

Final Report

Alzheimer Disease Prediction Using CNN And Datasets MRI

Team ID: 5 9 2 0 7 7

Team Members:

Dev Badodiya

Project Report Format

1. INTRODUCTION

- 1.1 Project Overview
- 1.2 Purpose

2. LITERATURE SURVEY

- 2.1 Existing problem
- 2.2 References
- 2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

- 3.1 Empathy Map Canvas
- 3.2 Ideation & Brainstorming

4. REQUIREMENT ANALYSIS

- 4.1 Functional requirement
- 4.2 Non-Functional requirements

5. PROJECT DESIGN

- 5.1 Data Flow Diagrams & User Stories
- 5.2 Solution Architecture

6. PROJECT PLANNING & SCHEDULING

- 6.1 Technical Architecture
- 6.2 Sprint Planning & Estimation
- 6.3 Sprint Delivery Schedule

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

- 7.1 Feature 1
- 7.2 Feature 2
- 7.3 Database Schema (if Applicable)

8. PERFORMANCE TESTING

- 8.1 Performance Metrics

9. RESULTS

- 9.1 Output Screenshots

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

- Source Code
- GitHub & Project Demo Link

Project Description:

Alzheimer's disease (AD) is a progressive and irreversible neurological disorder that affects the brain, leading to memory loss, cognitive impairment, and changes in behavior and personality. It is the most common cause of dementia among older adults and is characterized by the buildup of abnormal protein deposits in the brain, including amyloid plaques and tau tangles.

The exact cause of Alzheimer's disease is not yet fully understood, but it is believed to be influenced by a

combination of genetic, environmental, and lifestyle factors. Age is also a significant risk factor, with the risk of

developing the disease increasing significantly after the age of 65. The early symptoms of Alzheimer's disease

may include mild memory loss, difficulty with problem-solving, and changes in mood or behavior. As the

disease progresses, these symptoms become more severe, with individuals experiencing significant memory

loss, difficulty communicating, and a loss of the ability to perform daily activities.

By using deep learning models like Xception to analyze medical imaging data, it may be able to identify early

signs of Alzheimer's disease before symptoms become severe. This can help healthcare providers to provide

early treatment and support for patients and their families, ultimately leading to better outcomes for all involved.

Ideation Phase

Empathize & Discover

Date	13-12-2023
Team ID	PNT2023TMID592077
Project Name	Project- Alzheimer Disease Prediction
Maximum Marks	4 Marks

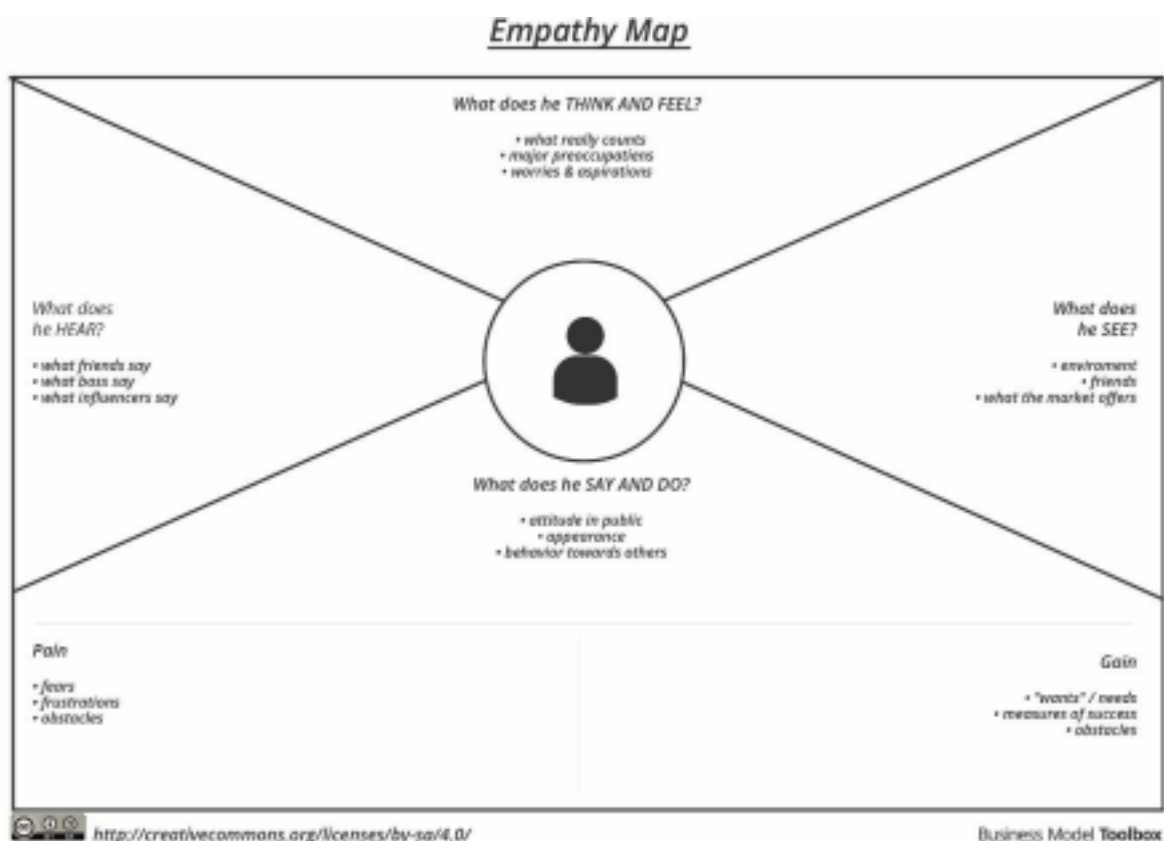
Empathy Map Canvas:

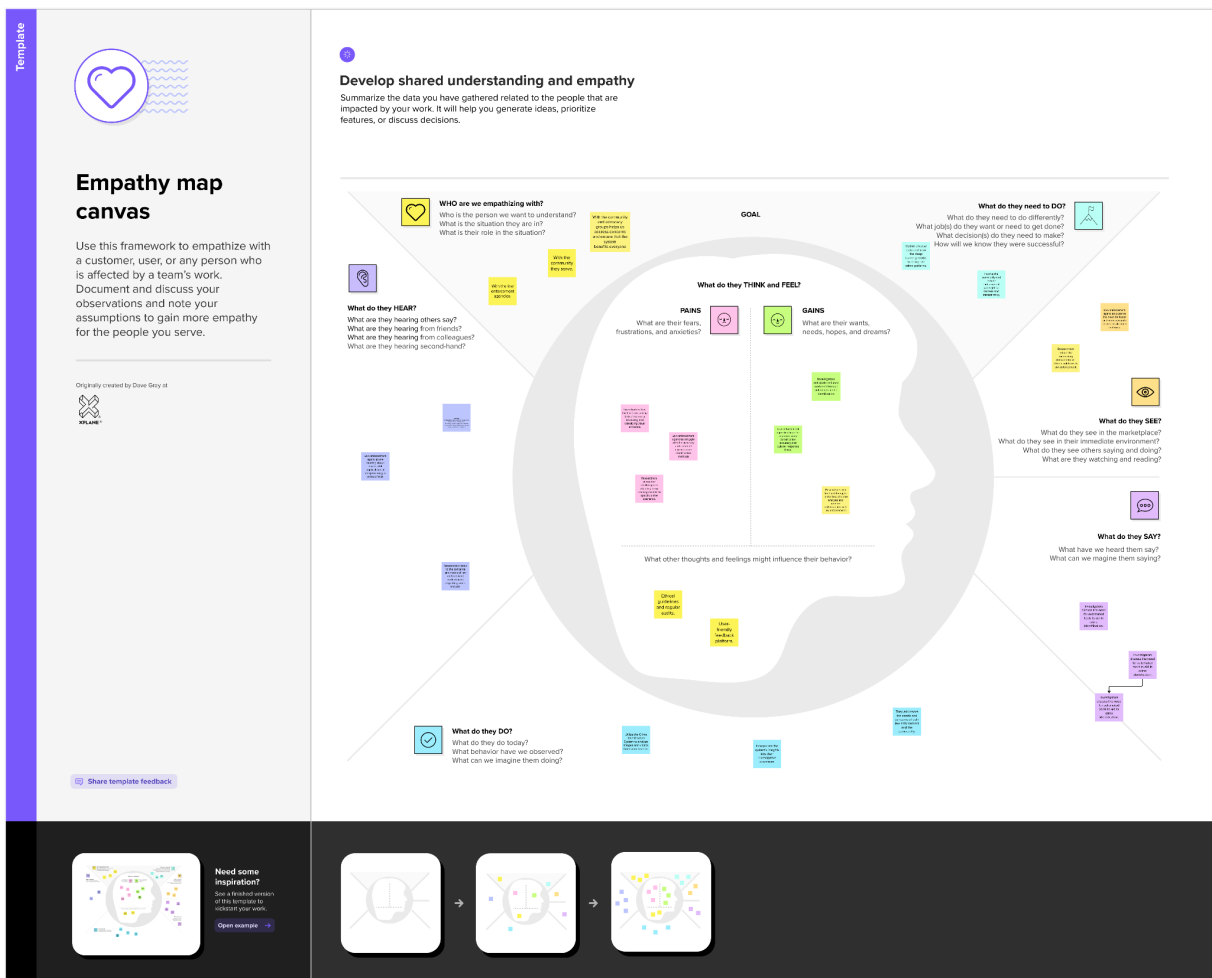
An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

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.

Example:





Reference: <https://www.mural.co/templates/empathy-map-canvas>

Ideation Phase
Brainstorm & Idea Prioritization Template

Date	13- 12-2023
Team ID	592077
Project Name	Alzheimer Disease Prediction
Maximum Marks	4 Marks

Brainstorm & Idea Prioritization Template:

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.

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.

Project Description-

This is project use CNN Datasets and Ai tech to analyze the Data of MRI to Detect the Disease .

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



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

A

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

C

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

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

"How might we develop an automated Crime Identification System, harnessing deep learning techniques, to swiftly and accurately analyze visual data from crime scenes? This system should adeptly classify different types of crimes based on visual evidence, mitigating the shortcomings of manual methods and empowering law enforcement in their investigative and preventive endeavors. By addressing these challenges, we aim to contribute to the broader goal of rectifying bias, discrimination, and excessive use of force in law enforcement practices, ultimately enhancing justice provision and maintaining law and order in the country."

Step-2: Brainstorm, Idea Listing and Grouping

2

Brainstorm

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

🕒 10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Aditya

Victim Support Services

Ethical AI Guidelines

Transparency in Algorithm Decision-making

Shresthi

Public Education Campaigns

Community Collaboration

Public Awareness Campaigns

Prakhar

Predictive Policing Oversight

Legal Framework for AI in Law Enforcement

Community Review Boards

Aastha

Use of Body Cameras

Real-time Feedback Mechanism

Diverse Dataset Collection

Person 5



Person 6



Person 7



Person 8



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 sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Step-3: Idea Prioritization

4

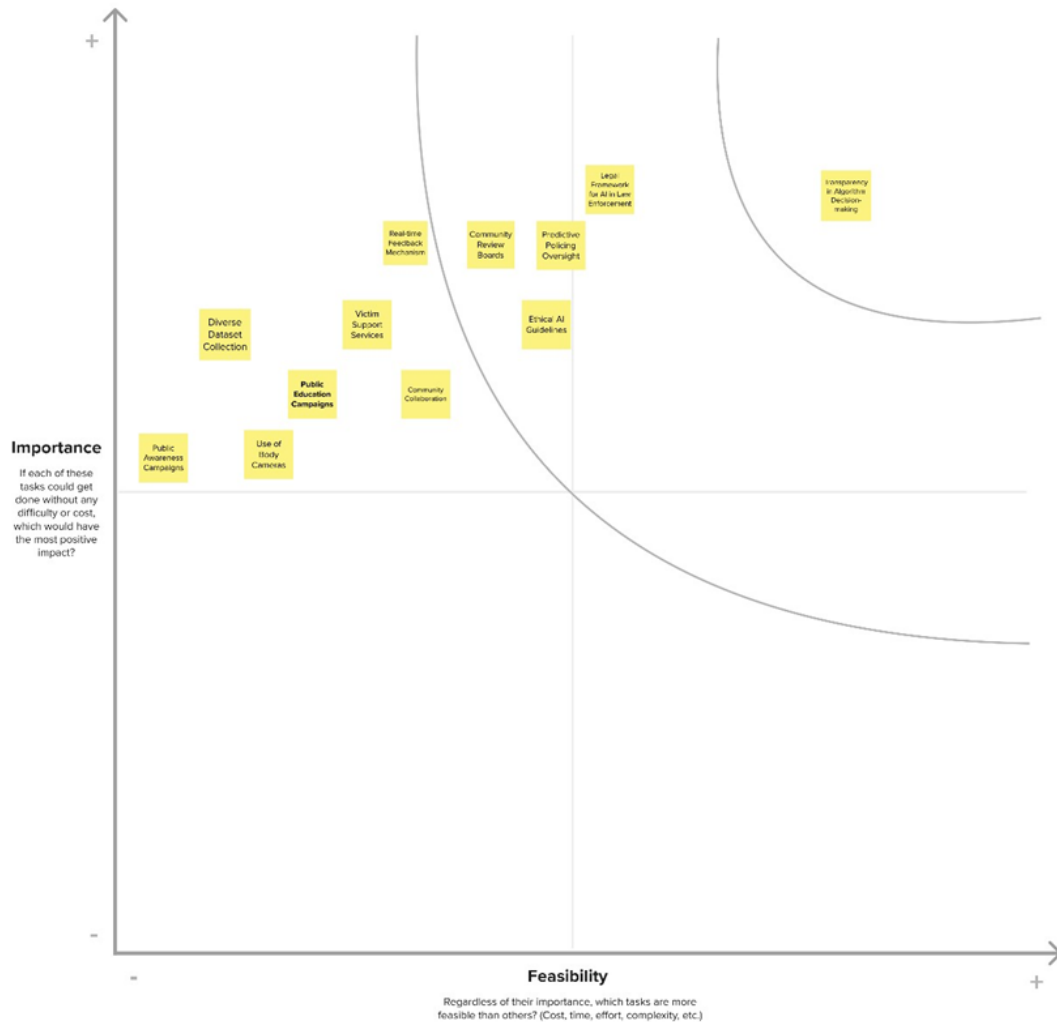
Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H** key on the keyboard.

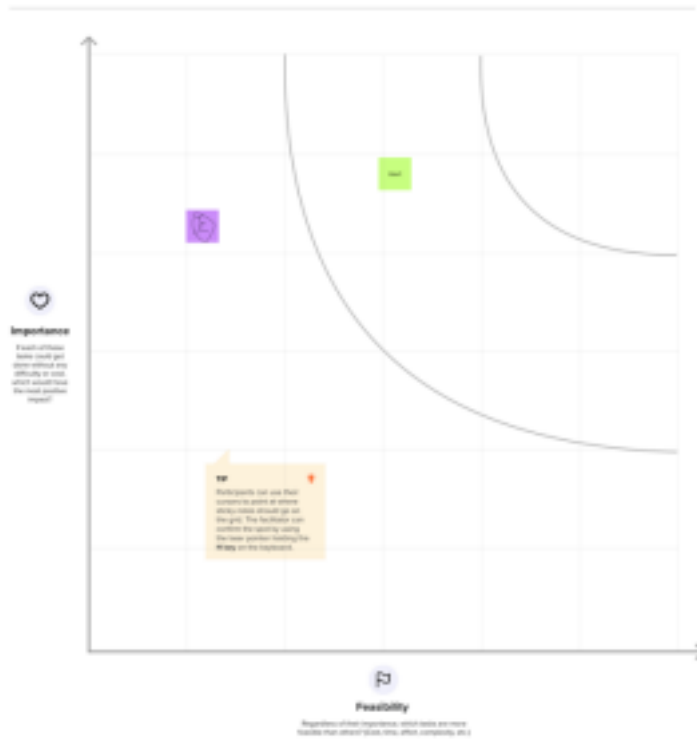


4

Priorities

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes



Project Design Phase-I
Proposed Solution Template

Date	13-12-2023
Team ID	PNT2022TMID592077
Project Name	Project - alzheimer disease prediction
Maximum Marks	2 Marks

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)	Problem is to Detect the MRI Alzheimer disease. With the help of MRI and CNN this should be solved.
2.	Idea / Solution description	CNN Datasets and fast model train and prediction .
3.	Novelty / Uniqueness	MRI Dataset fast predict.
4.	Social Impact / Customer Satisfaction	Easy to detect the Diseases on brain and fast treatment fo it.
5.	Business Model (Revenue Model)	1. Subscription Licensing: Law enforcement agencies pay for

		<p>access tiers based on usage and features.</p> <ol style="list-style-type: none"> 2. Support and Maintenance: Provide ongoing technical support, updates, and maintenance. 3. Partnerships and Collaborations: Integrate with existing law enforcement platforms.
6.	Scalability of the Solution	<ol style="list-style-type: none"> 1. Parallel Processing: Distribute workload for faster processing. 2. Cloud Infrastructure: Utilize scalable cloud platforms. 3. Load Balancing: Ensure even distribution of tasks. <p>Monitoring and Metrics: Track performance for proactive adjustments.</p>

Project Design Phase-I
Solution Architecture

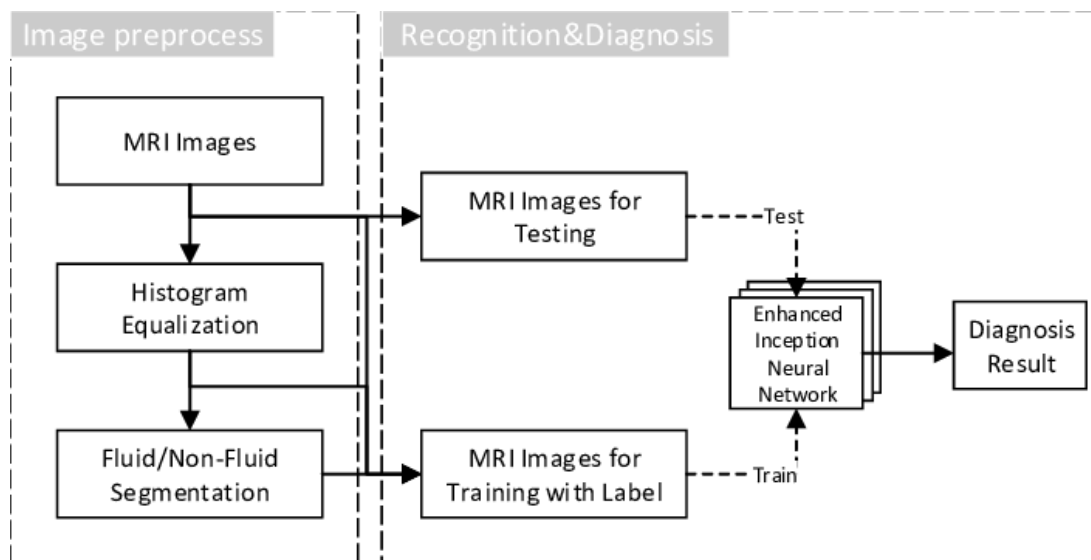
Date	13-12-2023
Team ID	PNT2022TMID592077
Project Name	Project - alzheimer disease prediction
Maximum Marks	4 Marks

Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

Example - Solution Architecture Diagram:

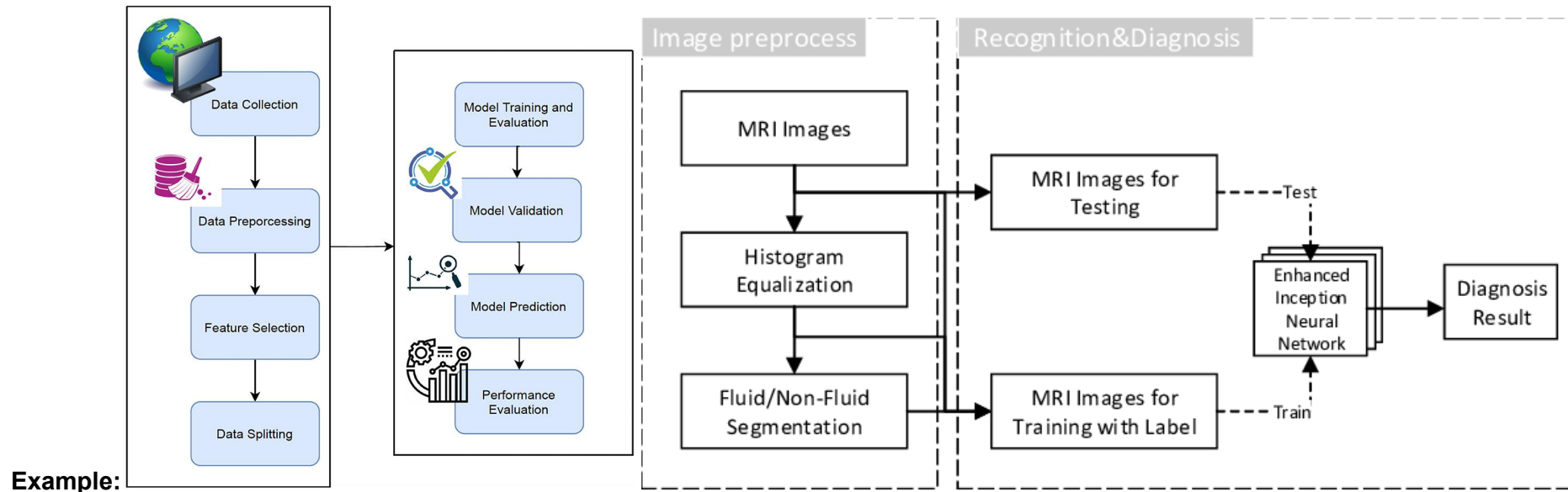


Project Design Phase-II
Data Flow Diagram & User Stories

Date	03 Dec 13, 2023
Team ID	PNT2022TMID592077
Project Name	Project - alzheimer disease prediction
Maximum Marks	4 Marks

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.



User Stories

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

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the	I can receive confirmation email &	High	Sprint-1

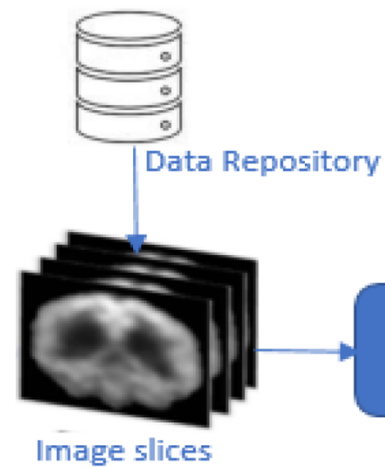
			application	click confirm		
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard					
Customer (Web user)						
Customer Care Executive						
Administrator						

Project Design Phase-II
Technology Stack (Architecture & Stack)

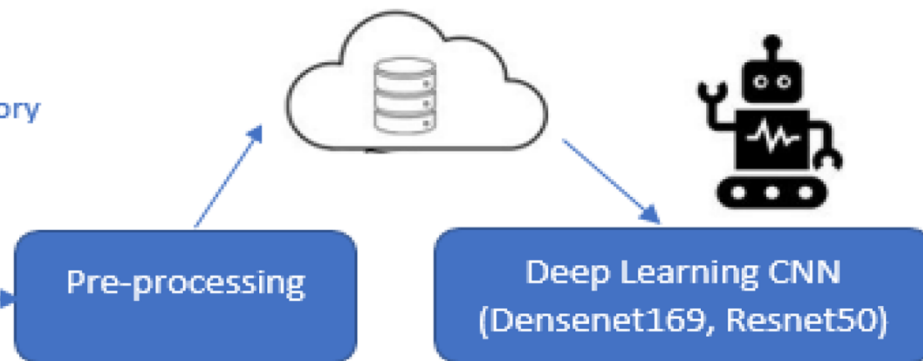
Date	13-12-2023
Team ID	PNT2022TMID592077
Project Name	Project - alzheimer disease prediction
Maximum Marks	4 Marks

Technical Architecture:

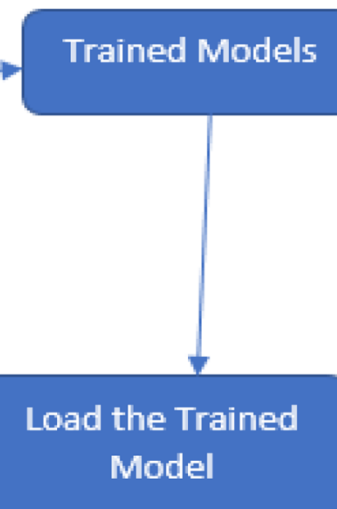
1- Model Building



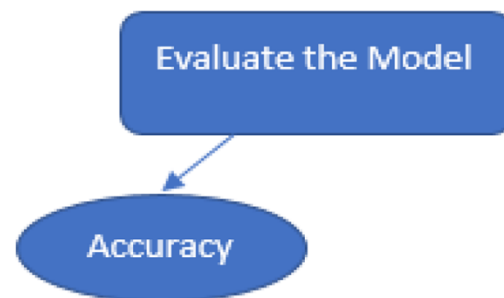
2- Model Training (Cloud or Standalone)



3- Export Trained Model



5- Model Evaluation



4- Model Testing

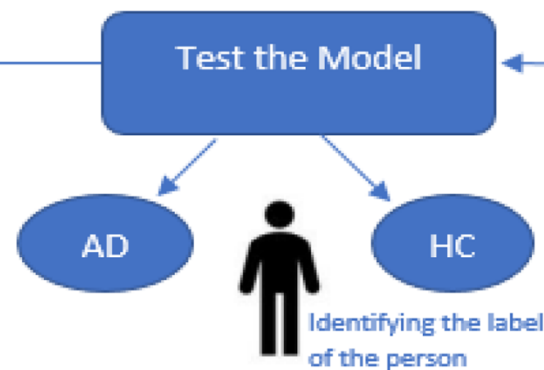


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	Python Cnn datasets
2.	Application Logic-1	Logic for a process in the application	Python
3.	Application Logic-2	Logic for a process in the application	MRI Data sets
4.	Application Logic-3	Logic for a process in the application	Python Libraries
5.	Database	Data Type, Configurations etc.	CNN model
6.	Cloud Database	Database Service on Cloud	Datasets mri
7.	File Storage	File storage requirements	Cloud Storages
8.	External API-1	Purpose of External API used in the application	MRI
9.	External API-2	Purpose of External API used in the application	Python prediction
10.	Machine Learning Model	Purpose of Machine Learning Model	CNN model
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Inbuild servers

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
------	-----------------	-------------	------------

1.	Open-Source Frameworks	List the open-source frameworks used	Deep learning Python
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	Cnn Python
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Python CNN

S.No	Characteristics	Description	Technology
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	Python CNN
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Python CNN

Project Planning Phase

Project Planning Template (Product Backlog, Sprint Planning, Stories, Story points)

Date	13-12-2023
Team ID	PNT2022TMID592077
Project Name	Project - alzheimer disease prediction
Maximum Marks	8 Marks

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	
Sprint-2		USN-3	As a user, I can register for the application through Facebook	2	Low	
Sprint-1		USN-4	As a user, I can register for the	2	Medium	

			application through Gmail			
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	1	High	
	Dashboard					

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022		
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022		

Velocity:

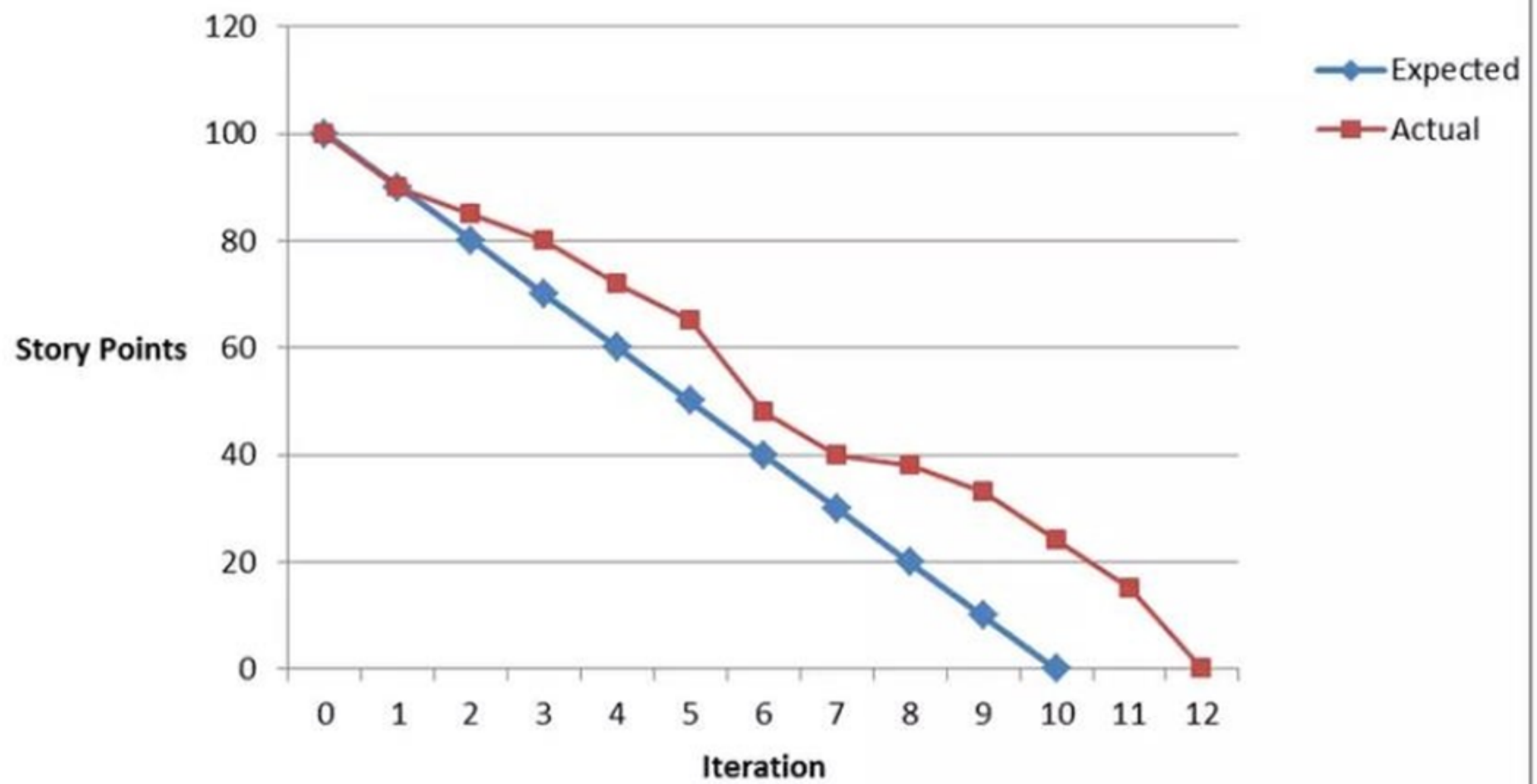
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{10} = 2$$

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.

Burn Down Chart



Alzheimer's Disease

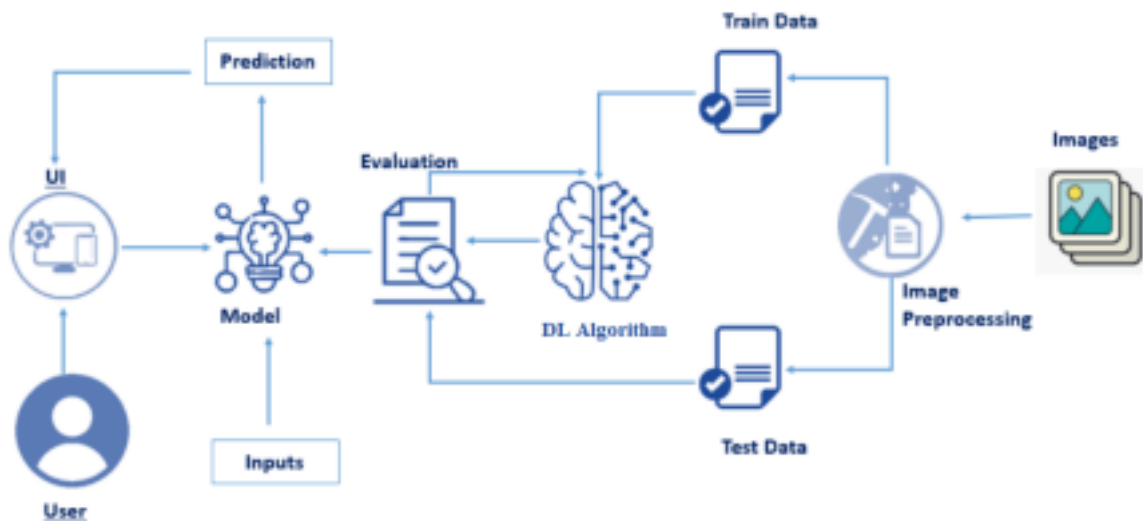
Project Description:

Alzheimer's disease (AD) is a progressive and irreversible neurological disorder that affects the brain, leading to memory loss, cognitive impairment, and changes in behavior and personality. It is the most common cause of dementia among older adults and is characterized by the buildup of abnormal protein deposits in the brain, including amyloid plaques and tau tangles.

The exact cause of Alzheimer's disease is not yet fully understood, but it is believed to be influenced by a combination of genetic, environmental, and lifestyle factors. Age is also a significant risk factor, with the risk of developing the disease increasing significantly after the age of 65. The early symptoms of Alzheimer's disease may include mild memory loss, difficulty with problem-solving, and changes in mood or behavior. As the disease progresses, these symptoms become more severe, with individuals experiencing significant memory loss, difficulty communicating, and a loss of the ability to perform daily activities.

By using deep learning models like Xception to analyze medical imaging data, it may be able to identify early signs of Alzheimer's disease before symptoms become severe. This can help healthcare providers to provide early treatment and support for patients and their families, ultimately leading to better outcomes for all involved.

Technical Architecture:



Project Flow:

- The user interacts with the UI to choose an image.
- The chosen image is processed by a Xception deep learning model.
- The Xception model is integrated with a Flask application.
- The Xception model analyzes the image and generates predictions.
- The predictions are displayed on the Flask UI for the user to see.
- This process enables users to input an image and receive accurate predictions quickly.

To accomplish this, we have to complete all the activities and tasks listed below

- o Data Collection.
 - o Create a Train and Test path.
- o Image Pre-processing.
 - o Import the required library
 - o Configure ImageDataGenerator class
 - o Handling imbalance data
 - o Splitting into train-test split
- o Model Building
 - o Pre-trained CNN model as a Feature Extractor
 - o Creating Sequential layers
 - o Configure the Learning Process
 - o Train the model
 - o Save the Model
 - o Test the model
- o Application Building
 - o Create an HTML file
 - o Build Python Flask Code
 - o Run the application

Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- **Deep Learning Concepts**

- o **CNN**: <https://towardsdatascience.com/basics-of-the-classic-cnn-a3dce1225add>
- o **VGG16**: <https://medium.com/@mygreatlearning/what-is-vgg16-introduction-to-vgg16-f2d63849f615>
- o **ResNet-50**: <https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33>
- o **Inception-V3**: <https://iq.opengenus.org/inception-v3-model-architecture/>
- o **Xception**: <https://pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/>

- **Flask**: Flask is a popular Python web framework, meaning it is a third-party Python library used for developing web applications.

Link: https://www.youtube.com/watch?v=Ij4l_CvBnt0

Project Structure:

Create a Project folder which contains files as shown below

Dataset	15-12-2022 14:30
> test	15-12-2022 14:30
> train	15-12-2022 14:30
01 archive.zip	10-11-2022 22:11
Flask	15-12-2022 14:31
> static	15-12-2022 14:31
> templates	15-12-2022 14:31
> uploads	15-12-2022 14:31
01 adp.h5	11-11-2022 21:02
app.py	12-11-2022 22:38
IBM Files	15-12-2022 14:31
Alzheimer's_DP_IBM_Dep.ipynb	12-11-2022 00:07
cnn_adp.tgz	11-11-2022 23:50
Training	15-12-2022 14:31
01 adp.h5	11-11-2022 21:02
Alzheimer's_disease_prediction.ipynb	15-11-2022 10:14
Alzheimer.docx	15-11-2022 10:10

- The Dataset folder contains the training and testing images for training our model.
- For building a Flask Application we need HTML pages stored in the **templates** folder, CSS for styling the pages stored in the static folder and a python script **app.py** for server side scripting
- The IBM folder consists of a trained model notebook on IBM Cloud.
- Training folder consists of Alzheimer's_disease_prediction.ipynb model training file & adp.h5 is saved model

Milestone 1: Data Collection

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

Activity 1: Download the dataset

Collect images of brain MRI then organize into subdirectories based on their respective names as shown in the project structure. Create folders of types of Alzheimer.

In this project, we have collected images of 4 types of brain MRI images like Mild Demented, Moderate Demented, Non Demented & Very Mild Demented and they are saved in the respective sub directories with their respective names.

You can download the dataset used in this project using the below

link Dataset:- [Alzheimer's Dataset \(4 class of Images \) | Kaggle](#)

Note: For better accuracy train on more images

We are going to build our training model on Google colab so we have to upload a dataset zip file on Google colab.

To upload a dataset zip file to Google Colab and then unzip it, you can follow these steps:

- Open Google Colab and create a new notebook.
 - Click on the "Files" icon on the left-hand side of the screen.
 - Click on the "Upload" button and select the zip file you want to upload.
 - Wait for the upload to complete. You should see the file appear in the "Files" section. •
- To unzip the file, you can use the following command:

```
!unzip '/content/archive.zip'
```

Archive: /content/archive.zip

Activity 2: Create training and testing dataset

To build a DL model we have to split training and testing data into two separate folders. But In the project dataset folder training and testing folders are presented. So, in this case we just have to assign a variable and pass the folder path to it.

```
trainPath = r"/content/Alzheimer_s Dataset/train"
testPath = r"/content/Alzheimer_s Dataset/test"
```

Milestone 2: Image Preprocessing

In this milestone we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although performing some geometric transformations of images like rotation, scaling, translation, etc.

Activity 1: Importing the libraries

Import the necessary libraries as shown in the image

```
from tensorflow.keras.layers import Dense, Flatten, Input, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.keras.applications.xception import Xception
import numpy as np
```

To understand the above imported libraries:-

- **ImageDataGenerator:** The ImageDataGenerator is a class in the tensorflow.keras.preprocessing.image module that generates batches of augmented image data in real-time during model training.
- **Keras:** Keras is a high-level neural network API written in Python that allows for fast experimentation and prototyping of deep learning models.
- **Dense Layer:** A dense layer in neural networks is a fully connected layer where each neuron in the layer is connected to every neuron in the previous layer, and each connection has a weight associated with it. •
- **Flatten Layer:** A flatten layer in neural networks is a layer that reshapes the input tensor into a one-dimensional array, which can then be passed to a fully connected layer.
- **Input Layer:** The input layer in neural networks is the first layer of the network that receives the input data and passes it on to the next layer for further processing.

- **Dropout:** Dropout refers to data, or noise, that's intentionally dropped from a neural network to improve processing and time to results.
- **image:** from tensorflow.keras.preprocessing import image imports the image module from Keras' tensorflow.keras.preprocessing package. This module provides a number of image preprocessing utilities, such as loading images, converting images to arrays, and applying various image transformations.
- **load_img:** load_img is a function provided by the tensorflow.keras.preprocessing.image module that is used to load an image file from the local file system. It takes the file path as input and returns a PIL (Python Imaging Library) image object.
- **Xception:** from tensorflow.keras.applications.xception import Xception imports the Xception class from the tensorflow.keras.applications.xception module, which provides a pre-trained implementation of the Xception convolutional neural network architecture for image classification tasks..
- **Numpy:** It is for performing mathematical functions

Activity 2: Configure ImageDataGenerator class

The ImageDataGenerator class is part of the tensorflow.keras.preprocessing.image module and is used for generating batches of augmented image data for training a neural network. Here's a breakdown of the parameters used in this example:

- **rescale=1./255:** This normalizes the pixel values of the image to the range of [0, 1], which is a common practice in deep learning models.
- **zoom_range=[0.99,1.01]:** This applies random zooming transformations to the image, which can help the model learn to be more robust to different scales of objects in the image.
- **brightness_range=[0.8,1.2]:** We can use it to adjust the brightness_range of any image for Data Augmentation.
- **horizontal_flip=True:** This applies random horizontal flipping to the image, which can help the model learn to be more invariant to the orientation of objects in the image.
- These data augmentation techniques can help increase the diversity and size of the training data, which can improve the performance of the model and reduce overfitting. You can use the train_datagen object to generate batches of augmented training data on the fly, as needed by the fit() method of a Keras model.

An instance of the ImageDataGenerator class can be constructed for train and test.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator as IDG
IMG_SIZE = 180
IMAGE_SIZE = [180, 180]
DIM = (IMG_SIZE, IMG_SIZE)
ZOOM = [.99, 1.01]
BRIGHT_RANGE = [0.8, 1.2]
HORZ_FLIP = True
FILL_MODE = "constant"
DATA_FORMAT = "channels_last"
WORK_DIR = "/content/Alzheimer's Dataset/train"
work_dr = IDG(rescale = 1./255, brightness_range=BRIGHT_RANGE, zoom_range=ZOOM, data_format=DATA_FORMAT, fill_mode=FILL_MODE,
              horizontal_flip=HORZ_FLIP)

train_data_gen = work_dr.flow_from_directory(directory=WORK_DIR, target_size=DIM, batch_size=6500, shuffle=False)
```

Activity 3: Handling imbalance data

```
train_data, train_labels = train_data_gen.next()

#before oversampling
print(train_data.shape, train_labels.shape)

(5121, 180, 180, 3) (5121, 4)
```

The next() function returns the next item in an iterator.

Checking the data before oversampling. As we can see there total images in the train data, i.e, 5121.

```
#Performing over-sampling of the data, since the classes are imbalanced
#after oversampling
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=42)

train_data, train_labels = sm.fit_resample(train_data.reshape(-1, IMG_SIZE * IMG_SIZE * 3), train_labels)

train_data = train_data.reshape(-1, IMG_SIZE, IMG_SIZE, 3)

print(train_data.shape, train_labels.shape)

(10240, 180, 180, 3) (10240, 4)
```

Data after oversampling. Now the data has been doubled due to oversampling. As we can observe, from 5121 to 10240.

Activity 4: Splitting into train test split

```
#Splitting the data into train, test, and validation sets
from sklearn.model_selection import train_test_split
train_data, test_data, train_labels, test_labels = train_test_split(train_data, train_labels, test_size = 0.2, random_state=42)
train_data, val_data, train_labels, val_labels = train_test_split(train_data, train_labels, test_size = 0.2, random_state=42)
```

Now we are splitting the labels into train-test split in 80:20 ratio & assigning them to train_data, train_labels & test_data, test_labels.

Validation sets of train_data, val_data, train_labels, val_labels.

Milestone 3: Model Building

Now it's time to build our model. Let's use the pre-trained model which is Xception, one of the convolution neural net (CNN) architecture which is considered as a very good model for Image classification.

Deep understanding on the Xception model – Link is referred to in the prior knowledge section. Kindly refer to it before starting the model building part.

Activity 1: Pre-trained CNN model as a Feature Extractor

For one of the models, we will use it as a simple feature extractor by freezing all the five convolution blocks to make sure their weights don't get updated after each epoch as we train our own model.

Here, we have considered images of dimension (180,180,3).

Also, we have assigned `include_top = False` because we are using convolution layer for features extraction and wants to train fully connected layer for our images classification(since it is not the part of Imagenet dataset)

```
IMAGE_SIZE = [180, 180]
```

Different Deep learning models are used in our project and the best model (Xception) is selected. The image input size of Xception model is 180, 180.

```
xcep_model = Xception(input_shape=IMAGE_SIZE+[3], weights='imagenet', include_top=False)

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception83683744/83683744 [=====] - 1s 0us/step

for layer in xcep_model.layers:
    layer.trainable=False
```

Activity 2: Creating Sequential layers

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SeparableConv2D, BatchNormalization, GlobalAveragePooling2D
custom_inception_model = Sequential([
    xcep_model,
    Dropout(0.5),
    GlobalAveragePooling2D(),
    Flatten(),
    BatchNormalization(),
    Dense(512, activation='relu'),
    BatchNormalization(),
    Dropout(0.5),
    Dense(256, activation='relu'),
    BatchNormalization(),
    Dropout(0.5),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    BatchNormalization(),
    Dense(4, activation='softmax')
], name = "inception_cnn_model")
```

For this purpose we have imported **Sequential** from `tensorflow.keras.models`. As the name suggests it is used to arrange the Keras layers in a sequential manner.

Now from layers we are importing **GlobalAveragePooling2D**. `GlobalAveragePooling2D` accepts as input 4D tensor. It operates the mean on the height and width dimensionalities for all the channels.

SeperableConv2D : This function is separable convolution which is a way to factorize a convolution kernel into two

smaller kernels.

BatchNormalization : Batch Norm is a normalization technique done between the layers of a Neural Network instead of in the raw data. It serves to speed up training and use higher learning rates, making learning easier.

ReLU activation function has been taken in the dense layers & softmax in the output layer.

A **dense** layer is a deeply connected neural network layer. It is the most common and frequently used layer.

Let us create a model object named model with inputs as resnet.input and output as dense layer.

The number of neurons in the Dense layer is the same as the number of classes in the training set. The neurons in the last Dense layer, use softmax activation to convert their outputs into respective probabilities. Understanding the model is a very important phase to properly use it for training and prediction purposes.

Activity 3: Configure the Learning Process

The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find errors or deviations in the learning process. Keras requires a loss function during the model compilation process.

Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using 'rmsprop' optimizer.

Metrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process.

We have defined metric parameter in a separate variable called METRICS.

```
import tensorflow
METRICS = [tensorflow.keras.metrics.CategoricalAccuracy(name='acc'),
            tensorflow.keras.metrics.AUC(name='auc')]
custom_inception_model.compile(optimizer='rmsprop',
                                loss=tensorflow.losses.CategoricalCrossentropy(),
                                metrics=METRICS)
```

Activity 4: Train the model

Now, let us train our model with our image dataset. The model is trained for 30 epochs and after every epoch, the current model state is saved if the model has the least loss encountered till that time. We can see that the training loss decreases in almost every epoch.

fit functions used to train a deep learning neural network

Arguments:

- Epochs: an integer and number of epochs we want to train our model for.
- validation_data can be either:

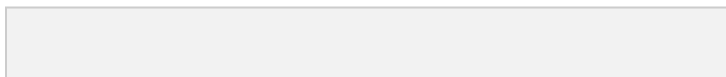
- an inputs and targets list
- a generator
- an inputs, targets, and sample_weights list which can be used to evaluate the loss and metrics for any model after any epoch has ended.

```
history = custom_inception_model.fit(train_data, train_labels, validation_data=(val_data, val_labels), epochs=30)
```

Epoch 1/30
205/205 [=====] - 41s 138ms/step - loss: 1.3664 - acc: 0.4218 - auc: 0.6887 - val_loss: 0.8754 - val_acc: 0.6816 - val_auc: 0.8884
Epoch 2/30
205/205 [=====] - 23s 111ms/step - loss: 0.9175 - acc: 0.5834 - auc: 0.8471 - val_loss: 0.7019 - val_acc: 0.6602 - val_auc: 0.9067
Epoch 3/30
205/205 [=====] - 23s 112ms/step - loss: 0.7889 - acc: 0.6388 - auc: 0.8889 - val_loss: 0.6434 - val_acc: 0.6992 - val_auc: 0.9214
Epoch 4/30
205/205 [=====] - 22s 108ms/step - loss: 0.7328 - acc: 0.6787 - auc: 0.9023 - val_loss: 0.6177 - val_acc: 0.7035 - val_auc: 0.9276
Epoch 5/30
205/205 [=====] - 23s 111ms/step - loss: 0.6794 - acc: 0.6998 - auc: 0.9167 - val_loss: 0.6120 - val_acc: 0.7132 - val_auc: 0.9294
Epoch 6/30
205/205 [=====] - 23s 111ms/step - loss: 0.6409 - acc: 0.7159 - auc: 0.9258 - val_loss: 0.5693 - val_acc: 0.7444 - val_auc: 0.9390
Epoch 7/30
205/205 [=====] - 23s 112ms/step - loss: 0.6185 - acc: 0.7345 - auc: 0.9338 - val_loss: 0.5607 - val_acc: 0.7498 - val_auc: 0.9419
Epoch 8/30
205/205 [=====] - 23s 110ms/step - loss: 0.5950 - acc: 0.7442 - auc: 0.9374 - val_loss: 0.5410 - val_acc: 0.7578 - val_auc: 0.9462
Epoch 9/30
205/205 [=====] - 23s 110ms/step - loss: 0.5635 - acc: 0.7659 - auc: 0.9437 - val_loss: 0.5349 - val_acc: 0.7602 - val_auc: 0.9474
Epoch 10/30
205/205 [=====] - 23s 111ms/step - loss: 0.5353 - acc: 0.7845 - auc: 0.9498 - val_loss: 0.4991 - val_acc: 0.7858 - val_auc: 0.9546
Epoch 11/30
205/205 [=====] - 23s 111ms/step - loss: 0.4883 - acc: 0.7967 - auc: 0.9574 - val_loss: 0.4878 - val_acc: 0.7889 - val_auc: 0.9564
Epoch 12/30
205/205 [=====] - 23s 111ms/step - loss: 0.5126 - acc: 0.7990 - auc: 0.9552 - val_loss: 0.4689 - val_acc: 0.8017 - val_auc: 0.9600
Epoch 13/30
205/205 [=====] - 22s 107ms/step - loss: 0.4609 - acc: 0.8205 - auc: 0.9625 - val_loss: 0.4541 - val_acc: 0.8035 - val_auc: 0.9627
Epoch 14/30
205/205 [=====] - 23s 112ms/step - loss: 0.4514 - acc: 0.8228 - auc: 0.9637 - val_loss: 0.4294 - val_acc: 0.8206 - val_auc: 0.9668
Epoch 15/30
205/205 [=====] - 22s 108ms/step - loss: 0.4479 - acc: 0.8317 - auc: 0.9644 - val_loss: 0.4415 - val_acc: 0.8194 - val_auc: 0.9658
Epoch 16/30
205/205 [=====] - 22s 108ms/step - loss: 0.4067 - acc: 0.8430 - auc: 0.9704 - val_loss: 0.4266 - val_acc: 0.8328 - val_auc: 0.9678
Epoch 17/30
205/205 [=====] - 22s 109ms/step - loss: 0.4091 - acc: 0.8521 - auc: 0.9708 - val_loss: 0.4180 - val_acc: 0.8359 - val_auc: 0.9692
Epoch 18/30
205/205 [=====] - 22s 108ms/step - loss: 0.3851 - acc: 0.8575 - auc: 0.9730 - val_loss: 0.4261 - val_acc: 0.8377 - val_auc: 0.9681
Epoch 19/30
205/205 [=====] - 22s 108ms/step - loss: 0.3857 - acc: 0.8562 - auc: 0.9732 - val_loss: 0.4132 - val_acc: 0.8426 - val_auc: 0.9703
Epoch 20/30
205/205 [=====] - 23s 112ms/step - loss: 0.3608 - acc: 0.8692 - auc: 0.9763 - val_loss: 0.3965 - val_acc: 0.8469 - val_auc: 0.9724
Epoch 21/30
205/205 [=====] - 22s 107ms/step - loss: 0.3585 - acc: 0.8683 - auc: 0.9763 - val_loss: 0.3992 - val_acc: 0.8322 - val_auc: 0.9721
Epoch 22/30
205/205 [=====] - 22s 108ms/step - loss: 0.3299 - acc: 0.8810 - auc: 0.9797 - val_loss: 0.3916 - val_acc: 0.8530 - val_auc: 0.9737
Epoch 23/30
205/205 [=====] - 23s 111ms/step - loss: 0.3308 - acc: 0.8787 - auc: 0.9798 - val_loss: 0.3840 - val_acc: 0.8523 - val_auc: 0.9751
Epoch 24/30
205/205 [=====] - 23s 112ms/step - loss: 0.3272 - acc: 0.8839 - auc: 0.9799 - val_loss: 0.3974 - val_acc: 0.8426 - val_auc: 0.9724
Epoch 25/30
205/205 [=====] - 22s 107ms/step - loss: 0.3123 - acc: 0.8883 - auc: 0.9816 - val_loss: 0.3919 - val_acc: 0.8517 - val_auc: 0.9740
Epoch 26/30
205/205 [=====] - 23s 111ms/step - loss: 0.3126 - acc: 0.8921 - auc: 0.9815 - val_loss: 0.3859 - val_acc: 0.8560 - val_auc: 0.9745
Epoch 27/30
205/205 [=====] - 23s 111ms/step - loss: 0.3092 - acc: 0.8924 - auc: 0.9818 - val_loss: 0.4378 - val_acc: 0.8371 - val_auc: 0.9694
Epoch 28/30
205/205 [=====] - 23s 111ms/step - loss: 0.2808 - acc: 0.9002 - auc: 0.9848 - val_loss: 0.3959 - val_acc: 0.8499 - val_auc: 0.9740
Epoch 29/30
205/205 [=====] - 23s 111ms/step - loss: 0.2783 - acc: 0.9026 - auc: 0.9853 - val_loss: 0.3989 - val_acc: 0.8542 - val_auc: 0.9753
Epoch 30/30
205/205 [=====] - 22s 107ms/step - loss: 0.2784 - acc: 0.9007 - auc: 0.9853 - val_loss: 0.3901 - val_acc: 0.8542 - val_auc: 0.9755

Activity 5: Save the Model

Out of all the models we tried (CNN, VGG19, Resnet50, Inception V3 & Xception) Xception gave us the best accuracy. So we are saving Xception as our final model



The model is saved with .h5 extension as follows

An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

Activity 6: Testing the model

Evaluation is a process during the development of the model to check whether the model is the best fit for the given problem and corresponding data. Load the saved model using `load_model`.



Index positions:

MildDemented = 0
ModerateDemented = 1
Nondemented = 2
VeryMildDemented = 3

Taking an image as input and checking the results



So our model will give the index position of the label. In this case 3rd index is for VeryMildDemented,

which has been predicted correctly.

Milestone 4: Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building python code

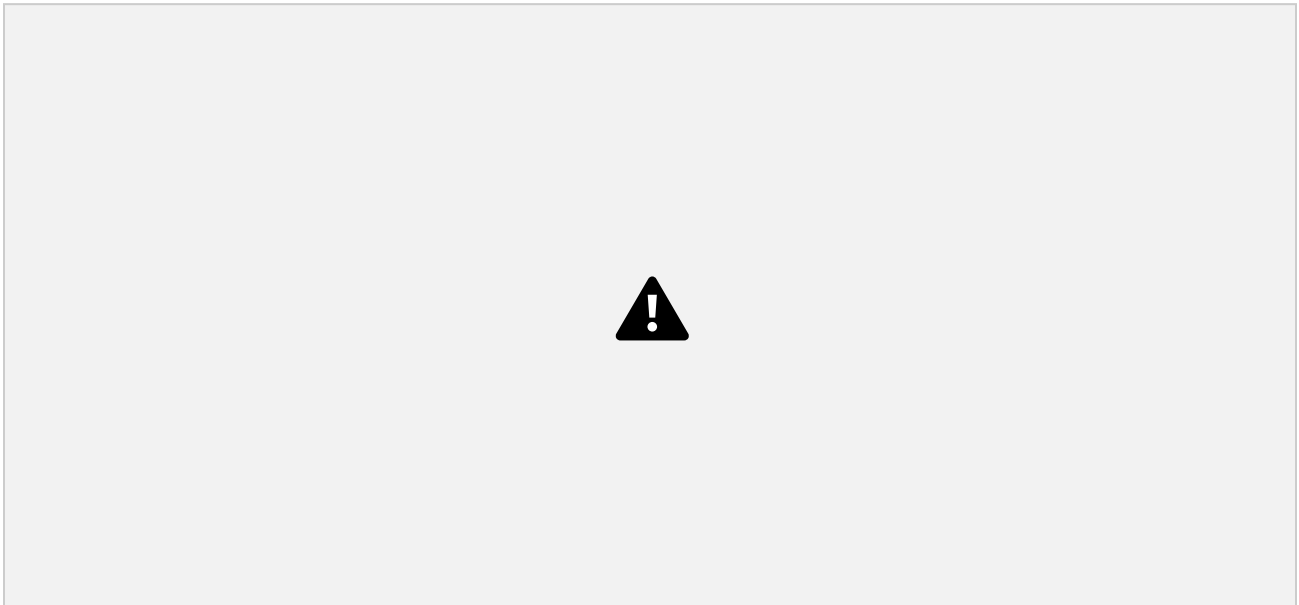
Activity1: Building Html Pages:

For this project create one HTML file namely

- alzheimers.html

Let's see how our alzheimers.html page looks like:

When you click on the Drop the image for detection button, you will be redirecting to the following page



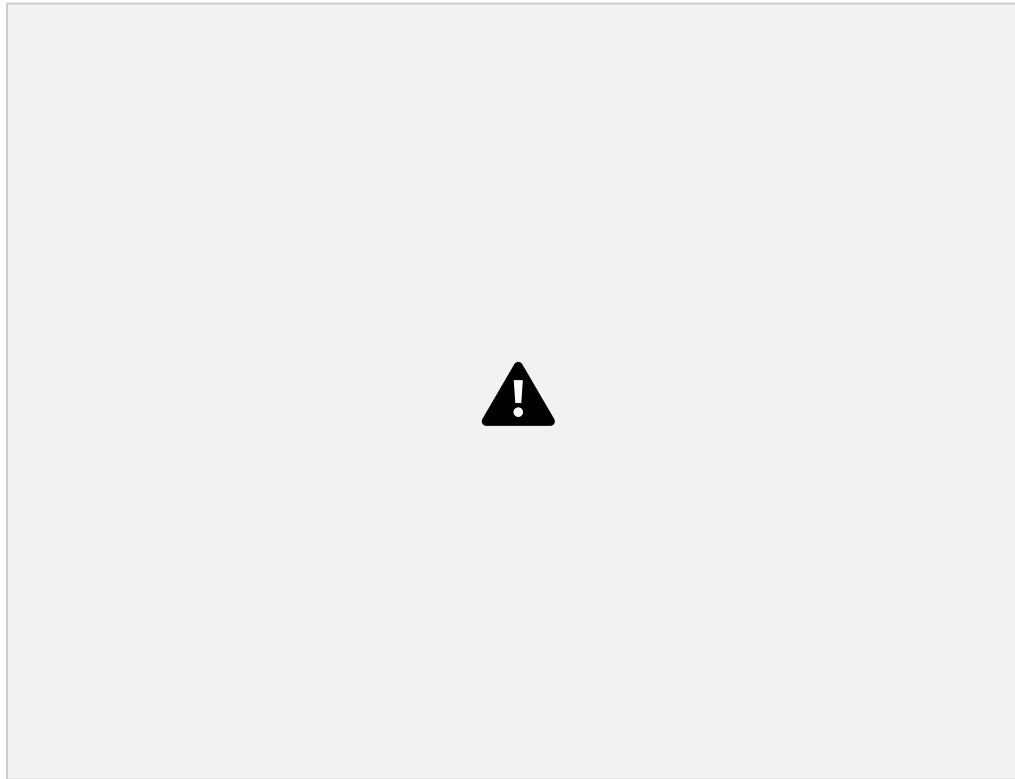
When you click on the Choose button, it will redirect you to the below page



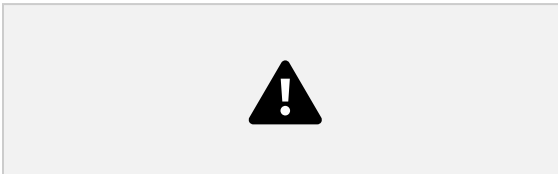
When you select the image it will show like this



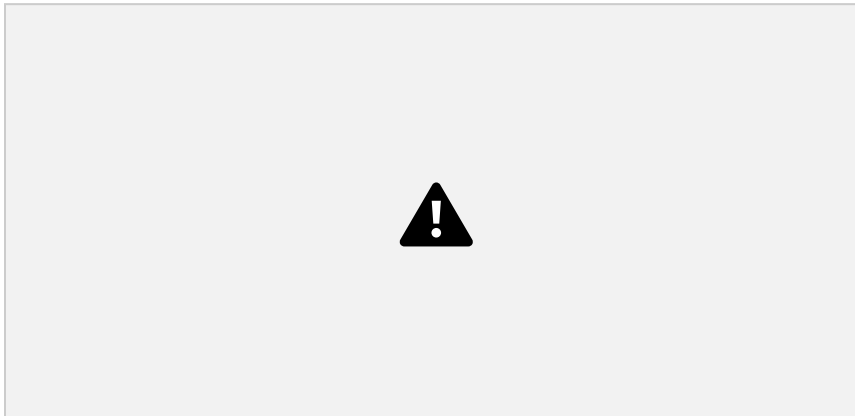
Activity 2: Build Python code:
Import the libraries



Loading the saved model and initializing the flask app



Render HTML pages:



Once we uploaded the file into the app, then verifying the file uploaded properly or not. Here we will be using declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with alzheimers.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.



Here we are routing our app to upload function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the `model.predict()` function. This function returns the prediction. And this prediction value will rendered to the text that we have mentioned in the `alzheimers.html` page earlier.

Main Function:



Activity 3: Run the application

- Open Spyder
- Navigate to the folder where your Python script is.
- Now click on the green play button above.
- Click on the predict button from the top right corner, enter the inputs, click on the Classify button, and see the result/prediction on the web.



The home page looks like this. When you click on the Predict button, you'll be redirected to the predict section



click on Drop the scan for detection button
Input :



Once you upload the image and click on Predict button, the output will be displayed in the below page Output:

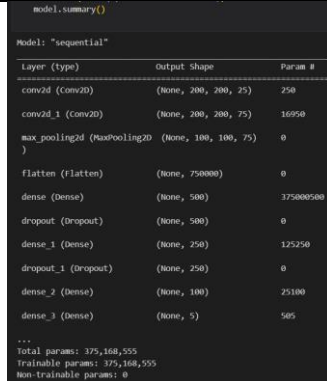
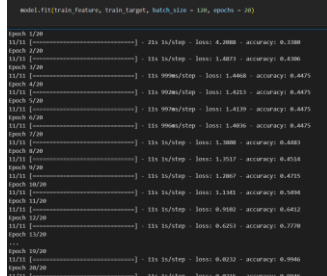
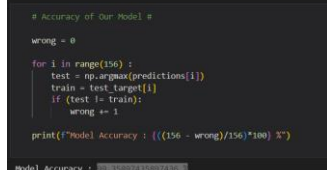


Project Development Phase Model Performance Test

Date	10 November 2022
Team ID	PNT2022TMID592077
Project Name	Project - alzheimer disease prediction
Maximum Marks	10 Marks

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Model Summary	-here is an summary of given model	 <pre> model.summary() Model: "sequential" Layer (type) Output Shape Param # ----- conv2d (Conv2D) (None, 200, 200, 25) 250 conv2d_1 (Conv2D) (None, 200, 200, 75) 16950 max_pooling2d (MaxPooling2D) (None, 100, 100, 75) 0 flatten (Flatten) (None, 750000) 0 dense (Dense) (None, 500) 375000500 dropout (Dropout) (None, 500) 0 dense_1 (Dense) (None, 250) 125250 dropout_1 (Dropout) (None, 250) 0 dense_2 (Dense) (None, 100) 25100 dense_3 (Dense) (None, 5) 505 ... Total params: 375,168,555 Trainable params: 375,168,555 Non-trainable params: 0 </pre>
2.	Accuracy	Training Accuracy - 0.9946 Validation Accuracy -0.0215	 <pre> model.fit(train_features, train_target, batch_size = 128, epochs = 20) Epoch 1/20: 21s 1s/step - loss: 4.2888 - accuracy: 0.0000 Epoch 2/20: 11s 1s/step - loss: 1.8873 - accuracy: 0.0000 Epoch 3/20: 11s 99ms/step - loss: 1.4882 - accuracy: 0.0075 Epoch 4/20: 11s 99ms/step - loss: 1.4023 - accuracy: 0.0075 Epoch 5/20: 11s 99ms/step - loss: 1.4039 - accuracy: 0.0075 Epoch 6/20: 11s 99ms/step - loss: 1.4006 - accuracy: 0.0075 Epoch 7/20: 11s 1s/step - loss: 1.3888 - accuracy: 0.0083 Epoch 8/20: 11s 1s/step - loss: 1.3517 - accuracy: 0.0125 Epoch 9/20: 11s 1s/step - loss: 1.2887 - accuracy: 0.0125 Epoch 10/20: 11s 1s/step - loss: 1.1381 - accuracy: 0.1000 Epoch 11/20: 11s 1s/step - loss: 0.9582 - accuracy: 0.0012 Epoch 12/20: 11s 1s/step - loss: 0.8253 - accuracy: 0.7778 Epoch 13/20: 11s 1s/step - loss: 0.8232 - accuracy: 0.9946 Epoch 14/20: 11s 1s/step - loss: 0.8225 - accuracy: 0.9946 Epoch 15/20: 11s 1s/step - loss: 0.8225 - accuracy: 0.9946 </pre>
3.	Confidence Score (Only Yolo Projects)	Model accuracy- 99.35897435897436 %	 <pre> # Accuracy of Our Model wrong = 0 for i in range(156): test = np.argmax(predictions[i]) train = test_target[i] if (test != train): wrong += 1 print("Model Accuracy : (((156 - wrong)/156)*100) %") Model Accuracy : 99.35897435897436 </pre>