|  |  |
| --- | --- |
| TEAM ID | PNT2022MID50751 |
| PROJECT NAME | A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION SYSTEM |
| TEAM MEMBERS | MAGAMARI, MAHALAKSHMI,  RAJESHWARI, RENUGA, SHAKI |

# INTRODUCTION

* 1. Project Overview
  2. Purpose

# LITERATURE SURVEY

* 1. Existing problem
  2. References
  3. Problem Statement Definition

# IDEATION & PROPOSED SOLUTION

* 1. Empathy Map Canvas
  2. Ideation & Brainstorming
  3. Proposed Solution
  4. Problem Solution fit

# REQUIREMENT ANALYSIS

* 1. Functional requirement
  2. Non-Functional requirements

# PROJECT DESIGN

* 1. Data Flow Diagrams
  2. Solution & Technical Architecture
  3. User Stories

# PROJECT PLANNING & SCHEDULING

* 1. Sprint Planning & Estimation
  2. Sprint Delivery Schedule
  3. Reports from JIRA

# CODING & SOLUTIONING

* 1. Features
  2. Database Schema

# TESTING

* 1. Test Cases
  2. User Acceptance Testing

# RESULTS

* 1. Performance Metrics

# ADVANTAGES & DISADVANTAGES

1. **CONCLUSION**

# FUTURE SCOPE

1. **APPENDIX**

Source Code

GitHub & Project Demo Link

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.

1. **LITERATURE SURVEY**

**Existing problem**

The existing machine learning algorithms having very low accuracy, so as to improve the accuracy, the Convolution Neural Network (CNN) can be used. However, every neural network has some error rate due to extensive training data,which is structured to meet the model’s requirements.

**References**

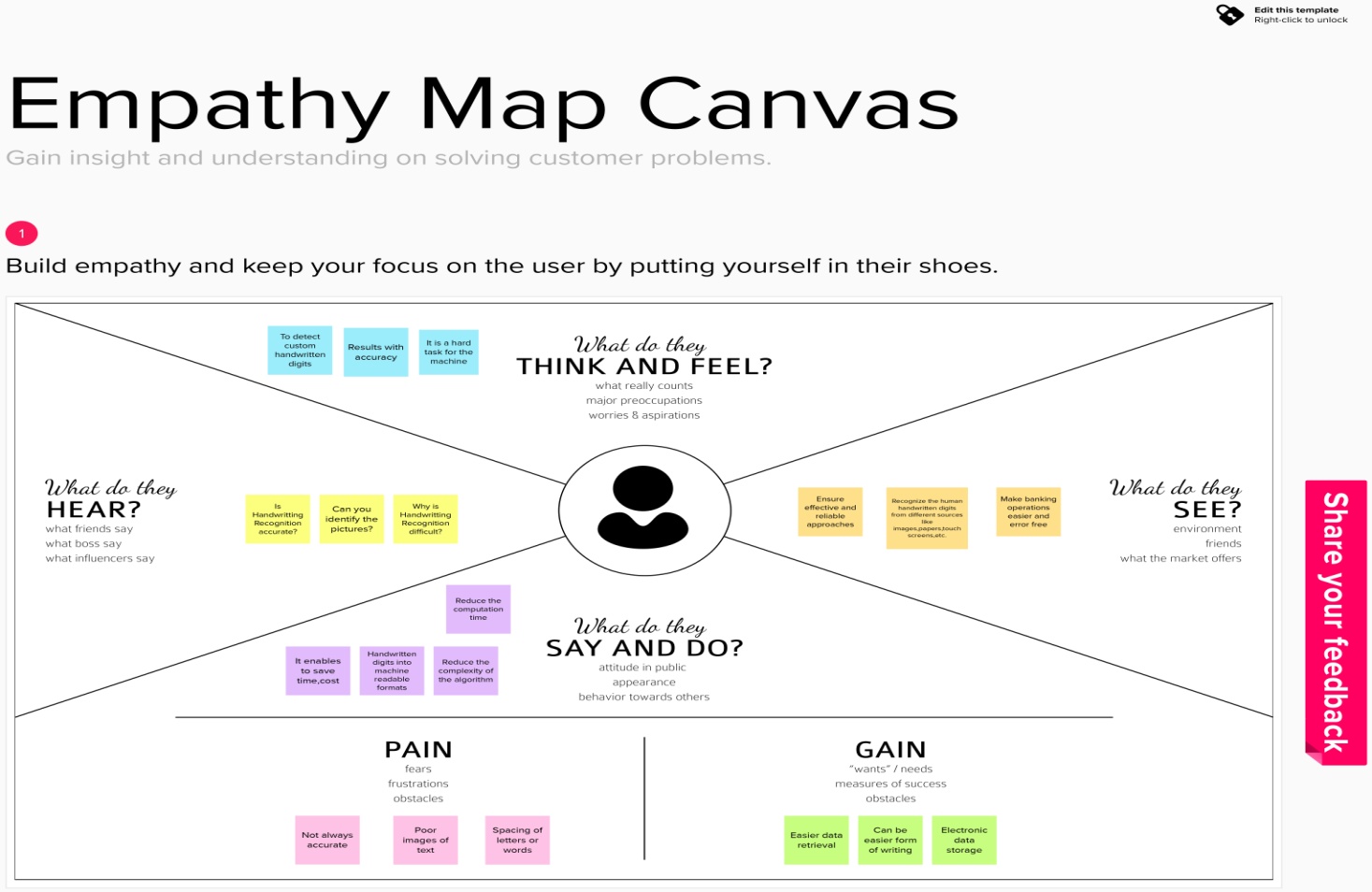
[1] K. Swetha, Y. Hithaishi, N. L. Tejaswini, P. Parthasaradhi, P.V. Venkateswara Rao “Handwritten Digit Recognition Using OpenCV and CNN”, *INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS(IJCRT)*, Volume 9, Issue 6 June 2021.

[2] Huan Li and Ping Hu, “AVN: An Adversarial Variation Network Model for Handwritten Signature Verification”, *IEEE TRANSACTION ON MULTIMEDIA,* VOL. 24, 2022

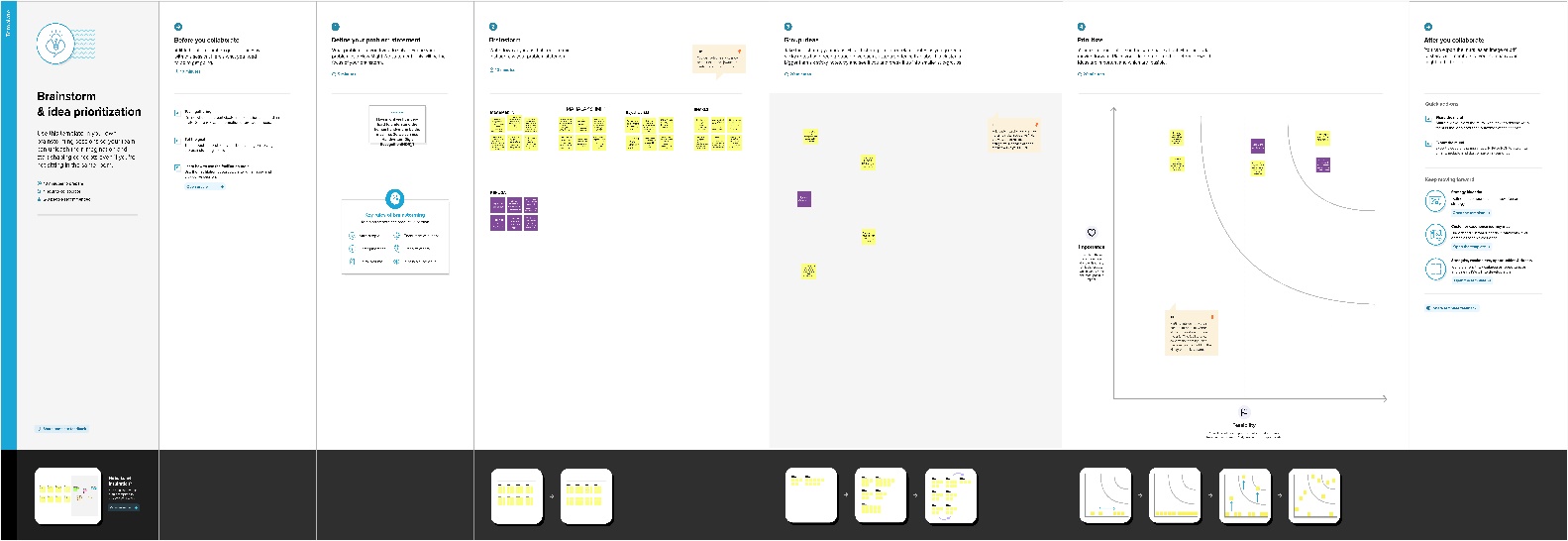
**Problem statement definition**

Every one have different handwriting style. It is very hard to understand the human handwriting by the machine. So we can use Handwritten Digit Recognition System (HDRS). HDR is the process of converting images of handwritten digit into digital format. Recognition accuracy and computation time still require further improvement. Application areas of handwritten digit recognition system includes medical, banking, student management and taxation process. HDR possesses great flexibility.

1. **IDEATION & PROPOSED SOLUTION**
   1. **Empathy Map Canvas**

****

* 1. **Ideation & Brainstorming**

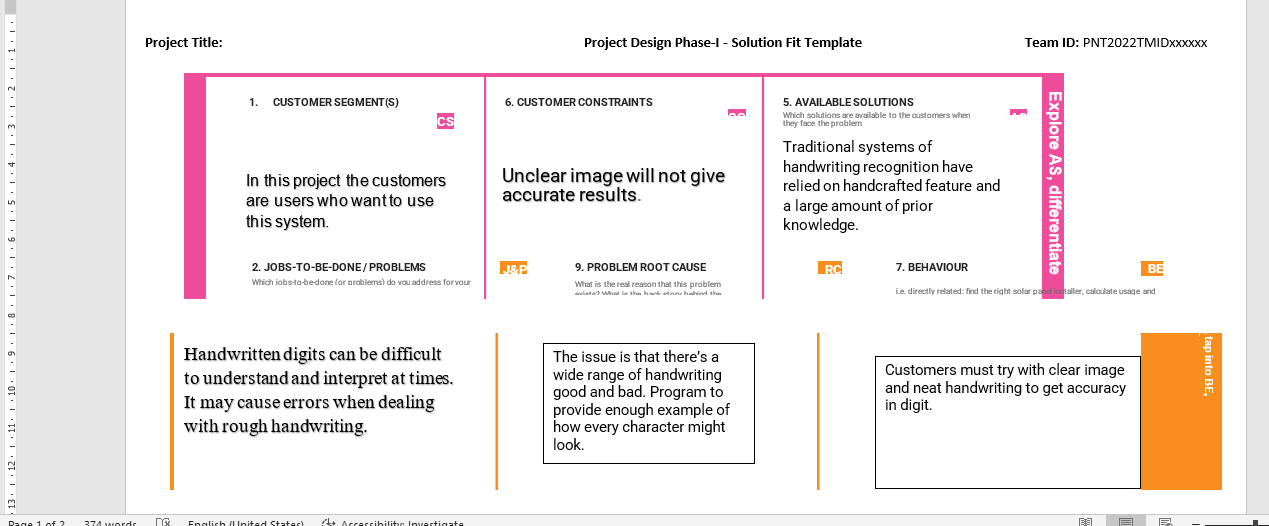


**3**

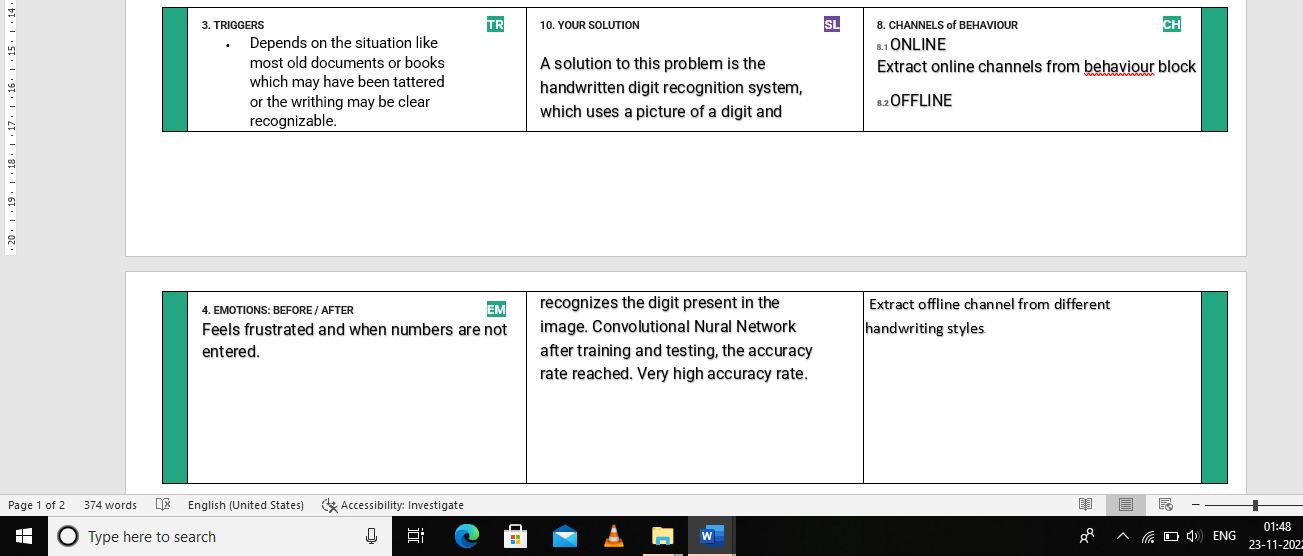
**Proposed Solution**

|  |  |  |
| --- | --- | --- |
| **S.N**  **o.** | **Parameters** | **Description** |
| 1. | Problem Statement (problem to be solved) | * Everyone have different handwriting style. It is very hard to understand the human handwriting by the machine. The purpose of this work is to minimize the number of pixels by using as input the data extracted and calculated from the initial image. |
| 2. | Idea / Solution Description | * The application is the capability of the computer to identify and understand the human handwritten digit automatically instead of typing every time. |
| 3. | Novelty / Uniqueness | * If any unidentified handwritten digits are uploaded, it show similar matched digits from the database and ask confirmation from the user. |
| 4. | Social Impact / Customer satisfaction | * The recommendation system and user interface of the platform will satisfies the customers in detection of banks for reading cheques, vehicle numbers, post office for arranging letters. |
| 5. | Business Model (Revenue Model) | * We are building a platform for the dealer so that he may sell his items through it and we will pau the dealer based on how well the consumer is satisfied with the product. |
| 6. | Scalability of the solution | * We will charge dealers based on the quality of goods they sell through our platform. This model can be expanded to include more attributes for more accurate detection. |

* 1. **Problem Solution fit**



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



**Explore AS, differentiate**

**Deﬁne CS, ﬁt into CC**

1. **REQUIREMENT ANALYSIS**
   1. **Functional requirement**

Following are the functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **FR**  **No.** | **Functional Requirement**  **(Epic)** | **Sub Requirement (Story / Sub-Task)** |
| **FR-1** | Login | Login with verified email id and password using mongodb |
| **FR-2** | Image Upload | Upload a handwritten digit image in a supported format |
| **FR-3** | Web Browser | Mobile or Desktop browser is needed to make use of digit  recognition. |

* 1. **Non-functional Requirements:**

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

|  |  |  |
| --- | --- | --- |
| **FR**  **No.** | **Non-Functional**  **Requirement** | **Description** |
| **NFR-**  **1** | **Usability** | Simple to understand the UI and easy to get the recognition  of handwritten digits |
| **NFR-**  **2** | **Security** | No security measures are taken for our application |
| **NFR-**  **3** | **Reliability** | Withstand without any occurrence of error for a long period  of time |
| **NFR-**  **4** | **Performance** | Light-weight application makes the performance better. |
| **NFR-**  **5** | **Availability** | New pages include will doesn’t affect the system |
| **NFR-**  **6** | **Scalability** | Large number of users can recognize the digits at a time  without any restriction. |

1. **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:**

[Diagram, timeline

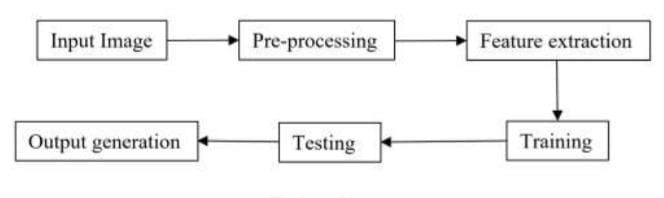
Description automatically generated](https://developer.ibm.com/patterns/visualize-unstructured-text/)

**DFD**

Diagram

Description automatically generated

* 1. **Solution & Technical Architecture**



* 1. **User stories**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Type** | **Functional Requireme nt (Epic)** | **User Story Numb**  **er** | **User Story / Task** | **Acceptance criteria** | **Priori ty** | **Release** |
| **Customer (Mobile user)** | Details | USN-1 | As a user, I can know the details of the fundamental usage of the application. | I can access information of details page | Low | Sprint-2 |
|  | Login | USN-2 | As a user, I will login with my email credentials and basic validation is verified. | I can access the next page if I am a verified user | High | Sprint-3 |
|  | Image upload | USN-3 | As a user, I will upload the handwritten digit image to the application | I can upload the image from the local  system | High | Sprint-2 |
|  | Recognized Result | USN-4 | As a user, I can see the predicted / recognized digits in the application | 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 details of the fundamental usage of the application. | I can access information of details page | Low | Sprint-2 |
|  | Login | USN-2 | As a user, I will login with my email credentials and basic validation is verified. | I can access the next page if I am a verified user | High | Sprint-3 |
|  | Image upload | USN-3 | As a user, I will upload the handwritten digit image to the application | I can upload the image from the local system | High | Sprint-2 |
|  | Recognized Result | USN-4 | As a user, I can see the predicted / recognized digits in the application | I can see the output of the recognized digit | High | Sprint-3 |

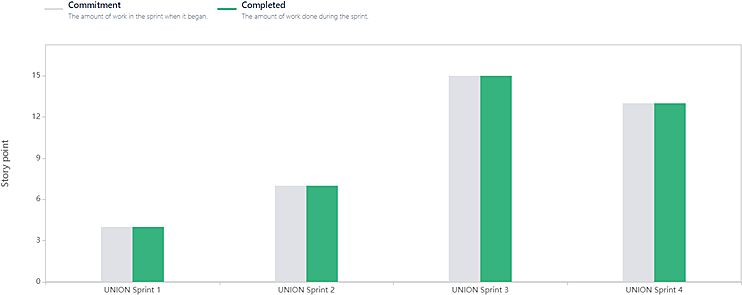
1. **PROJECT PLANNING & SCHEDULING**
   1. **Sprint Planning & Estimation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Sto ry Poi**  **nts** | **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 | Renuga,  Shaki |
| **Sprint- 1** |  | USN-2 | As a user, I can upload the image in any resolution. | 1 | Low | Mahalaksmi,magamari |
| **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 | Rajeshwari,  Renuga |
| **Sprint- 2** |  | USN-4 | As a user, I can pass the handwritten digit image for recognizing the digit. | 2 | Medium | Magamari,  Shaki |
| **Sprint- 2** |  | USN-5 | As a user, I can get the most suitable recognized digit. | 3 | High | Rajeshwari,Mahalakshmi |
| **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 | Magamari,  Renuga |
| **Sprint- 3** |  | USN-7 | As a user, I can know the details of the fundamental usage of the application. | 2 | Low | Mahalakshmi |
| **Sprint- 3** |  | USN-8 | As a user, I can see the predicted / recognized digits in the application | 8 | High | Rajeshwari, Shaki |
| **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 | Renuga  Magamari |

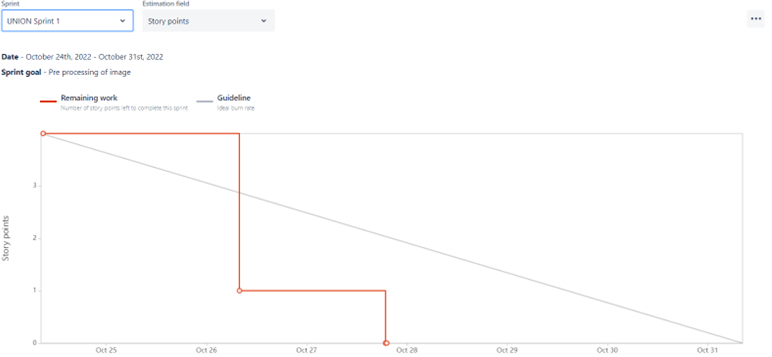
* 1. **Sprint Delivery Schedule**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Total Story Points** | **Durati on** | **Sprint Start Date** | **Sprint End Date (Planned)** | **Story Points Completed (as on Planned End Date)** | **Sprint Release Date**  **(Actual)** |
| **Sprint-1** | 20 | 6 Days | 24 Oct  2022 | 29 Oct 2022 | 20 | 29 Oct 2022 |
| **Sprint-2** | 20 | 6 Days | 31 Oct  2022 | 05 Nov 2022 | 20 | 05 Nov 2022 |
| **Sprint-3** | 20 | 6 Days | 07 Nov  2022 | 12 Nov 2022 | 20 | 12 Nov 2022 |
| **Sprint-4** | 20 | 6 Days | 14 Nov  2022 | 19 Nov 2022 | 20 | 19 Nov 2022 |

* 1. **Reports from JIRA Velocity Report**



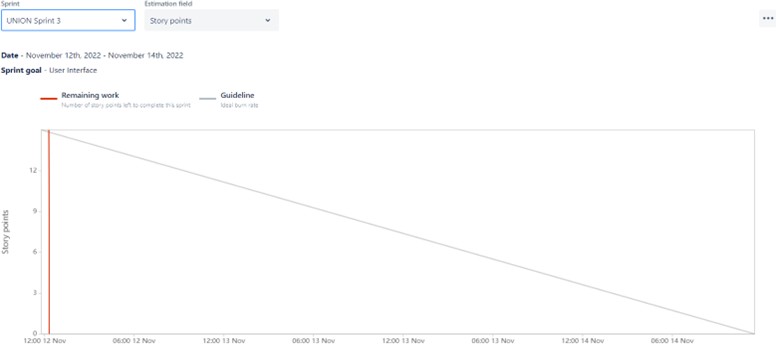
**Sprint 1 – Burndown Chart**



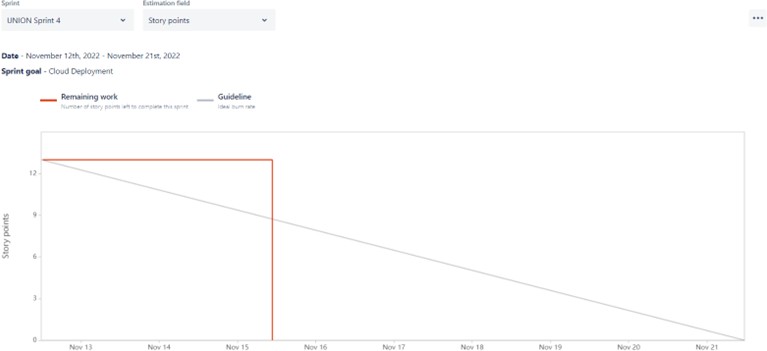
**Sprint 2 – Burndown Chart**



**Sprint 3 – Burndown Chart**



**Sprint 4 – Burndown Chart**

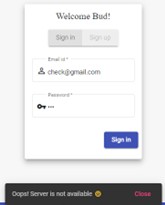


1. **CODING & SOLUTIONING**
   1. **Features**

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



* 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”.



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



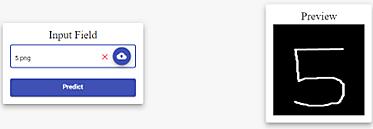
* If the user is available in mongodb collections, he/she is redirected to the next page. If the new user credentials are to be entered, it will done by creating a new document with necessary details in the mongodb.



* 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.



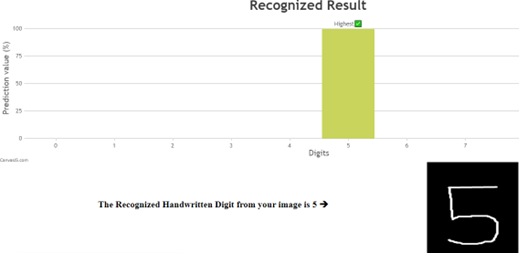
* Tiptool is useful to know about what we are doing. There is remove file button, to remove the file in formdata. Immediately after removing the file, the preview card is disappeared to maintain the flow.



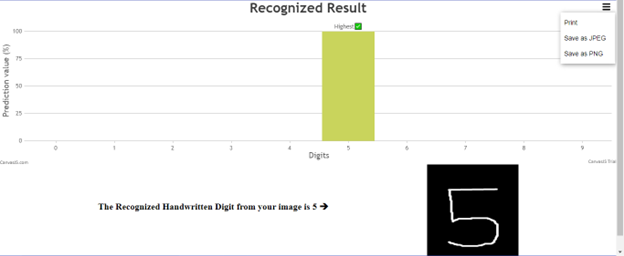
* Uploading the files is restricted only to image files. In special case, user enters a other format file, it will collapse the system. For overcoming that, new snackbar is created and shows the corresponding message if user enters the other format files.



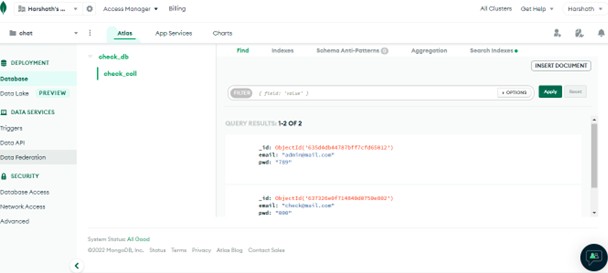
* Main feature of our project is showing the predicted results as a column chart, it makes the user to easily understand the results. Chart is created by canvas js library.



* Chart can be downloaded if it is needed.



* 1. **Database Schema**



**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 |

1. **RESULTS**
   1. **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 |  |

1. **ADVANTAGES & DISADVANTAGES ADVANTAGES**
2. User friendly interface makes the user to navigate easily to other pages.
3. It provides high accuracy.
4. The quick prediction will save the time of the users.
5. 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.
6. It provides results in graphical representation for easy understanding.
7. Handwritten Digit Recognizer is an angular js application and it is also deployed in github pages for easy access.
8. Login credentials are not static it will be fetched from mongodb atlas collections.
9. 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.
4. **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%.

1. **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.

# APPENDIX

**Source Code**

**DigitAPI.py**

1

2

3

4

5

6

7

8

9

10

11

12

13

14

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

15 app=create\_app()

16 if(' main '== name ):

17 app.run(host='0.0.0.0')

# endpoints.py

1

2

3

4

5

6

7

8

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/?retryWrit es=true&w=majority'

9 client = pymongo.MongoClient(uri) 10 db = client.check\_db

1. coll = db.check\_coll
2. print('connection has made') 13
3. def api\_endpoints(endpoints):
4. @endpoints.route('/verify', methods=['POST'])
5. def verify():
6. try:
7. email = request.form.get('email')
8. pwd = request.form.get("pwd")
9. flag = coll.find\_one({"email":email, "pwd":pwd})
10. status={
11. 'statuscode' : 200, 23 }
12. if(flag!=None):
13. status['statusmessage'] = "true"
14. else:
15. status['statusmessage'] = "false"
16. except Exception as e:
17. status={
18. 'statuscode' : 400,
19. 'statusmessage' : str(e) 32 }

33 return status 34

1. @endpoints.route('/upload', methods=['POST'])
2. def upload():
3. input = request.files.get("image")
4. global format
5. format = request.form.get("format")
6. img= Image.open(input)

41 img = img.resize((200,200))

1. img.save("files/input."+format)
2. return send\_file(path\_or\_file = "files/input."+format) 44
3. @endpoints.route('/predict', methods=['GET'])
4. def predict():
5. result = {};
6. img=Image.open("files/input."+format).convert("L")
7. img = img.resize((28,28))
8. im2arr=np.array(img)
9. im2arr = im2arr.reshape(1,28,28,1)
10. y\_pred = model.predict(im2arr)
11. result["value"] = int(np.argmax(y\_pred))
12. print("Predicted value is",result)
13. return result 56
14. @endpoints.route('/image', methods=['GET'])
15. def image():

59

60

return send\_file(path\_or\_file = "files/input."+format)

return endpoints

# Digit Recognizer (AngularJS files) : Login Component login.component.html

1. <div class="entire-login">
2. <mat-card class="mat-elevation-z8">
3. <mat-card-header class="flex-center">
4. <mat-card-title >
5. Welcome Bud!
6. </mat-card-title>
7. </mat-card-header>
8. <mat-card-actions class="flex-center">
9. <mat-button-toggle-group style="margin: auto;" appearance="legacy">
10. <mat-button-toggle value="sign\_in" checked="true">Sign in</mat-button-toggle>
11. <mat-button-toggle value="sign\_up" disabled="true">Sign up</mat-button-toggle>
12. </mat-button-toggle-group>
13. </mat-card-actions>
14. <mat-card-content class="card-content">
15. <mat-form-field appearance="outline">
16. <mat-label>Email id</mat-label>
17. <mat-icon matPrefix>perm\_identity</mat-icon>
18. <input id="email" name="email" matInput type='email' [formControl]="emailFC" placeholder="" #email/>
19. <mat-error \*ngIf="emailFC.hasError('required')">
20. Email id is required
21. </mat-error>
22. <mat-error \*ngIf="emailFC.hasError('email') &&

!emailFC.hasError('required')">

1. Valid email id is required
2. </mat-error>
3. </mat-form-field>
4. <mat-form-field appearance="outline">
5. <mat-label >Password</mat-label>

28

29

30

31

32

33

34

35

<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>

1. </mat-card-actions>
2. </mat-card-content>
3. </mat-card>
4. </div>

# login.component.css

1

2

3

4

5

6

7

8

9

10

11

.entire-login{ display: flex;

justify-content: center; align-items: center; height:85vh; background: #f7f7f7;

}

.card-content{ display: flex;

flex-direction: column;

12 }

1. .flex-center{
2. display: flex;
3. justify-content: center; 16 }
4. .mat-card{
5. /\* background-color: aliceblue; \*/
6. box-shadow: 50px;
7. font-family: 'Times New Roman', Times, serif; 21 }

**login.component.spec.ts**

1

2

3

// Done by Harshath.M

import

{

ComponentFixture,

TestBed

}

from

'@angular/core/testing';

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

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();

});

25 });

# login.component.ts

1

2

3

4

5

6

7

8

9

10

11

12

// 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']

13 })

1. export class LoginComponent implements OnInit {
2. email = ""; 16 pwd = "";
3. invalid= true;
4. showbutton = true; 19

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

21

22 }

23

1. ngOnInit(): void {}
2. emailFC = new FormControl('',[Validators.email,Validators.required]);
3. passwordFC = new FormControl('',[Validators.required]); 27
4. val\_credentials(email:string,pwd:string){
5. let formdata = new FormData();
6. formdata.append("email",email);
7. formdata.append("pwd",pwd);

32 let api\_url = "http://127.0.0.1:5000/api/";

1. this.http.post(api\_url+"verify",formdata).subscribe({
2. next:((res:any)=>{
3. if(res.statusmessage=='true'){
4. this.snackbar.open("Email and Password is verified😎","Welcome", {duration:2000});
5. this.route.navigate(['/upload']); 38 }
6. else if(res.statusmessage=="false"){
7. this.snackbar.open("Invalid Credentials ♂","Close",

{duration:4000}); 41 }

1. else{
2. this.snackbar.open("Oops! Something went wrong😟","Close", {duration:4000});

44 }

45 }),

46 error:(()=>{

1. this.showbutton=true;
2. this.snackbar.open("Oops! Server is not available

😟","Close", {duration:4000}); 49 })

50 });

51 }

52 }

# Page not found Component

**page-not-found.component.html**

1. <div>
2. <p>Page not found!!!</p>
3. <p>Please enter the correct URL...</p> 4 </div>

# page-not-found.component.css

1

2

3

4

5

6

7

div{

display: flex;

flex-direction: column; justify-content: center; align-items: center; height: 80%;

}

**page-not-found.component.spec.ts**

1. import { ComponentFixture, TestBed } from '@angular/core/testing';
2. import { PageNotFoundComponent } from './page-not- found.component';
3. describe('PageNotFoundComponent', () => {
4. let component: PageNotFoundComponent;
5. let fixture: ComponentFixture<PageNotFoundComponent>;
6. beforeEach(async () => {
7. await TestBed.configureTestingModule({
8. declarations: [ PageNotFoundComponent ] 9 })

10 .compileComponents(); 11

1. fixture = TestBed.createComponent(PageNotFoundComponent);
2. component = fixture.componentInstance;
3. fixture.detectChanges(); 15 });

16

1. it('should create', () => {
2. expect(component).toBeTruthy();

19 });

20 });

# page-not-found.component.ts

1

2

3

4

5

6

7

8

9

10

11

12

13

14

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 {

}

15 }

**Result Component result.component.html**

1

2

<div class="middle" \*ngIf="!load\_graph">

<mat-progress-spinner mode="indeterminate" strokeWidth="7" ></mat-progress-spinner>

</div>

<canvasjs-chart class="chart" \*ngIf="load\_graph"

diameter="60"

3

4

[options]="chartOptions" [styles]="{width: '100%', height:'360px'}"></canvasjs-chart>

5

1. <div \*ngIf="load\_graph" class="description">
2. <h2>The Recognized Handwritten Digit from your image is

{{pred\_value}} 🡺</h2>

1. <img [src]="preview">
2. </div>

# result.component.css

1. .middle{
2. display: flex;
3. justify-content: center;
4. align-items: center;
5. height:80%; 6 }
6. .chart{
7. margin-top:20px; 9 }
8. .metrics-table{
9. display: flex;
10. justify-content: center;
11. align-items: center;
12. flex-direction: column;
13. margin: 50px 0; 16 }
14. table{
15. min-width: 350px;
16. margin-bottom: 20px; 20 }
17. .description{
18. display: flex;
19. justify-content: space-evenly;
20. flex-wrap: wrap;
21. align-items: center; 26 }
22. .description h2{
23. font-family: 'Times New Roman', Times, serif;
24. font-weight: bold; 30 }

**result.component.spec.ts**

1 import { ComponentFixture, TestBed } from

'@angular/core/testing';

2

3

4

5

6

7

8

9

10

11

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

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

let fixture: ComponentFixture<ResultComponent>;

beforeEach(async () => {

await TestBed.configureTestingModule({ declarations: [ ResultComponent ]

12 })

13 .compileComponents(); 14

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

19

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

23 });

# result.component.ts

1

2

3

4

5

6

7

8

9

10

11

12

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']

13 })

14 export class ResultComponent implements OnInit { 15

16

17

18

19

20

21

22

23

24

25

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({

1. next:((res:any)=>{
2. let objecturl = URL.createObjectURL(res);
3. this.preview = this.domsanitizer.bypassSecurityTrustUrl(objecturl);

29 })

30 });

31

this.http.get(this.api\_url+"predict").subscribe((res:any)=>{

1. this.pred\_value = res.value;
2. this.open\_page();
3. this.load\_graph = true; 35 });

36 }

1. getDataPoints() {
2. let dataPoints =[];

39 for (var i = 0; i <= 9 ; i++)

1. dataPoints.push({
2. x: i,

42 y: 0

43 });

1. dataPoints[this.pred\_value]= { x : this.pred\_value ,y:100, indexLabel: "Highest\u2705"};
2. console.log(dataPoints);
3. return dataPoints; 47 }

48

1. open\_page(){
2. this.chartOptions = {

//https://canvasjs.com/angular-charts/chart-index-data-label/

1. animationEnabled: true,
2. exportEnabled: true,
3. theme: "light2",
4. title: {
5. text: "Recognized Result" 56 },
6. axisX: {
7. title: "Digits",
8. interval: 1 60 },
9. axisY:{
10. title: "Prediction value (%)",
11. maximum: 110,
12. interval:25

65

66

67

68

69

70

71

72 }

},

data: [{

type: "column", dataPoints: this.getDataPoints()

}]

}

}

# Upload Component upload.component.html

1. <div class="entire">
2. <mat-card class="card mat-elevation-z8">
3. <mat-card-title style="text-align: center;">Input Field</mat-card-title>
4. <mat-form-field appearance="outline">
5. <input hidden type='file' accept="image/\*" #fileclick (change)="select\_file($event)" >
6. <input readonly matInput value="{{this.fname}}" placeholder="Choose image file" >
7. <button \*ngIf="this.file" matSuffix (click)="deletefile()" matTooltip="Remove File" matTooltipPosition = "above" color="warn" mat-icon-button>
8. <mat-icon>close</mat-icon>
9. </button>
10. <button matSuffix mat-mini-fab color="primary" (click)="fileclick.click()" matTooltip="Select a file" matTooltipPosition="right">
11. <mat-icon>backup</mat-icon>
12. </button>
13. </mat-form-field>
14. <button (click)="predict()" mat-raised-button color="primary" style="min-height: 40px;">
15. <span>Predict</span>
16. </button>
17. </mat-card> 18
18. <mat-card class="card mat-elevation-z8"

\*ngIf="enable\_preview">

1. <mat-card-title style="text-align: center;">Preview</mat-card-title>

21

22

<img [src]="preview">

</mat-card>

23 </div>

# upload.component.css

1

1. .entire{
2. display: flex;
3. justify-content: space-evenly;
4. flex-wrap: wrap;
5. align-items: center;
6. height:85vh;
7. background: #f7f7f7; 9 }
8. .card{
9. display: flex;
10. flex-direction: column;
11. justify-content: center; 14 }
12. .mat-card{
13. box-shadow: 50px;
14. font-family: 'Times New Roman', Times, serif; 18 }

**upload.component.spec.ts**

1 import { ComponentFixture, TestBed } from

'@angular/core/testing';

2

3

4

5

6

7

8

9

10

11

12

13

14

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;

15

16

17

18

19

20

fixture.detectChanges();

});

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

});

# upload.component.ts

1

2

3

4

5

6

7

8

9

10

11

12

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']

13 })

14 export class UploadComponent implements OnInit { 15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

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{

1. this.file = event.target.files[0];
2. if(this.file){
3. this.fname = this.file.name;
4. this.fformat = this.file.type.split('/')[1];
5. if(this.all\_formats.indexOf(this.fformat)!=-1){
6. this.formdata= new FormData();
7. this.formdata.append('image', this.file);
8. this.formdata.append("format", this.fformat); 42

43 let api\_url = "http://127.0.0.1:5000/api/upload"; 44

this.http.post(api\_url,this.formdata,{responseType:'blob'}).subs cribe({

1. next:((res:any)=>{
2. let objecturl = URL.createObjectURL(res);
3. this.preview = this.domsanitizer.bypassSecurityTrustUrl(objecturl);

48 }),

49 error:(()=>{

50 this.snackbar.open("Oops! Server is not available

😟","Close", {duration:4000}); 51 }),

52 complete:(()=> this.enable\_preview=true) 53 });

54 }

1. else{
2. this.snackbar.open("Please select a jpg/jpeg/png file","Got it" ,{duration :3000});
3. this.fname='';
4. this.fformat='';
5. this.file=null; 60 }

61 }

62 }

1. catch(err){
2. console.log(err); 65 }

66 }

67

1. deletefile(){
2. this.fname='';
3. this.fformat='';
4. this.file=null;

72

73

74

75

76

this.formdata.delete("image");

this.formdata.delete("format") this.enable\_preview=false;

}

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

80

1. predict(){
2. if(this.file){
3. this.route.navigate(['result']); 84 }
4. else{
5. this.snackbar.open("Please select a file

","Okay",{duration:3000}); 87 }

88 }

89 }

**about-dialog.html**

1. <mat-card-title>About Handwritten Digit Recognizer</mat-card- title>
2. <mat-dialog-content> 3 <br>
3. <h3>Abstract</h3>
4. <p>Handwriting recognition is one of the compelling research works going on because every individual in this
5. world has their own style of writing. It is the capability of the computer to identify and understand
6. handwritten digits or characters automatically. Because of the progress in the field of science and
7. technology, everything is being digitalized to reduce human effort. Hence, there comes a need for
8. handwritten digit recognition in many real-time applications. MNIST data set is widely used for this
9. recognition process and it has 70000 handwritten digits. We use Artificial neural networks to train these
10. images and build a deep learning model. Web application is created where the user can upload an image of
11. a handwritten digit. this image is analyzed by the model and the detected result is returned on to UI.</p>

13

1. <h3>Procedure</h3>
2. <ol>
3. <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>
4. <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>
5. <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>

19 </ol>

20

1. <h3>Upload a image which is similar to the image shown below.</h3>
2. <img src=" https://drive.google.com/uc?export=view&id=1yBFVSuzMFnIFxXh7PmhG3 eR5nYLPZMDO" alt="example image" style="display: block; margin:auto;">

23

1. <h3>Developed by:</h3>
2. <ul>
3. <li>Harshath.M</li>
4. <li>Priyanga.S</li>
5. <li>Suvetha.M</li>
6. <li>Ajeeth Kumar.S</li> 30 </ul>

31

1. </mat-dialog-content>
2. <mat-dialog-actions align="end">
3. <button mat-button mat-dialog-close>Cancel</button> 35 </mat-dialog-actions>

# app-routing.module.ts

1 import { NgModule } from '@angular/core';

1. import { RouterModule, Routes } from '@angular/router';
2. import { LoginComponent } from './login/login.component';
3. import { UploadComponent } from './upload/upload.component';
4. import { ResultComponent } from './result/result.component';
5. import { PageNotFoundComponent } from './page-not-found/page-not- found.component';

7

1. const routes: Routes = [
2. {path:'', redirectTo:'login', pathMatch:'full'},
3. {path:'login', component:LoginComponent},
4. {path:'upload', component:UploadComponent},
5. {path:'result', component:ResultComponent},
6. {path:"\*\*", component:PageNotFoundComponent} 14 ];

15

1. @NgModule({
2. imports: [RouterModule.forRoot(routes)],
3. exports: [RouterModule] 19 })

20 export class AppRoutingModule { }

# app.component.css

1. .abt-btn{
2. background:#fff;
3. color: #3f51b5;

4 }

1. .toolbar{
2. display: flex;
3. justify-content: space-around;
4. flex-wrap: wrap; 9

10 }

1. .footer {
2. display: flex;
3. justify-content: space-around;
4. flex-wrap: wrap;
5. height: auto; 16 }
6. .toolbar span{
7. display: flex; 19

20 }

**app.component.html**

1. <mat-toolbar color="primary" class="toolbar">
2. <span>
3. <img src="assets\white icon.svg" alt="" height="30px">
4. &nbsp;Handwritten Digit Recognizer
5. </span>
6. <button mat-raised-button (click)="openDialog()" class="abt- btn">About</button>
7. </mat-toolbar>
8. <router-outlet></router-outlet>
9. <mat-toolbar color="primary" class="footer">
10. Developed by:
11. <li>Harshath.M</li>
12. <li>Priyanga.S</li>
13. <li>Suvetha.M</li>
14. <li>Ajeeth Kumar.S</li> 15 </mat-toolbar>

# app.component.spec.ts

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

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;

20

21

22

23

24

25

26

27

28

29

30

31

32

33

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');

});

34

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!');

});

35 });

**app.component.ts**

1

2

3

4

5

6

7

8

9

10

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']

})

11 export class AppComponent {

12

13

14

15

16

17

18

19 }

20

constructor(private dialog:MatDialog){}

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

}

title = 'Digit\_Recognizer';

21 @Component({

1. selector: 'about-dialog',
2. templateUrl:'./about-dialog.html' 24 })

25 export class AboutDialog{}

# app.module.ts

1. import { NgModule } from '@angular/core';
2. import { BrowserModule } from '@angular/platform-browser'; 3
3. import { AppRoutingModule } from './app-routing.module';
4. import { AppComponent } from './app.component';
5. import {MatCardModule} from '@angular/material/card';
6. import {MatFormFieldModule} from '@angular/material/form-field';
7. import {MatInputModule} from '@angular/material/input';
8. import { ReactiveFormsModule } from '@angular/forms';
9. import {MatButtonModule} from '@angular/material/button';
10. import {MatButtonToggleModule} from '@angular/material/button- toggle';
11. import { HttpClientModule } from '@angular/common/http'; 13 import {MatSelectModule} from '@angular/material/select'; 14 import {MatTableModule} from '@angular/material/table';

15 import {MatToolbarModule} from '@angular/material/toolbar'; 16 import {MatIconModule} from '@angular/material/icon';

1. import {MatTooltipModule} from '@angular/material/tooltip';
2. import {MatSnackBarModule} from '@angular/material/snack-bar'; 19 import {MatProgressSpinnerModule} from

'@angular/material/progress-spinner';

20 import {MatDialogModule} from '@angular/material/dialog'; 21

1. import \* as CanvasJSAngularChart from '../assets/canvasjs.angular.component';
2. var CanvasJSChart = CanvasJSAngularChart.CanvasJSChart; 24
3. import { AboutDialog } from './app.component';
4. import { BrowserAnimationsModule } from '@angular/platform- browser/animations';
5. import { LoginComponent } from './login/login.component'; 28 import { UploadComponent } from './upload/upload.component'; 29 import { ResultComponent } from './result/result.component';

30 import { PageNotFoundComponent } from './page-not-found/page-not- found.component';

31

1. @NgModule({
2. declarations: [
3. AppComponent,
4. LoginComponent,
5. UploadComponent,
6. ResultComponent,
7. PageNotFoundComponent,
8. CanvasJSChart,
9. AboutDialog 41 ],
10. imports: [
11. BrowserModule,
12. AppRoutingModule,
13. MatCardModule,
14. MatFormFieldModule,
15. MatInputModule,
16. ReactiveFormsModule,
17. MatButtonModule,
18. MatButtonToggleModule,
19. HttpClientModule,
20. MatSelectModule,
21. MatTableModule,
22. MatToolbarModule,
23. BrowserAnimationsModule,
24. MatIconModule,
25. MatTooltipModule,
26. MatSnackBarModule,
27. MatProgressSpinnerModule,
28. MatDialogModule 61 ],
29. providers: [],
30. bootstrap: [AppComponent] 64 })

65 export class AppModule { } 66

# angular.json

1 {

2 "$schema": "./node\_modules/@angular/cli/lib/config/schema.json",

1. "version": 1,
2. "newProjectRoot": "projects",
3. "projects": {
4. "Digit\_Recognizer": {
5. "projectType": "application",
6. "schematics": {},

9 "root": "",

1. "sourceRoot": "src",
2. "prefix": "app",
3. "architect": {
4. "build": {
5. "builder": "@angular-devkit/build-angular:browser",
6. "options": {
7. "outputPath": "dist/digit-recognizer",
8. "index": "src/index.html",
9. "main": "src/main.ts",
10. "polyfills": "src/polyfills.ts",
11. "tsConfig": "tsconfig.app.json",
12. "assets": [
13. "src/favicon.ico",
14. "src/assets"

24 ],

1. "styles": [
2. "./node\_modules/@angular/material/prebuilt- themes/indigo-pink.css",
3. "src/styles.css" 28 ],

29 "scripts": [] 30 },

1. "configurations": {
2. "production": {
3. "budgets": [

34 {

1. "type": "initial",
2. "maximumWarning": "1mb",
3. "maximumError": "2mb" 38 },

39 {

1. "type": "anyComponentStyle",
2. "maximumWarning": "2kb",
3. "maximumError": "4kb" 43 }

44 ],

45 "fileReplacements": [ 46 {

1. "replace": "src/environments/environment.ts",
2. "with": "src/environments/environment.prod.ts" 49 }

50 ],

51 "outputHashing": "all" 52 },

1. "development": {
2. "buildOptimizer": false,
3. "optimization": false,
4. "vendorChunk": true,
5. "extractLicenses": false,
6. "sourceMap": true,
7. "namedChunks": true 60 }

61 },

62 "defaultConfiguration": "production" 63 },

1. "serve": {
2. "builder": "@angular-devkit/build-angular:dev-server",
3. "configurations": {
4. "production": {
5. "browserTarget": "Digit\_Recognizer:build:production"

69 },

1. "development": {
2. "browserTarget": "Digit\_Recognizer:build:development"

72 }

73 },

74 "defaultConfiguration": "development" 75 },

1. "extract-i18n": {
2. "builder": "@angular-devkit/build-angular:extract- i18n",
3. "options": {
4. "browserTarget": "Digit\_Recognizer:build" 80 }

81 },

1. "test": {
2. "builder": "@angular-devkit/build-angular:karma",
3. "options": {
4. "main": "src/test.ts",
5. "polyfills": "src/polyfills.ts",
6. "tsConfig": "tsconfig.spec.json",
7. "karmaConfig": "karma.conf.js",
8. "assets": [
9. "src/favicon.ico",
10. "src/assets"

92 ],

1. "styles": [
2. "./node\_modules/@angular/material/prebuilt- themes/indigo-pink.css",
3. "src/styles.css" 96 ],

97 "scripts": [] 98 }

99 },

100

101

102

103

104

105

106

"deploy": {

"builder": "angular-cli-ghpages:deploy"

}

}

}

}

}

# package.json

1 {

2

3

4

5

6

7

8

9

10

11

12

13

14

15

"$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": {

1. "outputPath": "dist/digit-recognizer",
2. "index": "src/index.html",
3. "main": "src/main.ts",
4. "polyfills": "src/polyfills.ts",
5. "tsConfig": "tsconfig.app.json",
6. "assets": [
7. "src/favicon.ico",
8. "src/assets"

24 ],

1. "styles": [
2. "./node\_modules/@angular/material/prebuilt- themes/indigo-pink.css",
3. "src/styles.css" 28 ],

29 "scripts": [] 30 },

1. "configurations": {
2. "production": {
3. "budgets": [

34 {

1. "type": "initial",
2. "maximumWarning": "1mb",
3. "maximumError": "2mb" 38 },

39 {

1. "type": "anyComponentStyle",
2. "maximumWarning": "2kb",
3. "maximumError": "4kb" 43 }

44 ],

45 "fileReplacements": [ 46 {

1. "replace": "src/environments/environment.ts",
2. "with": "src/environments/environment.prod.ts" 49 }

50 ],

51 "outputHashing": "all" 52 },

1. "development": {
2. "buildOptimizer": false,
3. "optimization": false,
4. "vendorChunk": true,
5. "extractLicenses": false,
6. "sourceMap": true,
7. "namedChunks": true 60 }

61 },

62 "defaultConfiguration": "production" 63 },

1. "serve": {
2. "builder": "@angular-devkit/build-angular:dev-server",
3. "configurations": {
4. "production": {
5. "browserTarget": "Digit\_Recognizer:build:production"

69 },

1. "development": {
2. "browserTarget": "Digit\_Recognizer:build:development"

72 }

73 },

74 "defaultConfiguration": "development" 75 },

1. "extract-i18n": {
2. "builder": "@angular-devkit/build-angular:extract- i18n",
3. "options": {
4. "browserTarget": "Digit\_Recognizer:build" 80 }

81 },

1. "test": {
2. "builder": "@angular-devkit/build-angular:karma",
3. "options": {
4. "main": "src/test.ts",
5. "polyfills": "src/polyfills.ts",
6. "tsConfig": "tsconfig.spec.json",
7. "karmaConfig": "karma.conf.js",
8. "assets": [
9. "src/favicon.ico",
10. "src/assets"

92 ],

1. "styles": [
2. "./node\_modules/@angular/material/prebuilt- themes/indigo-pink.css",
3. "src/styles.css" 96 ],

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 97 "scripts": [] 98 }  99 }, | | | | | |
| 100 |  |  |  |  | "deploy": { |
| 101 |  |  |  |  | "builder": "angular-cli-ghpages:deploy" |
| 102 |  |  |  |  | } |
| 103 |  |  |  | } |  |
| 104 |  |  | } |  |  |
| 105 |  | } |  |  |  |
| 106 | } |  |  |  |  |
| **Links** |  |  |  |  |  |

[**Github Link**](https://github.com/IBM-EPBL/IBM-Project-12406-1659450617)

[**Demo Link**](https://youtu.be/odeMxKg9s_o)

[**Deployed angular UI link**](https://harshathm.github.io/Digit_Recognizer/)