

# **Professional Readiness for Innovation, Employability, and Entrepreneurship**

## **PROJECT REPORT**

-

**Title** : Detect Parkinson's disease  
**Team ID** : PNT2022TMID01470  
**Industry mentor** : Prof Swetha  
**Faculty mentor** : Ganagavalli K  
**Team Lead** : BENIGA W H (7376191CS139)  
**Members** : BALASUBRAMANIAM P (7376191CS137)  
DARSHINI R (7376191CS144)  
SHERINE BENITTA A (7376192IT233)

## **TABLE OF CONTENTS**

<b>1. INTRODUCTION.....</b>	<b>1</b>
1.1 Project Overview .....	1
1.2 Purpose.....	1
<b>2. LITERATURE SURVEY .....</b>	<b>2</b>
2.1 Existing problem .....	2
2.2 References .....	2
2.3 Problem Statement Definition.....	2
<b>3. IDEATION &amp; PROPOSED SOLUTION.....</b>	<b>3</b>
3.1 Empathy Map Canvas .....	
3.2 Ideation & Brainstorming.....	
3.3 Proposed Solution.....	
3.4 Problem Solution fit .....	
<b>4. REQUIREMENT ANALYSIS.....</b>	<b>8</b>
4.1 Functional requirement .....	
4.2 Non-Functional requirements.....	
<b>5. PROJECT DESIGN.....</b>	<b>9</b>
5.1 Data Flow Diagrams .....	
5.2 Solution & Technical Architecture .....	
5.3 User Stories.....	
<b>6. PROJECT PLANNING &amp; SCHEDULING.....</b>	<b>10</b>
6.1 Sprint Planning & Estimation .....	
6.2 Sprint Delivery Schedule .....	
6.3 Project Tracker.....	
6.4 Burndown chat.....	
6.5 Reports from JIRA .....	
<b>7. CODING &amp; SOLUTIONING .....</b>	<b>12</b>
7.1 Creating IBM Watson Studio for AI models.....	
<b>8. TESTING.....</b>	<b>15</b>
8.1 Test Cases Scenarios .....	
8.2 User Acceptance Testing.....	
8.3 UAT Report.....	
<b>9. RESULTS .....</b>	<b>16</b>
9.1 Performance Metrics .....	
<b>10. CONCLUSION .....</b>	<b>18</b>
<b>11. APPENDIX .....</b>	<b>20</b>
Source Code.....	
GitHub & Project Demo Link.....	29

## **1. INTRODUCTION**

## **1.1 Project Overview**

Parkinson disease is the second most common neurodegenerative disorder after Alzheimer disease. About 0.3% of the general population is affected, and the prevalence is higher among men than women, with a ratio of 1.5 to 1.0. Parkinson disease may be more common among white people than those of Asian or African descent.

## **1.2 Purpose**

To build a dashboard that helps the user to check whether he/she is affected by the Parkinson's Disease. The user have to enter their name, age and also have to upload the hand drawn image of a spiral or wave to predict whether he/she is affected. This helps the user to get the results quickly.

## **2. LITERATURE SURVEY**

### **2.1 Existing Problems**

#### **a) “ Early Identification of Parkinson’s Disease from Hand-drawn Images using Histogram of Oriented Gradients and Machine Learning Techniques ”**

To arrange more data from real hand-drawn tests, and to apply deep learning techniques to get a better solution . The Parkinson’s disease is predicted from the hand-drawn picture.

#### **b) “Reliable Parkinson’s Disease Detection by Analyzing Handwritten Drawings: Construction of an Unbiased Cascaded Learning System Based on Feature Selection and Adaptive Boosting Model”**

It proposes a cascaded learning system that cascades a Chi2 with adaptive boosting model. The Chi2 model ranks and selects a subset of relevant features.

#### **c) “Improvement of Feature Extraction Based on HOG”**

HOG is a common feature to describe local texture of image in the field of computer vision and pattern recognition. The feature needs to calculate the values of different gradients in the region.

#### d) “Machine Learning for the Diagnosis of Parkinson’s Disease: A Review of Literature”

To diagnose Parkinson’s disease and find the associated outcomes using ML methods.

## 2.2) PROBLEM STATEMENT

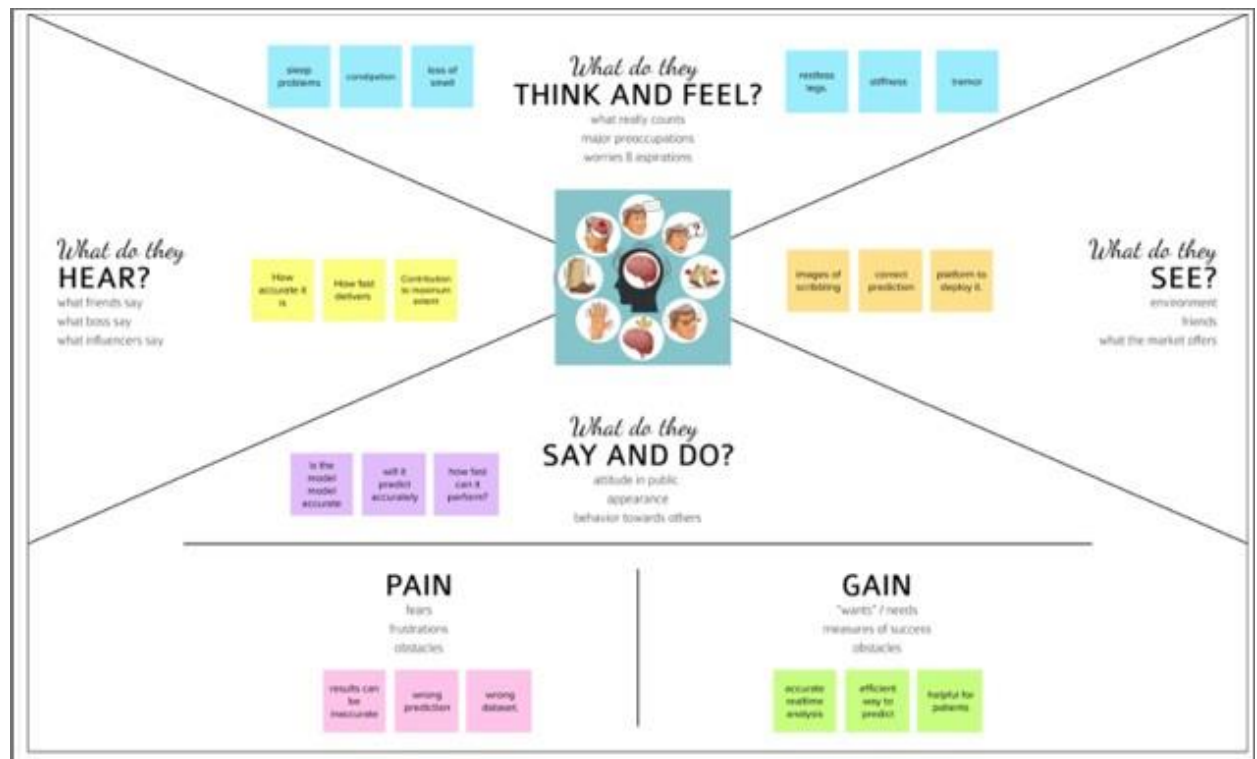
More than 10 million people are living with Parkinson’s Disease worldwide, according to the Parkinson’s Foundation. While Parkinson’s cannot be cured, early detection along with proper medication can significantly improve symptoms and quality of life.



## 3. IDEATION & PROPOSED SOLUTION

### 3.1 Empathy Map


- An empathy map is a simple, easy-to-digest visual that captures knowledge about a user’s behaviours and attitudes.
- It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it.
- The exercise of creating the map helps participants consider things from the user’s perspective along with his or her goals and challenges.



### 3.2 Ideation & Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Step-1: Team Gathering, Collaboration and Select the Problem Statement



## 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.

10 minutes to prepare  
 1 hour to collaborate  
 2-8 people recommended

[Share template feedback](#)

### Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

---

- Team gathering**  
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.
- Set the goal**  
Think about the problem you'll be focusing on solving in the brainstorming session.
- Learn how to use the facilitation tools**  
Use the Facilitation Superpowers to run a happy and productive session.  
[Open article](#)


### 1 Define your problem statement

The problem statement is to overcome the Parkinson's Disease by predicting it at the early stages by using HOG feature descriptor.

5 minutes

QUESTION

How might we [your problem statement]?



#### Key rules of brainstorming

To run as smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- It's possible, be stupid.

## Step-2: Brainstorm, Idea Listing and Grouping

### 2 Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

#### Brainstorm 1

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

#### Brainstorm 2

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

### 3 Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a semi-random label. If a cluster is bigger than an sticky notes, try and see if you can break it up into smaller sub-groups.

20 minutes

#### Working with model

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

#### Dataset

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

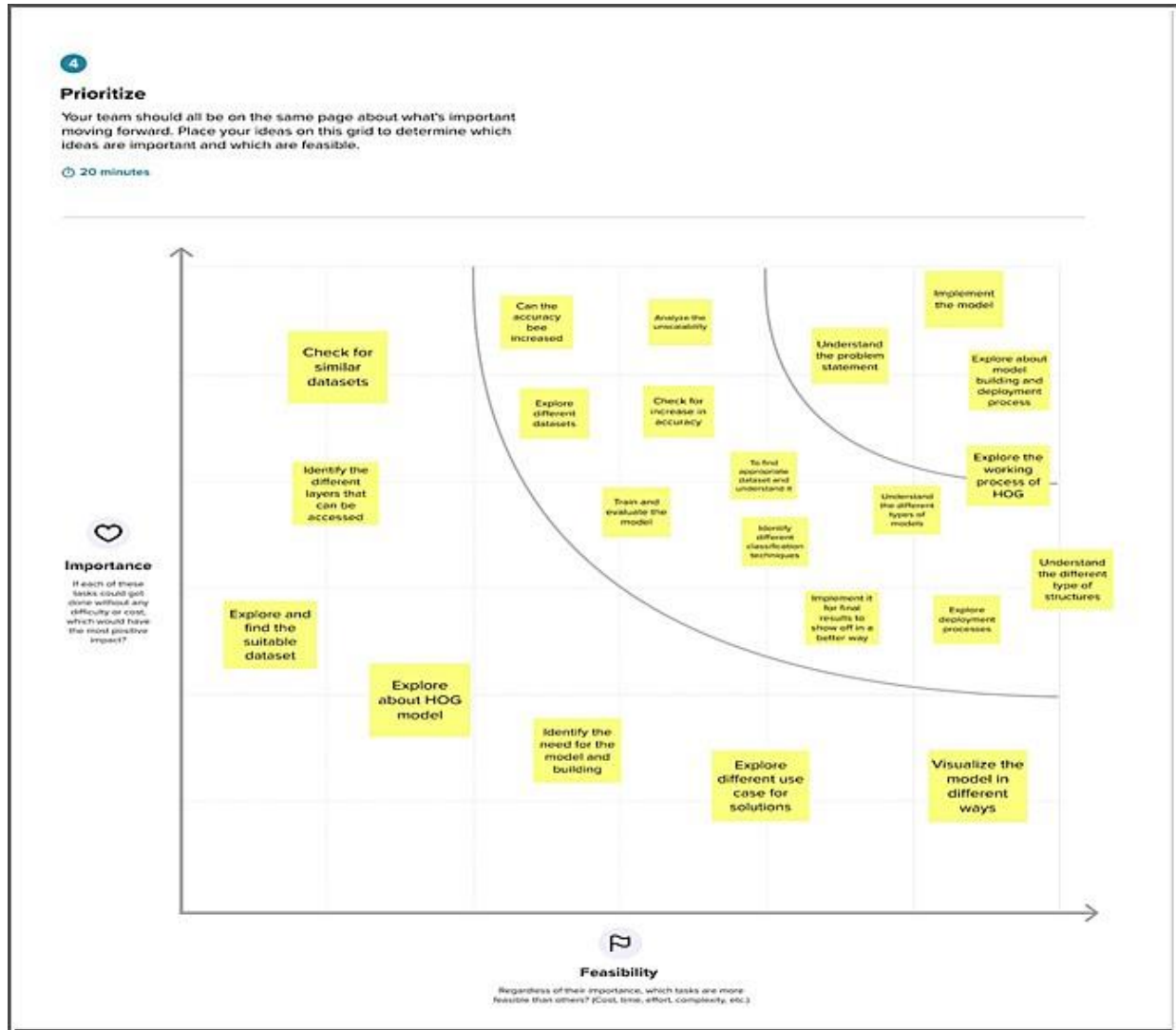
#### Visualization

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

#### Accuracy

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

### Step-3: Idea Prioritization



### 3.3 Proposed Solution

Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	More than 10 million people are living with Parkinson's disease worldwide, according to the Parkinson's Foundation. While Parkinson's cannot be cured, early detection along with proper medication can significantly improve symptoms and quality of life.
2.	Idea / Solution description	Our goal is to quantify the visual appearance (using HOG method) of these drawings and then train a machine learning model to Classify them. In this project, we are using, Histogram of Oriented Gradients (HOG) image descriptor along with a Random Forest classifier to automatically detect Parkinson's disease in hand-drawn images of spirals and waves.
3.	Novelty / Uniqueness	The entire project has been done with a fresh approach and as a change from regular kind of models. We have also tried in extending the dataset apart from the given dataset.
4.	Social Impact / Customer Satisfaction	The researchers found that the drawing speed was slower and the pen pressure is lower among Parkinson's patients. One of the indications of Parkinson's is tremors and rigidity in the muscles, making it difficult to draw smooth spirals and waves. It is possible to detect Parkinson's disease using the drawings alone instead of (measuring the speed and pressure of the pen on paper).
5.	Business Model (Revenue Model)	As this is a medical domain, it would be completely unethical to mint money out of it.
6.	Scalability of the Solution	This solution is completely scalable in all aspects, apart from being agile, simple and Glazing fast.



### 3.4 Problem Solution fit

Project Title: Detecting Parkinson's Disease using Machine Learning		Project Design Phase-I - Solution Fit Template		Team ID: PNT2022YMD18082	
Define CS, fit into CC	<b>1. CUSTOMER SEGMENT(S)</b> <span>CS</span> The user segment in mostly designed for doctors/nurse but aimed elderly people but mostly any one can use it.	<b>6. CUSTOMER CONSTRAINTS</b> <span>CC</span> As user segment in mostly designed for doctors/nurse and anyone but elderly people may not know how to use computers/stuff.	<b>5. AVAILABLE SOLUTIONS</b> <span>AS</span> There are plenty of available solutions but most of them around 80-90% of them are not user friendly.	Explore AS, differentiate	
	<b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <span>J&amp;P</span> The jobs to be done are to develop an website that is helpful and easily accessible for people of various age range. Just an upload to check if patients have Parkinson or not. One click type solutions.	<b>9. PROBLEM ROOT CAUSE</b> <span>RC</span> The major root cause medically remains mystery though researchers show it is linked with ageing and actively depression during young ages and insufficient amount of sleep.	<b>7. BEHAVIOUR</b> <span>BE</span> The users behaviour is typically to upload their hand drawn images and they expect to get reliable images.		
Focus on J&P, up into BE, understand RC	<b>3. TRIGGERS</b> <span>TR</span> Things that triggers the probability of Parkinson's to be more, their old age, confusion with inability to hold things and inability to write legibly.	<b>10. YOUR SOLUTION</b> <span>SL</span> Our solution is to come up with an not only reliable but knowing best solution an one stop platform where people can upload images of handwritters/handheld photos and with our implementation and website users can fetch results in no time, with the accuracy percentage or probability of how possible is for people to have Parkinson's.	<b>8. CHANNELS of BEHAVIOUR</b> <span>CH</span> <b>1.1 ONLINE</b> Users upload their input online and fetch results online as well. <b>1.2 OFFLINE</b> Users have to have an their inputs that must be uploaded in website.	EM & SL shows Ajruppi	
Identify strong TR & EM	<b>4. EMOTIONS: BEFORE / AFTER</b> <span>EM</span> People before detecting their Parkinson are confused, baffled anxious and unsure. But once they check themselves up they have an idea of what is going on with themselves.				

## 4. REQUIREMENT ANALYSIS

### Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Uploading Image	The data which with the further process will be carried out.
FR-2	Result Generation	With the given data the result can be generated as whether the person is healthy or affected.

### **Non-functional Requirements:**

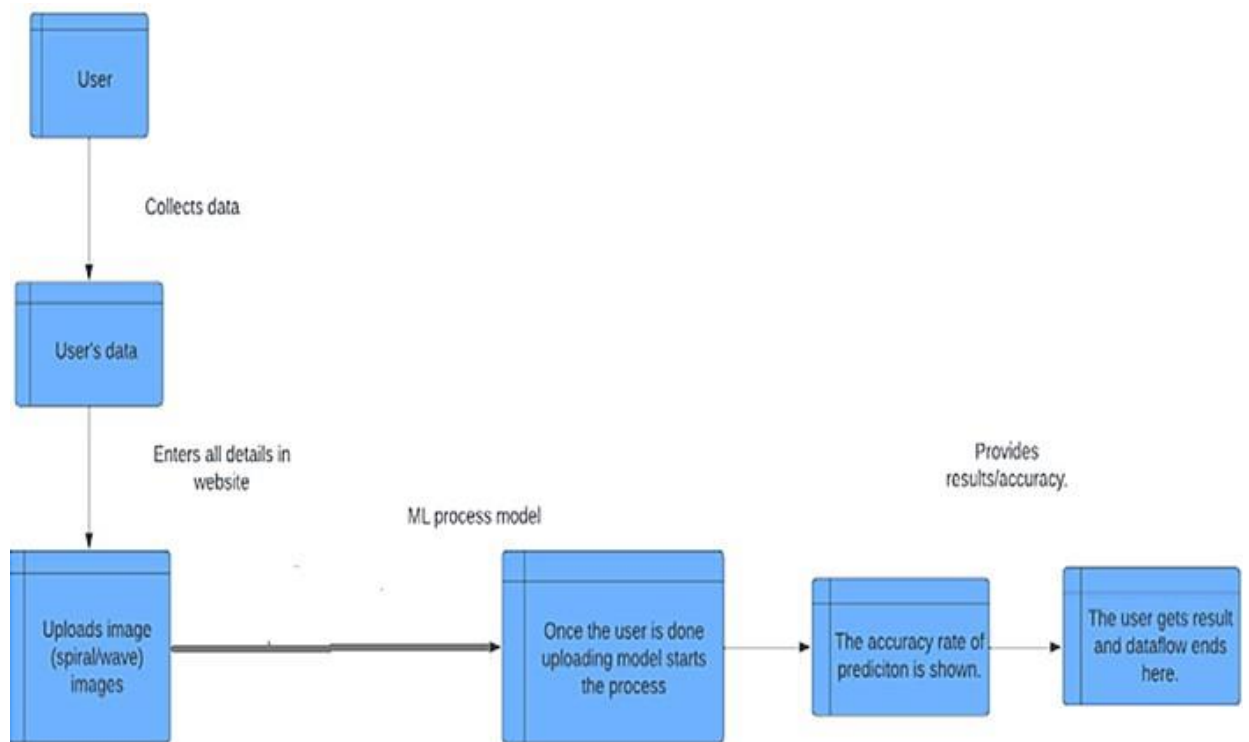
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	The user interface screen will be very much user friendly.
NFR-2	Security	The data given by the user will be very secure.
NFR-3	Reliability	Users can access the website all the time without any failure.
NFR-4	Performance	Load time for the user interface screen will be not more than 2 seconds.
NFR-5	Availability	Maximum down time will be about 4 hours.
NFR-6	Scalability	System can handle about 1000 users at any given time.

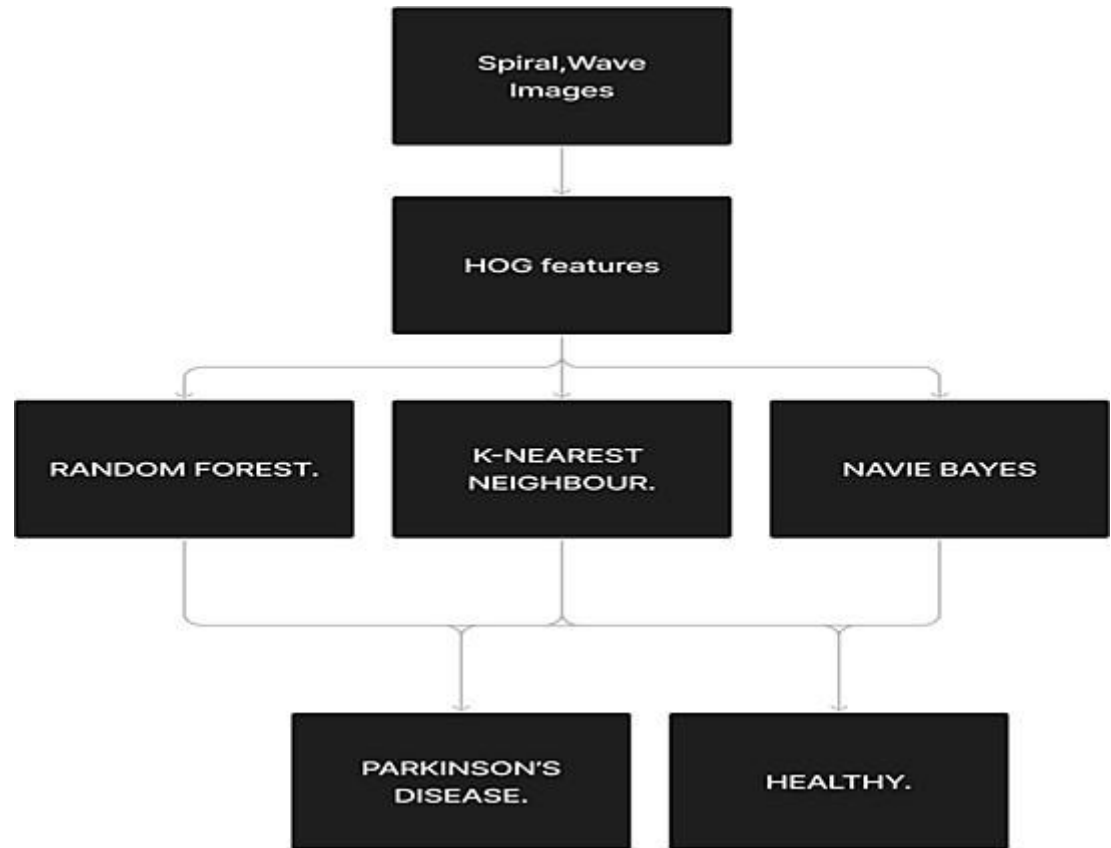
## **5. PROJECT DESIGN**

### **5.1 Data Flow Diagram**

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.



## 5.2 Solution & Technical Architecture



### 5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
User (Subject that uploads the image)	Uploads image	USN-1	As a user, I can upload the images to the website in order to obtain outcome to check for Parkinsons	I can upload any files.	High	Sprint-1
		USN-1	As a user I can view results and accuracy of the prediction as well.	I can get instant results one click away.	High	Sprint-1

## 6. PROJECT PLANNING & SCHEDULING

### 6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Home Page	USN-1	As a user, I can view the home page of the web application.	15	Low	S Yogeshwaran
Sprint-2	Data Entry	USN-2	As a user, I can enter details like images of spiral scribbling or wave scribbling.	15	Medium	S. Rajkumar
Sprint-3	Parkinson disease result display	USN-3	As a user, I can view final result whether I have Parkinson or not.	15	Medium	NS Sreevathsan
Sprint-4	Parkinson disease value Prediction	USN-4	As a user, I expect the application to predict whether I have Parkinson or not accurately.	15	Medium	A.Syed Aasim

## 6.2 Project Tracker, Velocity

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	15	6 Days	24 Oct 2022	29 Oct 2022	15	29 Oct 2022
Sprint-2	15	6 Days	31 Oct 2022	05 Nov 2022	15	05 Nov 2022
Sprint-3	15	6 Days	07 Nov 2022	12 Nov 2022	15	12 Nov 2022
Sprint-4	15	6 Days	14 Nov 2022	19 Nov 2022	15	19 Nov 2022

### Velocity

AV for sprint 1 = Total story points/ Sprint Duration =  $15/6 = 2.5$

AV for sprint 2 = Total story points/ Sprint Duration =  $15/6 = 2.5$

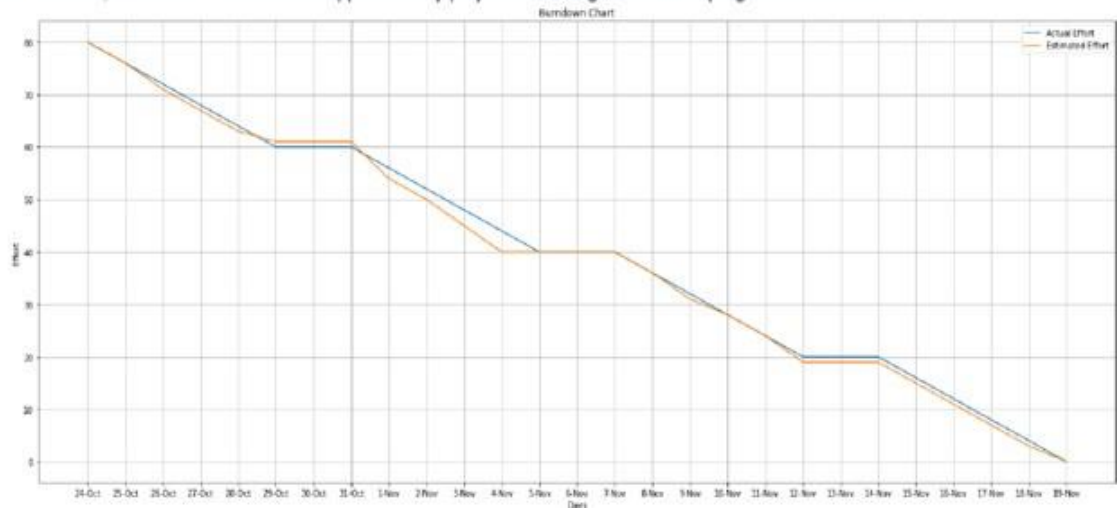
AV for sprint 3 = Total story points/ Sprint Duration =  $15/6 = 2.5$

AV for sprint 4 = Total story points/ Sprint Duration =  $15/6 = 2.5$

### Velocity Report

**Burndown Chart:**

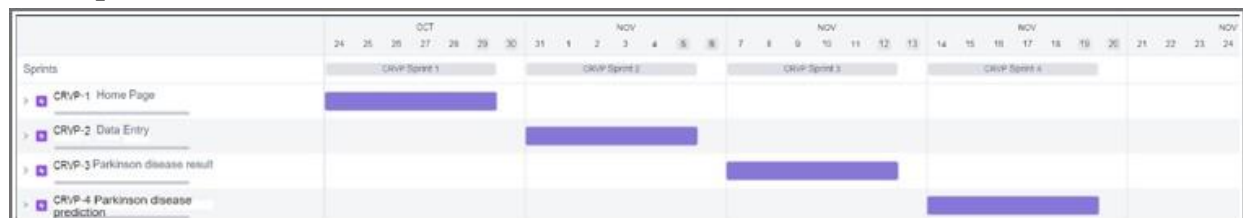
A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



**6.3 Milestone and Activity list**

Title	Description	Date
Literature Survey and Information Gathering	Gathering Information by referring the technical papers, research publications etc.	3 September 2022
Prepare Empathy Map	To capture user pain and gains Prepare List of Problem Statement	10 September 2022
Ideation	Prioritize a top 3 ideas based on feasibility and Importance	17 September 2022
Proposed Solution	Solution include novelty, feasibility, business model, social impact and scalability of solution	24 September 2022
Problem Solution Fit	Solution fit document	1 October 2022
Solution Architecture	Solution Architecture	1 October 2022
Customer Journey	To Understand User Interactions and experiences with application	8 October 2022
Functional Requirement	Prepare functional Requirement	12 October 2022
Data flow Diagrams	Data flow diagram	12 October 2022
Technology Architecture	Technology Architecture diagram	12 October 2022
Milestone & sprint delivery plan	Activity what we done &further plans	22 October 2022
Project Development- Delivery of sprint 1,2,3 &4	Develop and submit the developed code by testing it	24 October 2022 – 19 November 2022

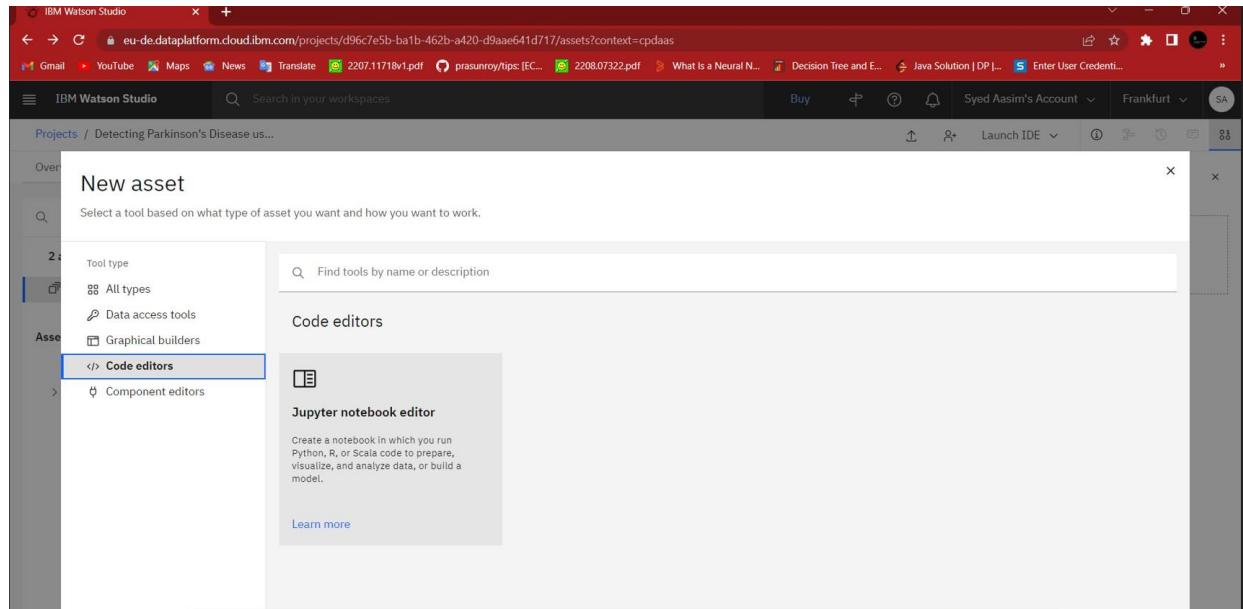
## 6.4 Reports from JIRA



## 7. CODING AND SOLUTIONS:

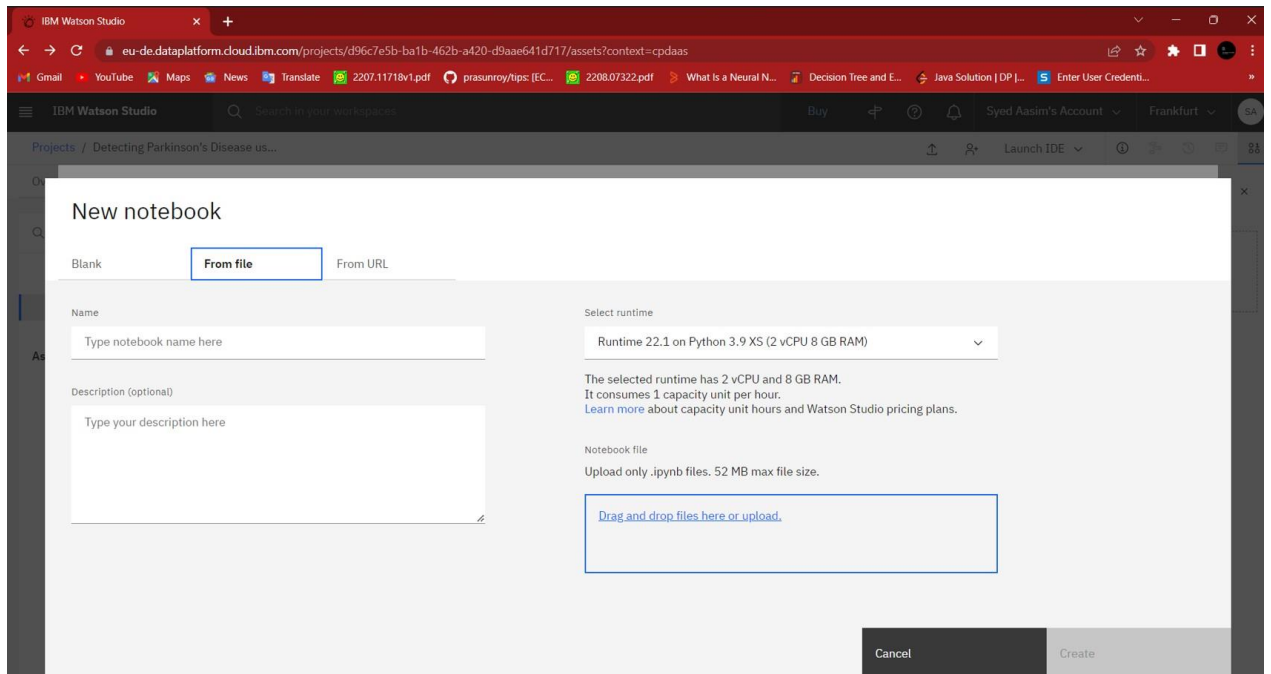
### USE IBM WATSON STUDIO AND ADD NOTEBOOKS TO GET RESULTS:

#### 1. Create notebook or upload it.

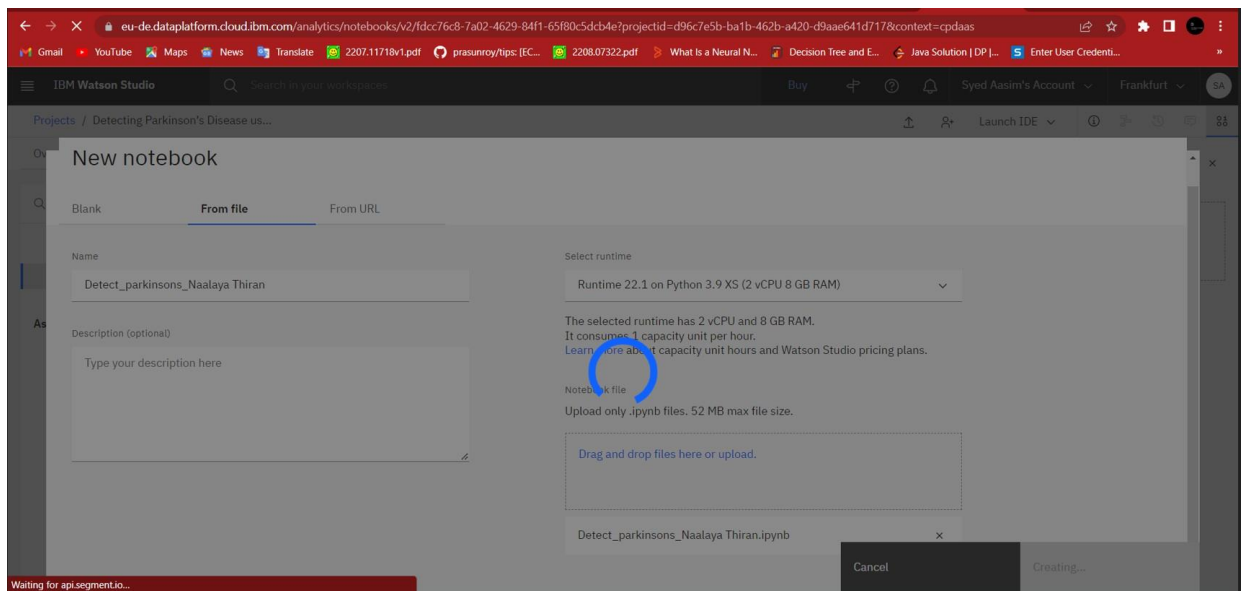


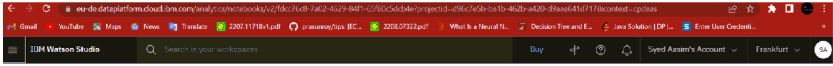
#### 2. Upload the notebook to get results.





### 3. Run the notebook and get results.





15%

Instantiating runtime for Detect\_parkinsons\_Naalaya Thiran

The selected runtime has 2 vCPU and 8 GB RAM.  
It consumes 1 capacity unit per hour.



87%

Instantiating runtime for Detect\_parkinsons\_Naalaya Thiran

It consumes 1 capacity unit per hour.



Detect\_parkinsons\_Naalaya Thir... x +

eu-de.dataplatform.cloud.ibm.com/analytic/notebooks/v2/ldcc76cb-7a02-4629-84f1-65f80c5dcb4e?projectId=d96c7e5b-ba1b-462b-a420-d9aae641d717&context=cpdaas

IBM Watson Studio

Projects / Detecting Parkinson's Disease us... / Detect\_parkinsons\_Naalaya Thir...

File Edit View Insert Cell Kernel Help

Run Format Code

```
image = cv2.threshold(image , 0 , 255,cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
features = quantify_image(image)
preds = dtree.predict([features])
label = le.inverse_transform(preds)[0]
color = (0 , 255 , 0) if label == "healthy" else (0 , 0 , 255)
cv2.putText(output , label , (3, 20) , cv2.FONT_HERSHEY_SIMPLEX , 0.5,color , 2)
images.append(output)
```

In [26]:

```
montage = build_montages(images , (128,128) ,(3,3))[0]
cv2.imshow("OUTPUT",montage)
cv2.waitKey(0)
```

Out[26]: -1

In [15]:

```
# RANDOM FOREST
predictions = model.predict(X_test)
cm = confusion_matrix(y_test , predictions).flatten()
print(cm)
(tn , fp , fn , tp) = cm
accuracy = (tp + tn) / float(cm.sum())
print(accuracy)
```

```
[14  1  3 12]
0.8666666666666667
```

In [16]:

```
# KNN
predictions = knn.predict(X_test)
cm = confusion_matrix(y_test , predictions).flatten()
print(cm)
(tn , fp , fn , tp) = cm
```

eu-de.dataplatform.cloud.ibm.com/analytic/notebooks/v2/ldcc76cb-7a02-4629-84f1-65f80c5dcb4e?projectId=d96c7e5b-ba1b-462b-a420-d9aae641d717&context=cpdaas

IBM Watson Studio

Projects / Detecting Parkinson's Disease us... / Detect\_parkinsons\_Naalaya Thir...

File Edit View Insert Cell Kernel Help

Run Format Code

```
image = cv2.threshold(image , 0 , 255,cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
features = quantify_image(image)
preds = dtree.predict([features])
label = le.inverse_transform(preds)[0]
color = (0 , 255 , 0) if label == "healthy" else (0 , 0 , 255)
cv2.putText(output , label , (3, 20) , cv2.FONT_HERSHEY_SIMPLEX , 0.5,color , 2)
images.append(output)
```

In [26]:

```
montage = build_montages(images , (128,128) ,(3,3))[0]
cv2.imshow("OUTPUT",montage)
cv2.waitKey(0)
```

Out[26]: -1

In [15]:

```
# RANDOM FOREST
predictions = model.predict(X_test)
cm = confusion_matrix(y_test , predictions).flatten()
print(cm)
(tn , fp , fn , tp) = cm
accuracy = (tp + tn) / float(cm.sum())
print(accuracy)
```

```
[14  1  3 12]
0.8666666666666667
```

In [16]:

```
# KNN
predictions = knn.predict(X_test)
cm = confusion_matrix(y_test , predictions).flatten()
print(cm)
(tn , fp , fn , tp) = cm
```

24°C Partly sunny

Search

ENG IN

08:35 22-11-2022

## 8. TESTING

<b>Test Scenarios no.</b>	<b>Test Scenarios</b>
<b>TS_001</b>	<b>Verify user is able to see the chatbot icon when website is launched</b>
<b>TS_002</b>	<b>Verify the UI elements in WEB app and user is clear of the options</b>
<b>TS_003</b>	<b>Verify user is able to see the “enter name:”</b>
<b>TS_004</b>	<b>Verify user is able to see the “Enter age:”</b>
<b>TS_005</b>	<b>Verify user is able to see the “Upload file/image”</b>
<b>TS_006</b>	<b>Verify users are able to type query in text fields.</b>
<b>TS_007</b>	<b>Verify users are able to upload images in the upload option.</b>
<b>TS_008</b>	<b>Verify user is able to get the response from web app</b>
<b>TS_009</b>	<b>Verify the user whether gets the appropriate response as per the users requirements.</b>
<b>TS_010</b>	<b>Verify user whether gets the response if the user enters the query with some error.</b>

Test case ID	Feature Type	Component	Test Scenario	Pre-Requlite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
Webapp_TC_001	Functional	Home Page	Verify user is able to use the Textfields and upload options		1. Enter URL and click go 2. Verify home page displayed or not	Users uploaded images	Parkinsons detection webapp icon should display.	Working as expected	Pass	Nil	N	-	Yogeshwaran, rajkumar
Webapp_TC_002	UI	Home Page	Verify the User enters in Parkinsons detection webapp icon popup		1. Enter URL and click go 2. Verify home page displayed	Users uploaded images	After 30 seconds Parkinsons detection webapp pop out with asking for any suggestion	Working as expected	Pass	Nil	N	-	Sreevathsan
Webapp_TC_003	Functional	Home page	Verify user is able to use the "enter name"		1. Enter URL and click go 2. Verify Parkinsons detection webapp icon popup displayed 3. Verify whether user able to enter age	Users uploaded images	User should see the greeting message from Parkinsons detection webapp	Working as expected	Pass	Nil	N	-	Syed asim
Webapp_TC_004	Functional	Parkinsons detection webapp	Verify user is able to use the "Enter age"		1. Enter URL and click go 2. Verify Parkinsons detection webapp icon popup displayed 3. Verify whether user able to enter age	Users uploaded images	User able to receive dynamic greeting message	Working as expected	Pass	Nil	N	-	Yogeshwaran, Rajkumar, Syed, sreevathsan
Webapp_TC_005	Functional	Parkinsons detection webapp	Verify user is able to use the "Upload file/image"	Suggestion of action by Parkinsons detection webapp	1. Enter URL and click go 2. Verify Parkinsons detection webapp icon popup displayed 3. Verify whether user able to enter /upload files	Users uploaded images	User able to select the action suggested by Parkinsons detection webapp	Working as expected	Pass	Nil	N	-	Syed, sreevathsan
Webapp_TC_006	Functional	Parkinsons detection webapp	Verify users are able to type query in text field.		1. Enter URL and click go 2. Verify Parkinsons detection webapp icon popup displayed 3. Verify whether user able to type query in text field or not.	Users uploaded images	User able to type the query in text field.	Working as expected	Pass	Nil	N	-	Sreevathsan, rajkumar
Webapp_TC_007	Functional	Parkinsons detection webapp	Verify users are able to upload images in the upload option.	Question is required	1. Enter URL and click go 2. Verify Parkinsons detection webapp icon 3. Verify whether user is able to type query in text field or not. 4. Verify whether the user gets the response even if the user types the wrong or unrelated query also	Users uploaded images	User get the response from webapp	Working as expected	Pass	Nil	N	-	Syed
Webapp_TC_008	Functional	Parkinsons detection webapp	Verify user is able to get the response from web app.	Question is required	1. Enter URL and click go 2. Verify Parkinsons detection webapp icon 3. Verify whether user is able to type query in text field or not. 4. Verify whether the user gets the response even if the user types the wrong or unrelated query also	Users get desired output	User get the response from webapp	Working as expected	Pass	Parkinsons detection webapp will respond as it doesn't have any idea about that	N	-	Sreevathsan, Yogeshwaran
Webapp_TC_009	Functional	Parkinsons detection webapp	Verify the user whether gets the appropriate response as per the users requirements.		1. Enter URL and click go 2. Verify Parkinsons detection webapp 3. Verify when clicking that icon, it shows greeting. 4. Verify whether user is able to type query in text field or not. 5. Verify whether the user gets the response even if the user types the wrong or unrelated query also	Users get desired output	User get the response from webapp	Working as expected	Pass	Nil	N	-	Yogeshwaran
Webapp_TC_010	Functional	Parkinsons detection webapp	Verify user whether gets the response if the user enters the query with some error.		1. Enter URL and click go 2. Verify Parkinsons detection webapp 3. Verify when clicking that icon, it shows greeting. 4. Verify whether user is able to type query in text field or not. 5. Verify whether the user gets the response even if the user types the wrong or unrelated query also	Users get desired output	User get the response from webapp	Working as expected	Pass	Nil	N	-	Syed, rajkumar

## 8.1 UAT:

The purpose of this document is to briefly explain the test coverage and open issues of the Detect Parkinson's disease project at the time of the release to User Acceptance Testing (UAT)

## 2. 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	3	2	2	1	8
Duplicate	2	0	3	0	5
External	1	0	0	1	2
Fixed	3	2	2	1	8
Not Reproduced	0	0	1	0	1
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	9	4	8	3	24

### 3. Test Case Analysis

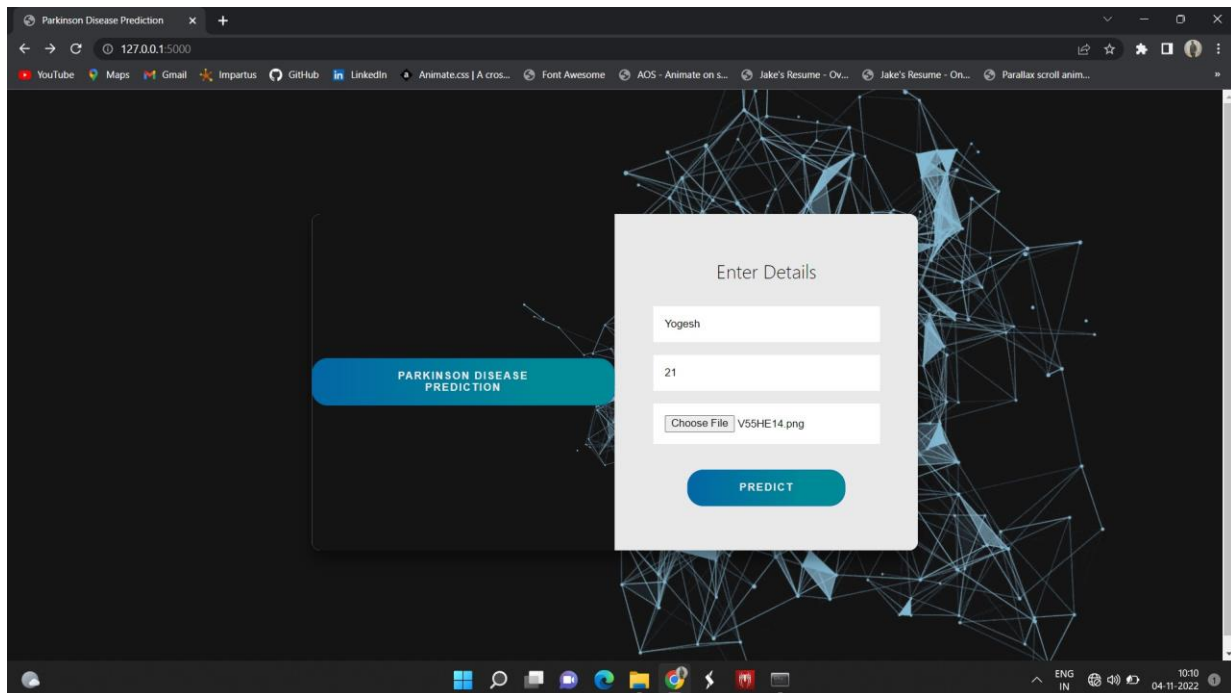
This report shows the number of test cases that have passed, failed, and untested

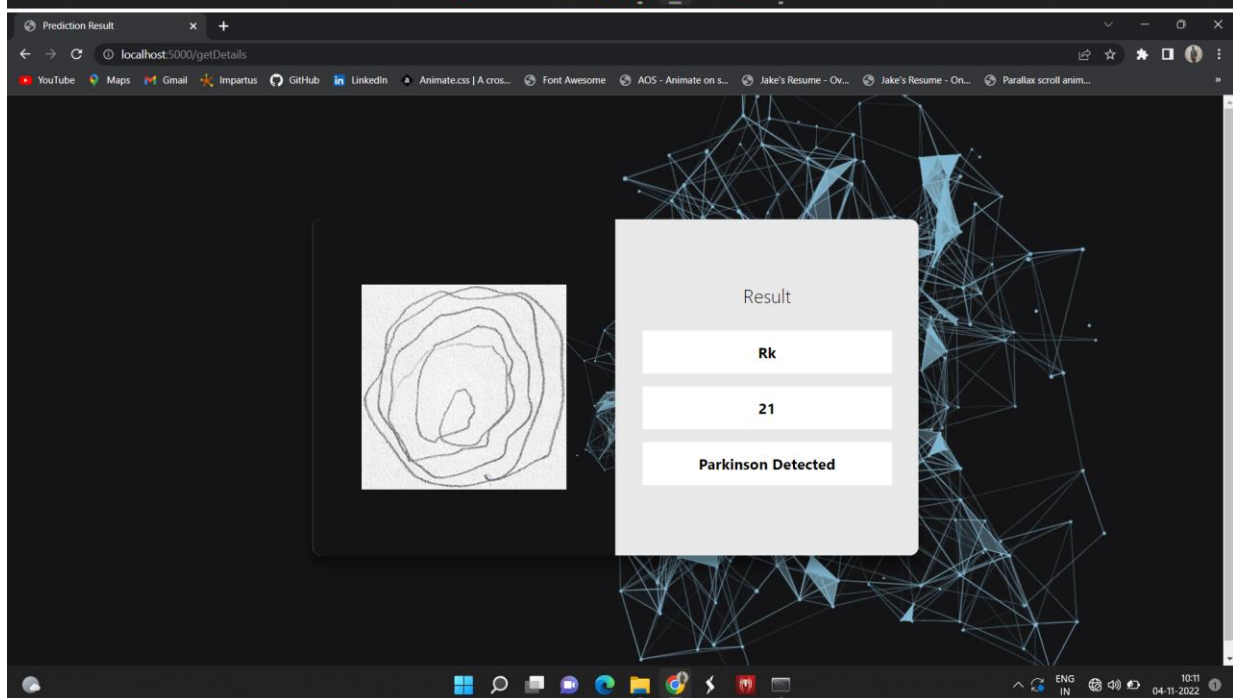
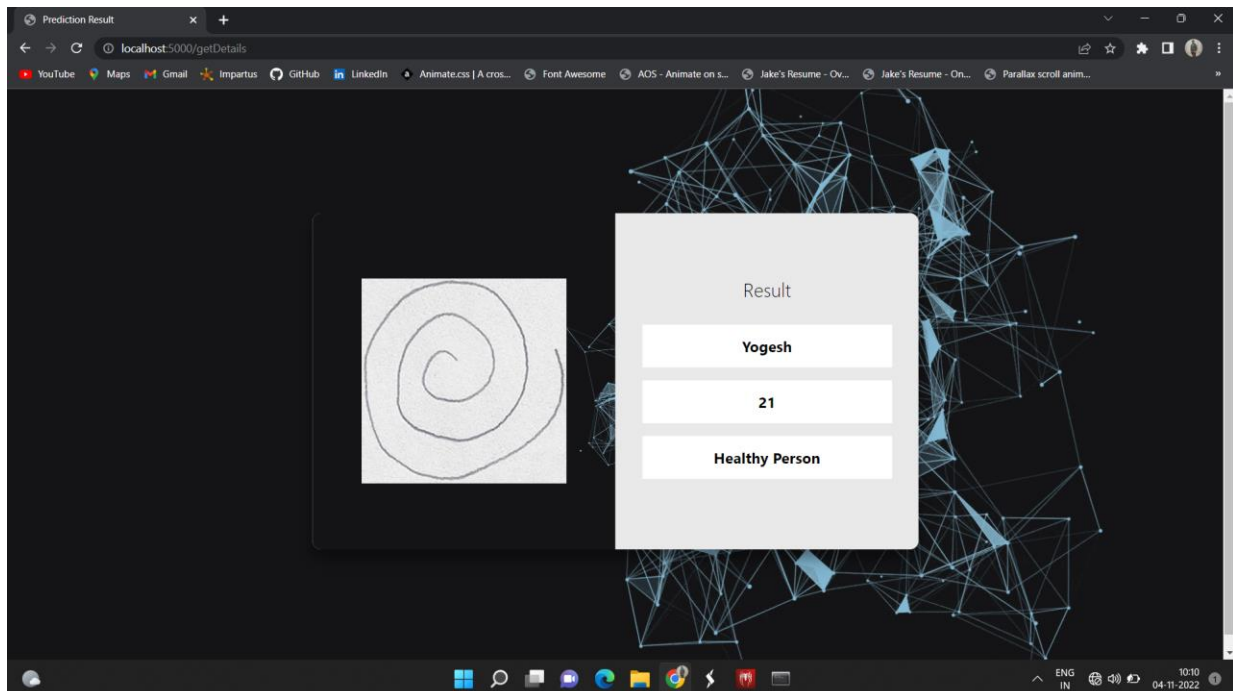
Section	Total Cases	Not Tested	Fail	Pass
Home Page	3	0	0	3
Data Entry Page	2	0	0	2
Output Page	2	0	0	2

## 9.RESULTS

### 9.1 Performance Metrics

For the Ai-based discourse for Project - Detect Parkinson's disease we build AN model for enabling to achieve results that help in predicting the precsence of parkinsons on patient and further help in the advancement of this field . We utilized the IBM service (Watson Studio) to provide accurate results.





## 10. CONCLUSION

Thus from the above given procedures and resources that were available we were able to build and develop and in the process learn a ton of things to create a prediction model that provides a very accurate result that helps in Parkinson's patients and in the field of medical research as well.

## 11. APPENDIX

### SOURCE CODE:

#### Hello.py

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import requests
import numpy as np
import cv2
from skimage import feature
import os
from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename
import pickle
from sklearn.preprocessing import LabelEncoder

app = Flask(__name__)
app.config['UPLOAD_PATH'] = 'static/uploads'

#le = LabelEncoder()
model = pickle.load(open('parkinsons.pkl', 'rb'))

def quantify_image(image):
    # compute the histogram
    features = feature.hog(image, orientations=9,
        pixels_per_cell=(10, 10), cells_per_block=(2, 2),
        transform_sqrt=True, block_norm="L1")
    # return the feature vector
    return features

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/getDetails', methods = ['POST'])
def getDetails():
    img=request.files['Image']
```



```

name=request.form['Name']
age=request.form['Age']
print(img.filename)
filename = secure_filename(img.filename)
img.save(os.path.join(app.config['UPLOAD_PATH'], filename))
print(os.path.join(app.config['UPLOAD_PATH'], filename))
image = cv2.imread(os.path.join(app.config['UPLOAD_PATH'], filename))
output = image.copy()
output = cv2.resize(output , (128,128))
image = cv2.cvtColor(image , cv2.COLOR_BGR2GRAY)
image = cv2.resize(image , (200,200))
image = cv2.threshold(image , 0 , 255,cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
features = quantify_image(image)
print(features)
preds = model.predict([features])
print(preds)
if preds[0]==1:
    return render_template('predictionResult.html',Name=name,Age=age,Result=" Parkinson Detected
",imageName= filename)
else:
    return render_template('predictionResult.html',Name=name,Age=age,Result=" Healthy Person ",imageName=
filename)
if __name__ == '__main__':
    app.run("127.0.0.1",5000)

```

## Index.html

```

<html>
<head>
<title>
    Parkinson Disease Prediction
</title>
<link rel="stylesheet" type="text/css" href="{ { url_for('static',filename='styles/styles.css') } }">
<body>
<div class="container right-panel-active">
<!-- Sign Up -->
<div class="container_form container--signup">
<form action="http://localhost:5000/getDetails" method="POST" enctype="multipart/form-data"
class="form" id="form1">
<h2 class="form_title">Enter Details</h2>
<input type="text" placeholder="Name" class="input" name="Name"/>
<input type="text" placeholder="Age" class="input" name="Age"/>
<input type="file" placeholder="Image" class="input" name="Image"/>
<button class="btn">Predict</button>
</form>
</div>

<!-- Overlay -->

```

```

    <div class="container_overlay">
      <div class="overlay">
        <div class="overlay_panel overlay--left">
          <button class="btn" id="signIn">Parkinson Disease Prediction</button>
        </div>
      </div>
    </div>
  </div>
</body>
</head>
</html>

```

```

<html>
  <head>
    <title>
      Prediction Result
    </title>
    <link rel= "stylesheet" type= "text/css" href= "{{ url_for('static',filename='styles/styles.css') }}">
    <body>
      <div class="container right-panel-active">
        <!-- Sign Up -->
        <div class="container_form container--signup">
          <div class="form" id="form1">
            <h2 class="form__title">Result</h2>
            <h3 class="input"> {{ Name }} </h3>
            <h3 class="input"> {{ Age }} </h3>
            <h3 class="input"> {{ Result }} </h3>
          </div>
        </div>

        <!-- Overlay -->
        <div class="container_overlay">
          <div class="overlay">
            <div class="overlay_panel overlay--left">
              
            </div>
          </div>
        </div>
      </div>
    </body>
  </head>
</html>

```

## STYLES.CSS

```

:root {
  /* COLORS */
  --white: #e9e9e9;

```

```

--gray: #333;
--blue: #0367a6;
--lightblue: #008997;

/* RADII */
--button-radius: 0.7rem;

/* SIZES */
--max-width: 758px;
--max-height: 420px;

font-size: 16px;
font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Oxygen,
            Ubuntu, Cantarell, "Open Sans", "Helvetica Neue", sans-serif;
}

body {
  align-items: center;
  background-color: var(--white);
  background: url("./Nerves.gif");
  background-attachment: fixed;
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
  display: grid;
  height: 100vh;
  place-items: center;
}

.form_title {
  font-weight: 300;
  margin: 0;
  margin-bottom: 1.25rem;
}

.link {
  color: var(--gray);
  font-size: 0.9rem;
  margin: 1.5rem 0;
  text-decoration: none;
}

.container {
  background-color: var(--white);
  border-radius: var(--button-radius);
  box-shadow: 0 0.9rem 1.7rem rgba(0, 0, 0, 0.25),
            0 0.7rem 0.7rem rgba(0, 0, 0, 0.22);

```

```
        height: var(--max-height);
        max-width: var(--max-width);
        overflow: hidden;
        position: relative;
        width: 100%;
    }

    .container_form {
        height: 100%;
        position: absolute;
        top: 0;
        transition: all 0.6s ease-in-out;
    }

    .container--signin {
        left: 0;
        width: 50%;
        z-index: 2;
    }

    .container.right-panel-active .container--signin {
        transform: translateX(100%);
    }

    .container--signup {
        left: 0;
        opacity: 0;
        width: 50%;
        z-index: 1;
    }

    .container.right-panel-active .container--signup {
        animation: show 0.6s;
        opacity: 1;
        transform: translateX(100%);
        z-index: 5;
    }

    .container_overlay {
        height: 100%;
        left: 50%;
        overflow: hidden;
        position: absolute;
        top: 0;
        transition: transform 0.6s ease-in-out;
        width: 50%;
        z-index: 100;
    }
```

```
.container.right-panel-active .container_overlay {  
    transform: translateX(-100%);  
}
```

```
.overlay {  
    background-color: var(--lightblue);  
    background: url("../Nerves.gif");  
    background-attachment: fixed;  
    background-position: center;  
    background-repeat: no-repeat;  
    background-size: cover;  
    height: 100%;  
    left: -100%;  
    position: relative;  
    transform: translateX(0);  
    transition: transform 0.6s ease-in-out;  
    width: 200%;  
}
```

```
.container.right-panel-active .overlay {  
    transform: translateX(50%);  
}
```

```
.overlay_panel {  
    align-items: center;  
    display: flex;  
    flex-direction: column;  
    height: 100%;  
    justify-content: center;  
    position: absolute;  
    text-align: center;  
    top: 0;  
    transform: translateX(0);  
    transition: transform 0.6s ease-in-out;  
    width: 50%;  
}
```

```
.overlay--left {  
    transform: translateX(-20%);  
}
```

```
.container.right-panel-active .overlay--left {  
    transform: translateX(0);  
}
```

```
.overlay--right {  
    right: 0;
```

```
        transform: translateX(0);
    }

.container.right-panel-active .overlay--right {
    transform: translateX(20%);
}

.btn {
    background-color: var(--blue);
    background-image: linear-gradient(90deg, var(--blue) 0%, var(--lightblue) 74%);
    border-radius: 20px;
    border: 1px solid var(--blue);
    color: var(--white);
    cursor: pointer;
    font-size: 0.8rem;
    font-weight: bold;
    letter-spacing: 0.1rem;
    padding: 0.9rem 4rem;
    text-transform: uppercase;
    transition: transform 80ms ease-in;
}

.form > .btn {
    margin-top: 1.5rem;
}

.btn:active {
    transform: scale(0.95);
}

.btn:focus {
    outline: none;
}

.form {
    background-color: var(--white);
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
    padding: 0 3rem;
    height: 100%;
    text-align: center;
}

.input {
    background-color: #fff;
    border: none;
```

```
padding: 0.9rem 0.9rem;
margin: 0.5rem 0;
width: 100%;
}
```

```
@keyframes show {
    0%,
    49.99% {
        opacity: 0;
        z-index: 1;
    }

    50%,
    100% {
        opacity: 1;
        z-index: 5;
    }
}
```

## Detected-parkinsons.ipynb

```
!pip install imutils
```

```
pip install opencv-python
```

```
pip install pickle-mixin
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
from skimage import feature
from imutils import build_montages
from imutils import paths
import numpy as np
import cv2
import os
import pickle
```

```
trainingPath=r"D:\STUDIES\Project\Nalaya Thiran Dataset\dataset-20220920T082711Z-001\dataset\spiral\training"
testingPath=r"D:\STUDIES\Project\Nalaya Thiran Dataset\dataset-20220920T082711Z-001\dataset\spiral\testing"
```

```
def load_split ( path ) :
    #grab the list of images in the input directory , then initialize
    # the list of data ( i.e. , images ) and class labels
    imagePath = list ( paths.list_images ( path ) )
    data = [ ]
    labels = [ ]
```

```

#loop over the image paths
for imagePath in imagePaths :
    #extract the class label from the filename
    label = imagePath.split ( os.path.sep ) [ -2 ]
    print(label)
    #load the input image , convert it to grayscale , and resize
    # it to 200x200 pixels , ignoring aspect ratio
    image = cv2.imread ( imagePath )
    image = cv2.cvtColor ( image , cv2.COLOR_BGR2GRAY )
    image = cv2.resize ( image , ( 200 , 200 ) )
    # threshold the image such that the drawing appears as white
    # on a black background
    image = cv2.threshold(image , 0 , 255 , cv2.THRESH_BINARY_INV | cv2 . THRESH_OTSU)[ 1 ]
    # quantify the image
    features = quantify_image(image)
    print(features)
    # update the data and labels lists , respectively
    data.append ( features )
    labels.append ( label )
# return the data and labels
return ( np.array ( data ) , np.array ( labels ) )

def quantify_image(image):
    # compute the histogram
    features = feature.hog(image, orientations=9,
        pixels_per_cell=(10, 10), cells_per_block=(2, 2),
        transform_sqrt=True, block_norm="L1")
    # return the feature vector
    return features

(X_train, y_train)=load_split(trainingPath)
(X_test,y_test)=load_split(testingPath)

# encode the labels as integers
le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)
print(X_train.shape,y_train.shape)

X_train

#RANDOM FOREST
print( " [ INFO ] training model Random Forest " )
model = RandomForestClassifier(n_estimators = 100 )
model.fit(X_train , y_train)

#KNN
from sklearn.neighbors import KNeighborsClassifier

```



```

print( " [ INFO ] training model KNN " )
knn = KNeighborsClassifier(n_neighbors=20)
knn.fit(X_train, y_train)

#DECISION TREE
from sklearn.tree import DecisionTreeClassifier
dTree = DecisionTreeClassifier(criterion="gini", random_state=42,max_depth=10, min_samples_leaf=5)
dTree.fit(X_train,y_train)

testPaths = list(paths.list_images(testingPath))
idxs = np.arange(0 , len(testPaths))
print(idxs)
idxs = np.random.choice(idxs , size=(25,),replace=False)
print(idxs)
images = []

# RANDOM FOREST
for i in idxs:
    image = cv2.imread(testPaths[i])
    print(image)
    output = image.copy()
    output = cv2.resize(output , (128,128))
    image = cv2.cvtColor(image , cv2.COLOR_BGR2GRAY)
    image = cv2.resize(image , (200,200))
    image = cv2.threshold(image , 0 , 255,cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
    print(image)
    features = quantify_image(image)
    print(features)
    preds = model.predict([features])
    label = le.inverse_transform(preds)[0]
    print(label)
    color = (0 , 255 , 0) if label == "healthy" else (0 , 0 ,255)
    cv2.putText(output , label , (3, 20) , cv2.FONT_HERSHEY_SIMPLEX , 0.5,color , 2)
    images.append(output)

#KNN
for i in idxs:
    image = cv2.imread(testPaths[i])
    output = image.copy()
    output = cv2.resize(output , (128,128))
    image = cv2.cvtColor(image , cv2.COLOR_BGR2GRAY)
    image = cv2.resize(image , (200,200))
    image = cv2.threshold(image , 0 , 255,cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
    features = quantify_image(image)
    preds = knn.predict([features])
    label = le.inverse_transform(preds)[0]
    color = (0 , 255 , 0) if label == "healthy" else (0 , 0 ,255)
    cv2.putText(output , label , (3, 20) , cv2.FONT_HERSHEY_SIMPLEX , 0.5,color , 2)

```

```

images.append(output)

#Decision Tree
for i in idxs:
    image = cv2.imread(testPaths[i])
    output = image.copy()
    output = cv2.resize(output , (128,128))
    image = cv2.cvtColor(image , cv2.COLOR_BGR2GRAY)
    image = cv2.resize(image , (200,200))
    image = cv2.threshold(image , 0 , 255,cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
    features = quantify_image(image)
    preds = dTree.predict([features])
    label = le.inverse_transform(preds)[0]
    color = (0 , 255 , 0) if label == "healthy" else (0 , 0 ,255)
    cv2.putText(output , label , (3, 20) , cv2.FONT_HERSHEY_SIMPLEX , 0.5,color , 2)
    images.append(output)

montage = build_montages(images , (128,128) ,(3,3))[0]
cv2.imshow("OUTPUT",montage)
cv2.waitKey(0)

# RANDOM FOREST
predictions = model.predict(X_test)
cm = confusion_matrix(y_test , predictions).flatten()
print(cm)
(tn , fp , fn , tp) = cm
accuracy = (tp + tn) / float(cm.sum())
print(accuracy)

# KNN
predictions = knn.predict(X_test)
cm = confusion_matrix(y_test , predictions).flatten()
print(cm)
(tn , fp , fn , tp) = cm
accuracy = (tp + tn) / float(cm.sum())
print(accuracy)

#DECISION TREE
predictions = dTree.predict(X_test)
cm = confusion_matrix(y_test , predictions).flatten()
print(cm)
(tn , fp , fn , tp) = cm
accuracy = (tp + tn) / float(cm.sum())
print(accuracy)

pickle.dump(model , open('parkinsons.pkl','wb'))

```

**LINK FOR THE DEMO VIDEO**

[https://drive.google.com/file/d/1mFhkaUf-CaMZXcpU4smAKEEyIU6fiKWd/view?usp=share link](https://drive.google.com/file/d/1mFhkaUf-CaMZXcpU4smAKEEyIU6fiKWd/view?usp=share_link)