,
              private route:Router,
              private domsanitizer:DomSanitizer) { }

  ngOnInit(): void {
  }

  select_file(event : any){
    try{
      this.file = event.target.files[0];
      if(this.file){
        this.fname = this.file.name;
        this.fformat = this.file.type.split('/')[1];
        if(this.all_formats.indexOf(this.fformat)!=-1){
          this.formdata= new FormData();
          this.formdata.append('image', this.file);
          this.formdata.append("format", this.fformat);

          let api_url = "http://127.0.0.1:5000/api/upload";
          this.http.post(api_url,this.formdata,{responseType:'blob'}).subscribe({
            next:((res:any)=>{
              let objecturl = URL.createObjectURL(res);
              this.preview = this.domsanitizer.bypassSecurityTrustUrl(objecturl);
            }),
            error:(()=>{
              this.snackbar.open("Oops! Server is not available □","Close", {duration:4000});
            }),
            complete:(()=> this.enable_preview=true)
          });
        }
        else{
          this.snackbar.open("Please select a jpg/jpeg/png file","Got it" ,{duration :3000});
          this.fname='';
          this.fformat='';
        this.file=null;
        }
      }
    }
    catch(err){
    console.log(err);
    }
  }

  deletefile(){
    this.fname='';
    this.fformat='';
    this.file=null;
```

```
      this.formdata.delete("image");
      this.formdata.delete("format")
      this.enable_preview=false;
    }

  // .subscribe(next?: ((value: string) => void) | null | undefined,
  // error?: ((error: any) => void) | null | undefined,
  //  complete?: (() => void) | null | undefined): Subscription (+2 overloads)

    predict(){
        if(this.file){
        this.route.navigate(['result']);
        }
        else{
          this.snackbar.open("Please select a file ","Okay",{duration:3000});
        }
    }
}
```

about-dialog.html

```
<mat-card-title>About Handwritten Digit Recognizer</mat-card-title>
<mat-dialog-content>
<br>
  <h3>Abstract</h3>
<p>Handwriting recognition is one of the compelling research works going on because every individual in this
    world has their own style of writing. It is the capability of the computer to identify and understand
    handwritten digits or characters automatically. Because of the progress in the field of science and
    technology, everything is being digitalized to reduce human effort. Hence, there comes a need for
    handwritten digit recognition in many real-time applications. MNIST data set is widely used for this
    recognition process and it has 70000 handwritten digits. We use Artificial neural networks to train these
    images and build a deep learning model. Web application is created where the user can upload an image of
    a handwritten digit. this image is analyzed by the model and the detected result is returned on to UI.</p>

  <h3>Procedure</h3>
  <ol>
    <li><b>Login -- </b>This is first page when you entered into the webapp. If you entered the user credentials(i.e.,Email id, Password) correctly, you are redirected to the next page</li>
    <li><b>Upload -- </b>In this page, you can upload the handwritten digit image from your local system. Immediately after pick the image, preview of the image is shown to you for extra verification.</li>
    <li><b>Result -- </b>The predicted value of the image that you upload is shown in this page. Column chart is also provided to see the result in graphical representation.</li>
  </ol>

  <h3>Upload a image which is similar to the image shown below.</h3>
  <img src="https://drive.google.com/uc?export=view&id=1yBFVSuzMFnIFxXh7PmhG3eR5nYLPZMDO" alt="example image" style="display: block; margin:auto;">

  <h3>Developed by:</h3>
  <ul>
    <li>Gomathi.N </li>
    <li>Rajeshwari.C</li>
    <li>Sakthi Manisha.M</li>
  <li>Tamilselvi.P</li>
  </ul>

</mat-dialog-content>
<mat-dialog-actions align="end">
  <button mat-button mat-dialog-close>Cancel</button>
</mat-dialog-actions>
```

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { LoginComponent } from './login/login.component';
import { UploadComponent } from './upload/upload.component';
import { ResultComponent } from './result/result.component';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';

const routes: Routes = [
  {path:'', redirectTo:'login', pathMatch:'full'},
  {path:'login', component:LoginComponent},
  {path:'upload', component:UploadComponent},
  {path:'result', component:ResultComponent},
  {path:"**", component:PageNotFoundComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule {  }
```

app.component.css

```
.abt-btn{
    background:#fff;
    color: #3f51b5;
}
.toolbar{
    display: flex;
    justify-content: space-around;
    flex-wrap: wrap;

}
.footer {
  display: flex;
  justify-content: space-around;
  flex-wrap: wrap;
  height: auto;
}
.toolbar span{
  display: flex;

}
```

app.component.html

```html
<mat-toolbar color="primary" class="toolbar">
    <span>
      <img src="assets\white icon.svg" alt="" height="30px">
      &nbsp;Handwritten Digit Recognizer
    </span>
    <button mat-raised-button (click)="openDialog()" class="abt-btn">About</button>
</mat-toolbar>
<router-outlet></router-outlet>
<mat-toolbar color="primary" class="footer">
  Developed by:
    <li>Harshath.M</li>
    <li>Priyanga.S</li>
    <li>Suvetha.M</li>
    <li>Ajeeth Kumar.S</li>
</mat-toolbar>
```

app.component.spec.ts

```ts
import { TestBed } from '@angular/core/testing';
import { RouterTestingModule } from '@angular/router/testing';
import { AppComponent } from './app.component';

describe('AppComponent', () => {
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [
        RouterTestingModule
      ],
      declarations: [
        AppComponent
      ],
    }).compileComponents();
  });

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
  });

  it(`should have as title 'Digit_Recognizer'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app.title).toEqual('Digit_Recognizer');
  });

  it('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.nativeElement as HTMLElement;
    expect(compiled.querySelector('.content 
span')?.textContent).toContain('Digit_Recognizer app is 
running!');
  });
});
```

app.component.ts

```
import { Component } from '@angular/core';
import { MatDialog } from '@angular/material/dialog';
import { MAT_DIALOG_DATA } from '@angular/material/dialog';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {
  constructor(private dialog:MatDialog){}

  openDialog(){
    this.dialog.open(AboutDialog);
  }

  title = 'Digit Recognizer';
}

@Component({
  selector: 'about-dialog',
  templateUrl:'./about-dialog.html'
})
export class AboutDialog{}
```

app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import {MatCardModule} from '@angular/material/card';
import {MatFormFieldModule} from '@angular/material/form-field';
import {MatInputModule} from '@angular/material/input';
import { ReactiveFormsModule } from '@angular/forms';
import {MatButtonModule} from '@angular/material/button';
import {MatButtonToggleModule} from '@angular/material/button-
toggle';
import { HttpClientModule } from '@angular/common/http';
import {MatSelectModule} from '@angular/material/select';
import {MatTableModule} from '@angular/material/table';
import {MatToolbarModule} from '@angular/material/toolbar';
import {MatIconModule} from '@angular/material/icon';
import {MatTooltipModule} from '@angular/material/tooltip';
import {MatSnackBarModule} from '@angular/material/snack-bar';
import {MatProgressSpinnerModule} from
'@angular/material/progress-spinner';
import {MatDialogModule} from '@angular/material/dialog';

import * as CanvasJSAngularChart from
'../assets/canvasjs.angular.component';
var CanvasJSChart = CanvasJSAngularChart.CanvasJSChart;

import { AboutDialog } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';
import { LoginComponent } from './login/login.component';
import { UploadComponent } from './upload/upload.component';
import { ResultComponent } from './result/result.component';
import { PageNotFoundComponent } from './page-not-found/page-not-
found.component';
```

```

@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    UploadComponent,
    ResultComponent,
    PageNotFoundComponent,
    CanvasJSChart,
  AboutDialog
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    MatCardModule,
    MatFormFieldModule,
    MatInputModule,
    ReactiveFormsModule,
    MatButtonModule,
    MatButtonToggleModule,
    HttpClientModule,
    MatSelectModule,
    MatTableModule,
    MatToolbarModule,
    BrowserAnimationsModule,
    MatIconModule,
    MatTooltipModule,
    MatSnackBarModule,
    MatProgressSpinnerModule,
  MatDialogModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

angular.json

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "Digit_Recognizer": {
      "projectType": "application",
      "schematics": {},
      "root": "",
      "sourceRoot": "src",
      "prefix": "app",
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/digit-recognizer",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.app.json",
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "styles": [
              "./node_modules/@angular/material/prebuilt-themes/indigo-pink.css",
            "src/styles.css"
            ],
            "scripts": []
          },
          "configurations": {
            "production": {
              "budgets": [
                {
                  "type": "initial",
                  "maximumWarning": "1mb",
                "maximumError": "2mb"
                },
                {
                  "type": "anyComponentStyle",
                  "maximumWarning": "2kb",
                "maximumError": "4kb"
                }
              ],
       "fileReplacements": [
        {
         "replace": "src/environments/environment.ts",
        "with": "src/environments/environment.prod.ts"
        }
       ],
       "outputHashing": "all"
      },
      "development": {
       "buildOptimizer": false,
       "optimization": false,
       "vendorChunk": true,
       "extractLicenses": false,
       "sourceMap": true,
      "namedChunks": true
      }
     },
     "defaultConfiguration": "production"
    },
    "serve": {
     "builder": "@angular-devkit/build-angular:dev-server",
     "configurations": {
      "production": {
       "browserTarget":
"Digit_Recognizer:build:production"
      },
      "development": {
       "browserTarget":
"Digit_Recognizer:build:development"
      }
     },
     "defaultConfiguration": "development"
    },
    "extract-i18n": {
     "builder": "@angular-devkit/build-angular:extract-
i18n",
     "options": {
     "browserTarget": "Digit_Recognizer:build"
     }
    },
    "test": {
     "builder": "@angular-devkit/build-angular:karma",
     "options": {
            "main": "src/test.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.spec.json",
            "karmaConfig": "karma.conf.js",
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "styles": [
              "./node_modules/@angular/material/prebuilt-themes/indigo-pink.css",
            "src/styles.css"
            ],
            "scripts": []
          }
        },
          "deploy": {
            "builder": "angular-cli-ghpages:deploy"
          }
        }
      }
    }
  }
```

package.json

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "Digit_Recognizer": {
      "projectType": "application",
      "schematics": {},
      "root": "",
      "sourceRoot": "src",
      "prefix": "app",
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/digit-recognizer",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.app.json",
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "styles": [
              "./node_modules/@angular/material/prebuilt-themes/indigo-pink.css",
            "src/styles.css"
            ],
            "scripts": []
          },
          "configurations": {
            "production": {
              "budgets": [
                {
                  "type": "initial",
                  "maximumWarning": "1mb",
                "maximumError": "2mb"
                },
                {
                  "type": "anyComponentStyle",
                  "maximumWarning": "2kb",
                "maximumError": "4kb"
                }
              ],
              "fileReplacements": [
                {
                  "replace": "src/environments/environment.ts",
                "with": "src/environments/environment.prod.ts"
                }
              ],
              "outputHashing": "all"
            },
            "development": {
              "buildOptimizer": false,
              "optimization": false,
              "vendorChunk": true,
              "extractLicenses": false,
              "sourceMap": true,
            "namedChunks": true
            }
          },
          "defaultConfiguration": "production"
        },
        "serve": {
          "builder": "@angular-devkit/build-angular:dev-server",
          "configurations": {
            "production": {
              "browserTarget": "Digit_Recognizer:build:production"
            },
            "development": {
              "browserTarget": "Digit_Recognizer:build:development"
            }
          },
          "defaultConfiguration": "development"
        },
        "extract-i18n": {
          "builder": "@angular-devkit/build-angular:extract-i18n",
          "options": {
          "browserTarget": "Digit_Recognizer:build"
          }
        },
        "test": {
          "builder": "@angular-devkit/build-angular:karma",
          "options": {
            "main": "src/test.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.spec.json",
            "karmaConfig": "karma.conf.js",
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "styles": [
              "./node_modules/@angular/material/prebuilt-themes/indigo-pink.css",
            "src/styles.css"
            ],
               "scripts": []
             }
           },
           "deploy": {
             "builder": "angular-cli-ghpages:deploy"
           }
         }
       }
     }
   }
```