



Says

What have we heard them say?
What can we imagine them saying?

What We Have Heard Them Say:

1. "I sometimes forget my own name. It's terrifying."
2. "I wish there was a way to slow down the memory loss."
3. "I'm worried about becoming a burden to my family."
4. "It's frustrating not being able to remember important dates."
5. "I don't want to lose my independence. That's very important to me."

What We Can Imagine Them Saying:

1. "I hope that one day, there will be a cure for Alzheimer's."
2. "It's essential that healthcare providers are patient and understanding with Alzheimer's patients."
3. "I imagine that technology could be a lifeline for people like me."
4. "I wish people would be more empathetic and less judgmental about Alzheimer's."
5. "I want to be part of a community that understands what I'm going through."

Thinks

What are their wants, needs, hopes, and dreams?
What other thoughts might influence their behavior?



Desires and Needs:

- Individuals affected by Alzheimer's often desire solutions that can help them retain their independence and quality of life. They need reliable tools for early detection and support in managing their symptoms.

Influence of Caregivers:

- Thoughts often revolve around the impact of Alzheimer's on their caregivers and family members. They may consider how their condition affects the emotional well-being of those close to them.

Emotional State:

- The emotional state, including feelings of confusion, frustration, and sadness, significantly influences their thoughts. They may think about ways to manage their emotions.



Alzheimer patient

Alzheimer's disease is a brain disorder that slowly destroys memory and thinking skills and, eventually, the ability to carry out the simplest tasks

Researches Alzheimer's:

- Family members and caregivers may actively research Alzheimer's disease to understand its symptoms and progression.
- Technology enthusiasts might explore various technologies and tools related to Alzheimer's.

Seeks Support and Resources:

- Caregivers and family members may look for support groups, online forums, and resources to help them care for Alzheimer's patients
- Healthcare providers may seek educational resources and training on the latest diagnostic tools and treatments.

Seeks Medical Advice:

- Alzheimer's patients may visit healthcare providers to seek a diagnosis or treatment.
- Caregivers might accompany patients to medical appointments and be involved in decision-making.

Impact on Relationships:

- Alzheimer's often strains relationships, as the individual may struggle to recognize loved ones, causing a sense of loss. Family members may express worry, leading to emotional distress.

Concerns and Worries:

- The most significant concern for the individual is often the fear of losing their independence. They worry about becoming a burden to their loved ones and the uncertainty of what the future holds

Emotions Related to Alzheimer's:

- The individual often experiences a range of emotions, including confusion, frustration, and sadness. They may frequently find themselves struggling with memory loss and cognitive challenges, which can be emotionally distressing.



Does

What behavior have we observed?
What can we imagine them doing?

Feels

What are their fears, frustrations, and anxieties?
What other feelings might influence their behavior?



PAIN

Frustration and distress from memory loss.

Fear of losing independence.

Strain on relationships and communication.

GAIN

Early detection of Alzheimer's, aiding timely intervention.

Enhanced quality of life for patients and their caregivers.


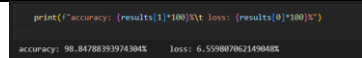
Decreased burden on healthcare systems and families

Project Development Phase Model Performance Test

Date	10 November 2023
Team ID	PNT2022TMIDxxxxxx
Project Name	Alzheimerms Detection
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	CNN to predict alzheimers disease from an image scan	
2.	Accuracy	Training Accuracy – 95.29 Test Accuracy -	
3.	Confidence Score (Only Yolo Projects)	Class Detected - Confidence Score -	

Solution Architecture Summary: Alzheimer Detection System

1. Introduction:

The Alzheimer Detection System combines frontend technologies, backend processing, and machine learning for early Alzheimer's detection. This report outlines the architecture facilitating user interaction, model prediction, and result visualization.

2. Frontend:

2.1 UI:

Bootstrap and EJS create a user-friendly interface for image uploads.

2.2 NodeJS and Express:

NodeJS with Express.js manages user requests, rendering views, and communication with the backend.

3. Backend:

3.1 Flask API:

The Flask API processes image data, invoking a CNN model with SMOTE for improved predictions.

3.2 Model:

The CNN model, saved in H5 format, efficiently predicts Alzheimer's likelihood.

4. Integration:

4.1 User Interaction:

Users upload images, and NodeJS forwards data to the Flask API.

4.2 Processing:

The Flask API employs the model, and results are sent to the frontend.

4.3 Visualization:

ChartJS dynamically displays bar charts representing prediction probabilities.

5. Deployment:

5.1 Cloud Hosting:

Deployable on cloud platforms for scalability.

5.2 Containerization:

Docker containers encapsulate components for easy deployment.

6. Scalability and Extensibility:

Designed for scalability with cloud services and containerization. Extensible for future enhancements.

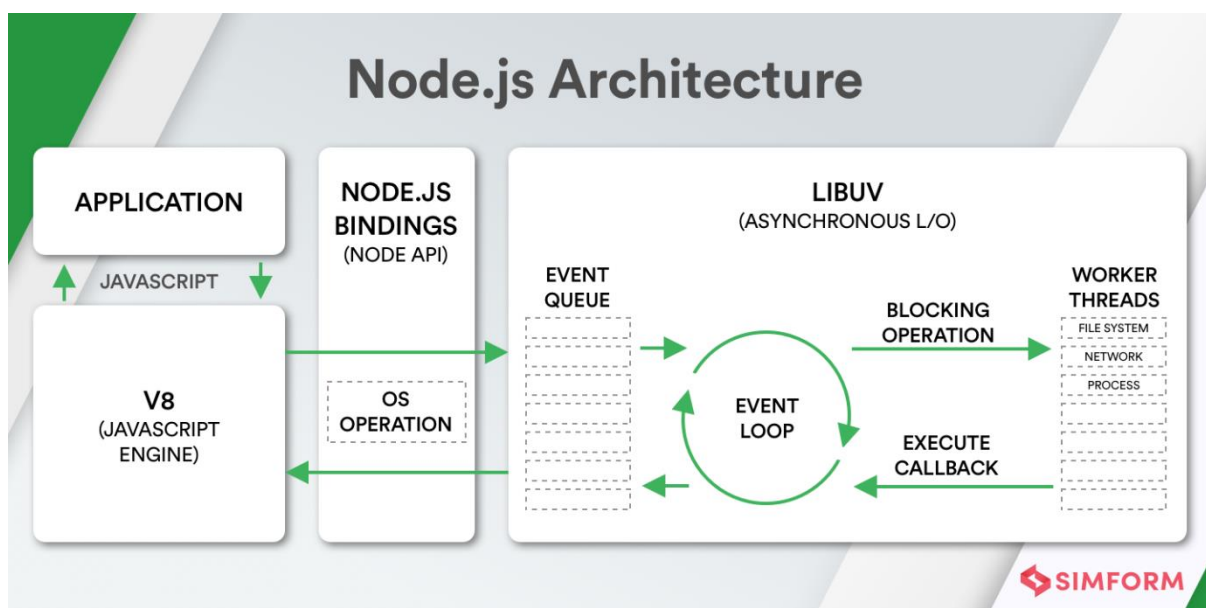
7. Security:

Implement secure communication with HTTPS and access controls for data protection.

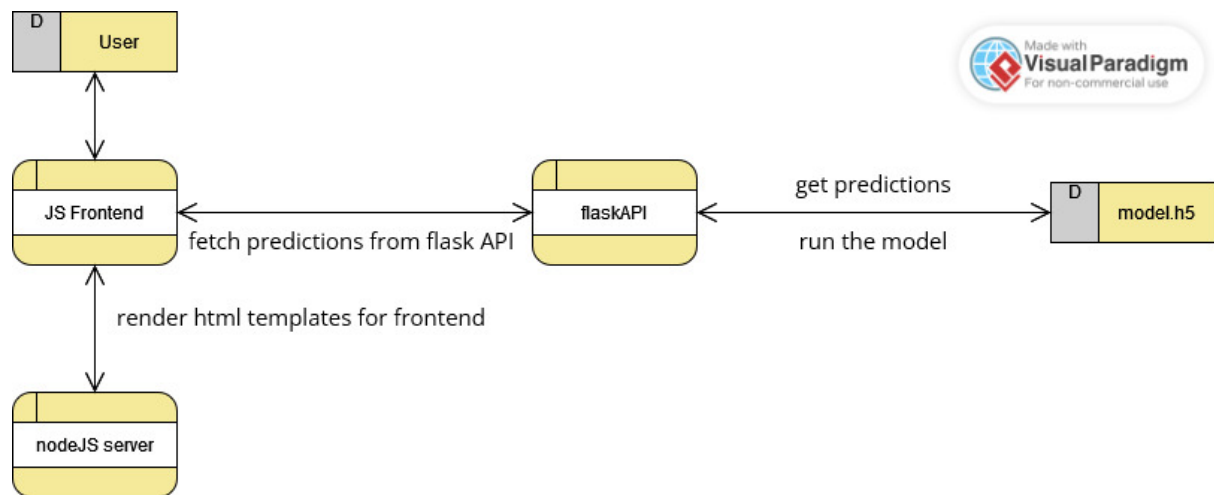
8. Conclusion:

The Alzheimer Detection System's modular architecture integrates frontend, backend, and machine learning, providing a flexible, scalable, and accurate tool for Alzheimer's prediction

Architecture:



DataFlow Diagram:



Project Design Phase-I

Date	23 rd of October, 2023
Team ID	Dhanush
Project Name	Alzheimers detection
Maximum marks	2 Marks

S.No.	Parameter	Description
1.	Problem Statement	Build and train a model to predict whether a said patient has Alzheimer's or not
2.	Idea	<ul style="list-style-type: none">• Gather Data and preprocess it• Build a CNN and train the model• Optimise the model• Save the weights• Build a UI to interact with the model
3.	Uniqueness	This is not that big of a unique idea but uniqueness can be found in optimization, processing of our data, handling null, nan values etc. Uniqueness can be found in how we created/trained our model rather than our idea, since this project has been done before many times.
4.	Social impact	This is mostly useful in medical industry especially Alzheimer's patients.
5.	Business model	People can profit off from this model by making it as subscription-based service and distributing it to the hospitals and other health monitoring services
6	Scalability of the solution	This project can be scaled with other projects by using saved weights file (model.h5) after training our model.

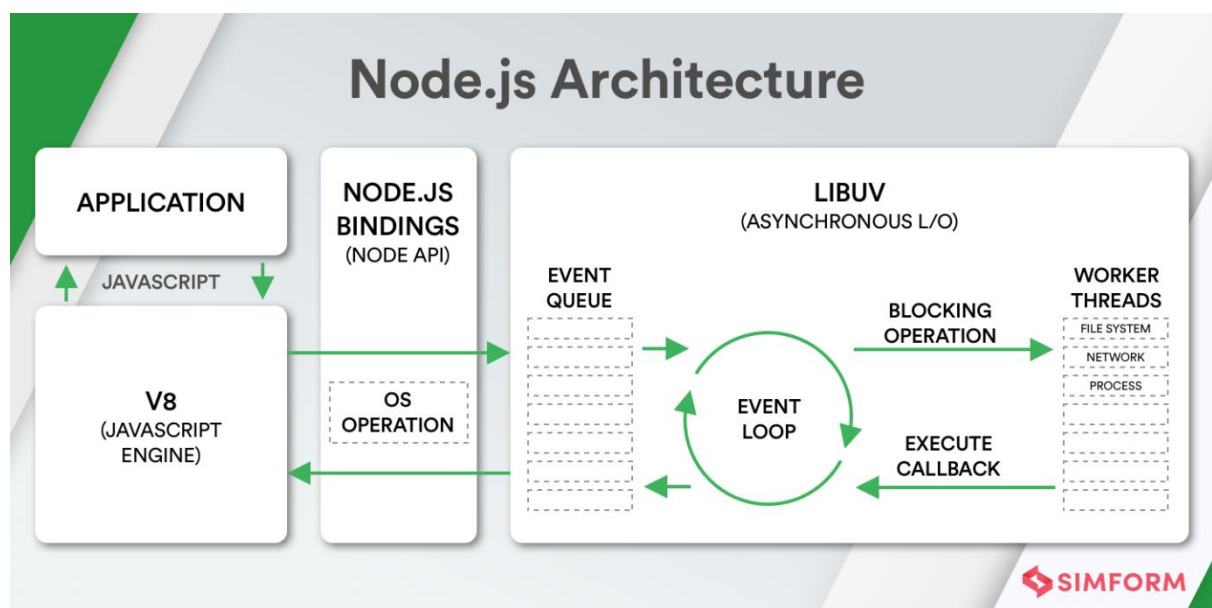
Project Description:

The Alzheimer's Disease Detection System is an innovative project designed to tackle the challenges presented by Alzheimer's disease (AD), a progressive and irreversible neurological disorder affecting memory, cognition, and behavior. As the most common cause of dementia among older adults, AD is characterized by abnormal protein deposits in the brain, such as amyloid plaques and tau tangles. While the exact cause of AD remains incompletely understood, factors like genetics, environment, and lifestyle are believed to contribute, with age being a significant risk factor, particularly after the age of 65.

The project leverages deep learning models, specifically utilizing Xception, to analyze medical imaging data for the early detection of Alzheimer's disease. By identifying subtle signs before symptoms become severe, the system aims to provide healthcare providers with valuable tools for early intervention and support. Early symptoms may include mild memory loss and difficulty with problem-solving, progressing to severe memory impairment and an inability to perform daily activities. This proactive approach to detection holds the promise of improving outcomes for patients and their families, offering the potential for early treatment and support.

Through the integration of advanced deep learning techniques into the healthcare landscape, the Alzheimer's Disease Detection System seeks to enhance diagnostic capabilities, enabling timely interventions that may ultimately lead to better outcomes and an improved quality of life for those affected by Alzheimer's disease.

Technical Architecture:



Project Flow:

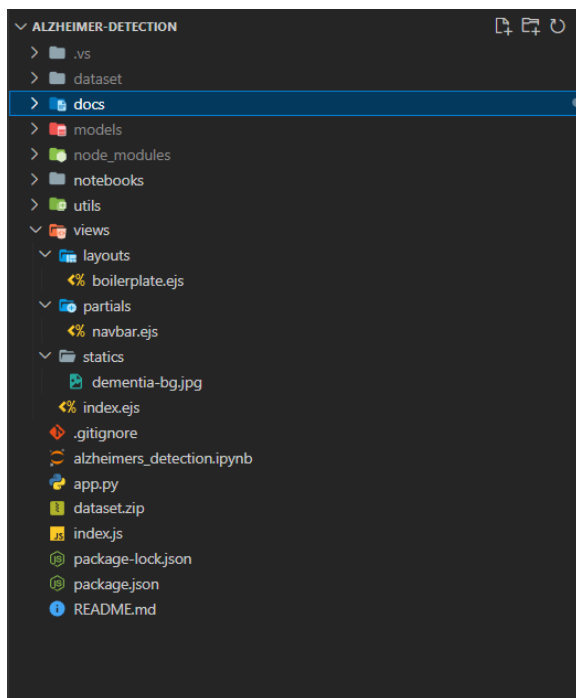
- The user interacts with the UI to choose an image.
- The chosen image is sent through a protocol as a request body
- The flask API receives the image from the client
- The flask API then uses the model to predict and send the prediction as a response
- The predictions are displayed on the website using chartJS run on nodeJS.
- 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

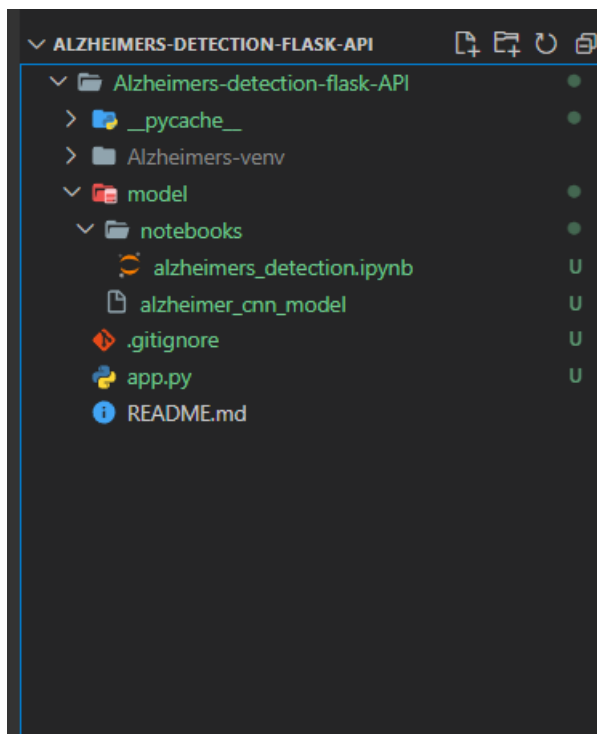
1. Data Collection.
2. Create a Train and Test path.
3. Image Pre-processing.
4. Import the required library
5. Configure ImageDataGenerator class
6. Handling imbalance data
7. Splitting into train-test split
8. Model Building
9. Pre-trained CNN model as a Feature Extractor
10. Creating Sequential layers
11. Configure the Learning Process
12. Train the model
13. Save the Model
14. Test the model
15. Application Building
16. Build a flask API to handle incoming images and run prediction on it
17. Build an expressJS server and ejs template to implement UI
18. Use fetch to the flaskAPI to send image scan and get predictions
19. Run the application

Project Structure:

1)NodeJS structure:



2)flask API:



Milestone 1: Data Collection

In the initial phase of our project, we focused on gathering a diverse and representative dataset crucial for training our Alzheimer's detection model. Leveraging reputable open sources such as Kaggle and the UCI repository, we obtained a comprehensive set of brain MRI images. These images, categorized into four distinct types – Mild Demented, Moderate Demented, Non-Demented, and Very Mild Demented – were meticulously organized into subdirectories as outlined in the project structure. This meticulous organization facilitates a streamlined approach to dataset management.

For simplicity, we have made the dataset used in this project accessible through the following link: [Alzheimer's Dataset \(4 classes of Images\)](#).

Note: To ensure optimal model accuracy, we encourage the inclusion of additional images in the training set.

```
drive sample_data

from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

To facilitate seamless integration with Google Colab, where our training model will be constructed, we recommend uploading the dataset as a compressed zip file. Follow these steps on Google Colab to achieve this:

1. Open a new notebook on Google Colab.
2. Navigate to the "Files" icon on the left-hand panel.
3. Select "Upload" and choose the zip file containing the dataset.
4. Wait for the upload to conclude, and confirm the file's presence in the "Files" section.

To unzip the file, execute the appropriate command in a code cell.

```
!unzip "/content/drive/MyDrive/Alzheimers detection/dataset.zip" -d "/content"
```

Streaming output truncated to the last 5000 lines.

```
inflating: /content/dataset/train/MildDemented/mildDem21.jpg
inflating: /content/dataset/train/MildDemented/mildDem210.jpg
inflating: /content/dataset/train/MildDemented/mildDem211.jpg
inflating: /content/dataset/train/MildDemented/mildDem212.jpg
inflating: /content/dataset/train/MildDemented/mildDem213.jpg
inflating: /content/dataset/train/MildDemented/mildDem214.jpg
inflating: /content/dataset/train/MildDemented/mildDem215.jpg
inflating: /content/dataset/train/MildDemented/mildDem216.jpg
inflating: /content/dataset/train/MildDemented/mildDem217.jpg
```

Activity 2: Dataset Preparation

In preparation for building our deep learning model, we focused on creating distinct training and testing datasets. In our project dataset folder, we observed the existence of pre-arranged training and testing folders, streamlining this crucial step. We assigned variables to the respective folder paths, ensuring a seamless transition into the model development phase. This strategic organization not only simplifies dataset management but also adheres to best practices in machine learning model training.

This meticulous approach sets the stage for the subsequent milestones, where the model will undergo training on the enriched dataset, paving the way for robust Alzheimer's disease detection capabilities.

Activity 1: Importing the libraries

Import the necessary libraries as shown in the image

```
import tensorflow as tf
from tensorflow import keras
from keras import layers
from matplotlib import pyplot as plt
import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
```

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.

- **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.

```
train_datagen=ImageDataGenerator(rescale=1./255)
test_datagen=ImageDataGenerator(rescale=1./255)
```

Activity 4: Splitting into train test split

```
test=test_datagen.flow_from_directory("dataset/test",
                                     classes=classNames,
                                     color_mode="grayscale",
                                     class_mode="categorical",
                                     batch_size=32,
                                     keep_aspect_ratio=True)

Found 1279 images belonging to 4 classes.

train=train_datagen.flow_from_directory("dataset/train",
                                       classes=classNames,
                                       color_mode="grayscale",
                                       class_mode="categorical",
                                       batch_size=32,
                                       keep_aspect_ratio=True)
```

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

In this stage of our project, the focus shifts to constructing an effective model for Alzheimer's disease detection. Instead of leveraging a pre-trained model like Xception, our custom model is designed using the Keras Sequential API, tailored to the specific requirements of our task. The provided code snippet illustrates the architecture of our Convolutional Neural Network (CNN).

The model begins with a sequence of convolutional layers, each followed by a rectified linear unit (ReLU) activation function for non-linearity. Max-pooling layers are interspersed to down-sample spatial dimensions. To combat overfitting, dropout layers are strategically placed, preventing excessive reliance on specific neurons during training. The flattened layer prepares the data for transition to densely connected layers, introducing higher-level features.

Following the convolutional layers, the model incorporates densely connected layers with varying units and ReLU activation functions. The final layer employs a softmax activation function, indicative of the multi-class classification nature of our task. Notably, the choice of activation functions, kernel sizes, and dropout rates is designed to balance the model's complexity and generalization capabilities.

Subsequently, the model is compiled using categorical crossentropy as the loss function, the Adam optimizer for efficient weight updates, and accuracy as the metric for evaluation. The training process is executed on a GPU, enhancing computational efficiency.

It's essential to acknowledge that this approach provides flexibility for fine-tuning and adapting the model architecture based on performance evaluation. As we progress through subsequent milestones, we anticipate iteratively refining the model and enhancing its ability to accurately detect Alzheimer's disease stages in brain MRI images.

```
model=keras.Sequential([
    layers.Conv2D(filters=16,kernel_size=(4,4),activation='relu',use_bias=True,bias_initializer='random_normal'),
    layers.MaxPool2D(strides=(1,1)),
    layers.Conv2D(filters=32,kernel_size=(4,4),activation='relu',use_bias=True,bias_initializer='random_normal'),
    layers.MaxPool2D(strides=(1,1)),
    layers.Conv2D(filters=64,kernel_size=(4,4),activation='relu',use_bias=True,bias_initializer='random_normal'),
    layers.MaxPool2D(strides=(1,1)),
    layers.Dropout(rate=0.25),
    layers.Flatten(),

    layers.Dense(units=32,use_bias=True,bias_initializer='random_normal',activation='relu'),
    layers.Dropout(rate=0.25),
    layers.Dense(units=16,use_bias=True,bias_initializer='random_normal',activation='relu'),
    layers.Dense(units=8,use_bias=True,bias_initializer='random_normal',activation='relu'),
    layers.Dense(units=4,use_bias=True,bias_initializer='random_normal',activation='softmax')
])
```

```
with tf.device("/GPU:0"):
    model.compile(loss="categorical_crossentropy",optimizer="adam",metrics="accuracy")
```

```
history=0
with tf.device("/GPU:0"):
    history=model.fit(train,steps_per_epoch=161,epochs=5,validation_data=test,validation_steps=40)
```

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=0
with tf.device("/GPU:0"):
    history=model.fit(train,steps_per_epoch=161,epochs=5,validation_data=test,validation_steps=40)
```

```
Epoch 1/5
161/161 [=====] - 44s 275ms/step - loss: 0.1732 - accuracy: 0.9459 - val_loss: 4.3481 - val_accuracy: 0.6013
Epoch 2/5
161/161 [=====] - 44s 274ms/step - loss: 0.1687 - accuracy: 0.9490 - val_loss: 5.0139 - val_accuracy: 0.5801
Epoch 3/5
161/161 [=====] - 44s 274ms/step - loss: 0.1615 - accuracy: 0.9518 - val_loss: 4.5455 - val_accuracy: 0.5966
Epoch 4/5
161/161 [=====] - 44s 272ms/step - loss: 0.1627 - accuracy: 0.9520 - val_loss: 4.3783 - val_accuracy: 0.5872
Epoch 5/5
161/161 [=====] - 44s 273ms/step - loss: 0.1588 - accuracy: 0.9529 - val_loss: 4.1747 - val_accuracy: 0.5504
```

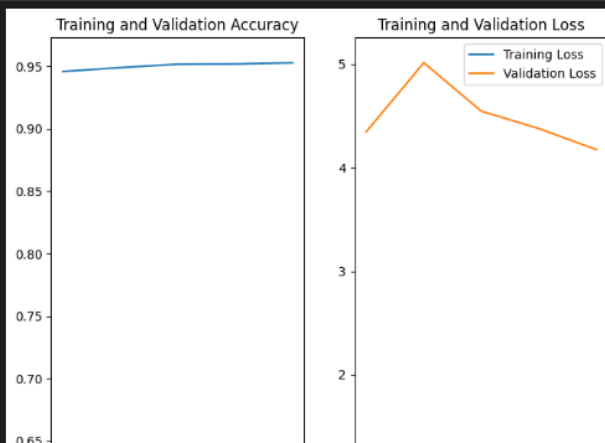
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(5)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Activity 5: Save the Model

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

```
model.save("/content/drive/MyDrive/Colab Notebooks/models/alzheimer_detection.h5")
```

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.

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

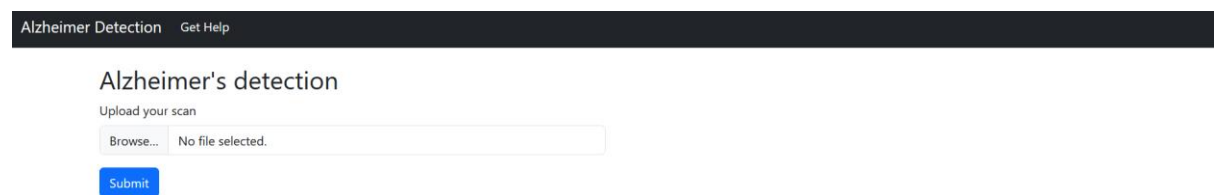
- model prediction using flask
- Building python code

Activity1: html templates:

For this project create one HTML file namely

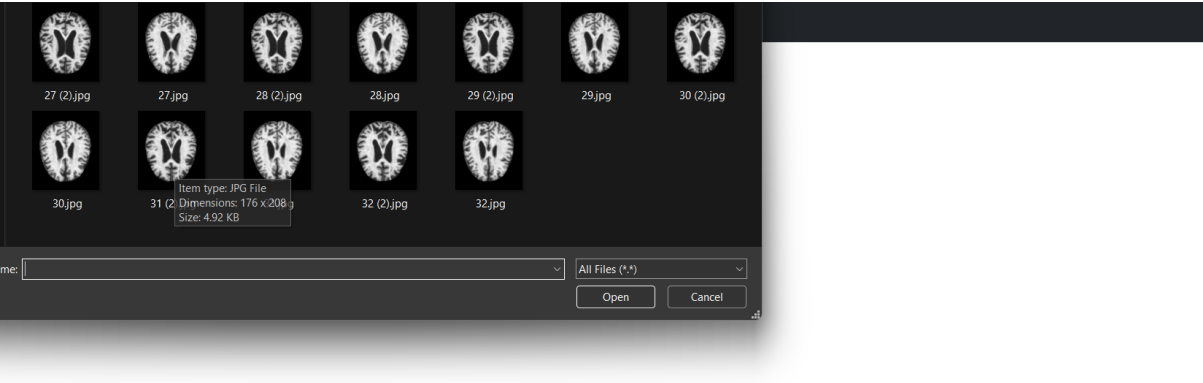
- index.ejs

Let's see how our index.ejs page looks like:



The screenshot shows a web application interface for Alzheimer's detection. At the top, there is a dark navigation bar with the text "Alzheimer Detection" and a link "Get Help". Below the navigation bar, the main heading is "Alzheimer's detection". Underneath the heading, there is a section titled "Upload your scan". This section contains a file upload interface with a "Browse..." button and a text field that says "No file selected.". Below the file upload interface, there is a blue "Submit" button.

Output:



Alzheimer Detection Get Help

Alzheimer's detection

Upload your scan

Browse... 32 (2).jpg

Submit

You have moderate dementia

Caring for someone with moderate dementia requires increased patience and adaptability. Establishing a clear routine remains important, but flexibility becomes crucial as cognitive challenges intensify. Providing more hands-on assistance with daily activities becomes necessary, and creating a safe environment becomes paramount to prevent accidents. Communication may become more challenging, requiring simplified language and increased support. Memory aids and engaging activities should be tailored to the individual's evolving abilities. Social interaction remains valuable, though the caregiver may need to take a more active role. Seeking regular guidance from healthcare professionals becomes even more critical, and joining support groups for caregivers can offer additional insights and emotional support. As the caregiving demands escalate, maintaining one's well-being through self-care becomes an even more vital aspect of the caregiving journey.

Alzheimer's Report

Category	Value
mild_demented	0.0
moderate_demented	1.0
non_demented	0.0
very_mild_demented	0.0

nodeJS server:

```
const express = require("express");
const app = express();
const path = require("path");
const ejsMate = require("ejs-mate");
const methodOverride = require("method-override");
const multer = require("multer");

const storage = multer.memoryStorage();
const upload = multer({ storage: storage });

const modelPath = "file://models/alzheimer_detection/model.json";

app.use(express.urlencoded({ extended: true }));
app.use(express.json());
app.use(methodOverride("_method"));
app.use(express.static(path.join(__dirname, "public")));

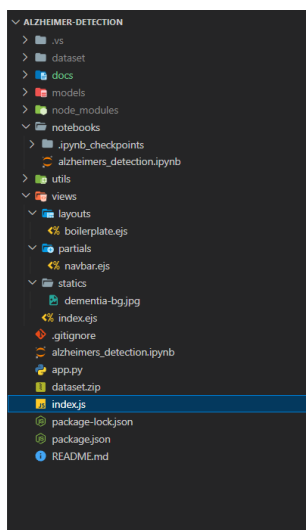
app.engine("ejs", ejsMate);

app.set("view engine", "ejs");
app.set("views", path.join(__dirname, "views"));

app.listen(8080, () => {
  console.log("listening on 8080");
});

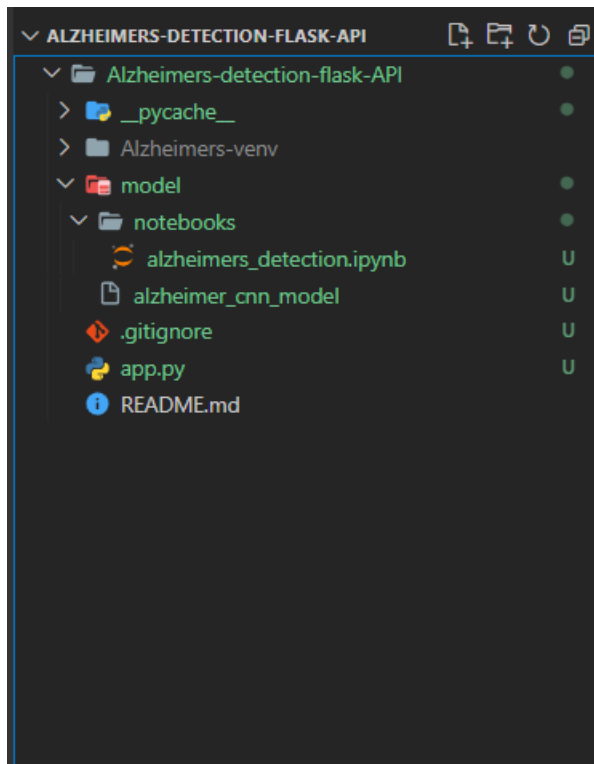
app.get("/", (req, res) => {
  res.render("index");
});
```

Structure:



Python flask API:

Structure:



Code:

```
from PIL import Image
import io
from flask import Flask, request, jsonify
from flask_cors import CORS
import numpy as np
import tensorflow as tf
MODEL_PATH="model/alzheimer_cnn_model"
app = Flask(__name__)
CORS(app)

# Load your TensorFlow model
# model_path = 'model/alzheimer_detection.h5'
# model = tf.keras.models.load_model(model_path)

# def preprocess_image(image, target_size=(224, 224)):
#     # Load the image, resize it to the target size, and convert it to an
#     array
#     img = tf.keras.preprocessing.image.load_img(image,
# target_size=target_size)
#     img_array = tf.keras.preprocessing.image.img_to_array(img)
#     # Expand the dimensions to match the model's expected input shape
#     img_array = np.expand_dims(img_array, axis=0)
```

```

#     # Preprocess the image (normalize values)
#     return tf.keras.applications.mobilenet.preprocess_input(img_array)

def image_to_buffer(image):
    buffer = io.BytesIO()
    image.save(buffer)
    return buffer.getvalue()

def resize_img(img_buffer, size=(255, 255)):
    try:
        # Ensure the image format is compatible with PIL
        pil_image =
Image.open(io.BytesIO(img_buffer)).resize(size).convert('RGB')
    except Exception as e:
        print(f"Error opening image: {e}")
        return None

    # Convert grayscale to RGB

    numpy_image = tf.keras.preprocessing.image.img_to_array(pil_image)
    tensor = tf.convert_to_tensor([numpy_image], dtype=tf.float32)

    return tensor

def image_to_tensor(image):
    image_buffer=image_to_buffer(image)
    tensor= resize_img(image_buffer,(176,176))
    return tensor

def predict(model_path,input_tensor):
    model = tf.keras.models.load_model(model_path, compile=False)
    predictions=model.predict(input_tensor)
    print(predictions)
    return predictions

@app.route("/",methods=['GET'])
def home():
    return jsonify({'msg':'You will do it'})
@app.route('/get-result', methods=['POST'])
def handle_api():
    image=request.files['image']

    tensor=image_to_tensor(image)
    print(tensor.shape)
    predictions=predict(MODEL_PATH,tensor)[0]
    predictions = [float(value) for value in predictions]
    print(predictions)

```

```

    predictions={
      "mild_demented":predictions[0],
      "moderate_demented":predictions[1],
      "non_demented":predictions[2],
      "very_mild_demented":predictions[3]
    }
    return jsonify({'class': predictions})

if __name__ == '__main__':
    app.run(debug=True, port=5000, host="localhost")

```

index.ejs (HTML template):

```

<% layout('layouts/boilerplate') -%>

<div class="row">
  <div class="col-6">
    <h2 class="h2">Alzheimer's detection</h2>
    <form id="alzheimerForm" action="/blah">
      <div class="mb-3">
        <label for="imageUpload" class="form-label">Upload your
scan</label>
        <input type="file" class="form-control" id="imageUpload"
name="imageUpload"
placeholder="name@example.com">
      </div>
      <button class="btn btn-primary">Submit</button>
    </form>
    <div class=" mt-3" id="reportSummary">

    </div>
  </div>
  <div class="col-6">
    <canvas id="reportCard">

    </canvas>
  </div>
</div>
<script>
  const form = document.querySelector("#alzheimerForm")
  const imageScan = document.querySelector("#imageUpload")

  const uploadScan = async (file, url) => {
    if (!file) {
      //do something

      return null
    }
  }

```



```

    }
    const formData = new FormData()
    formData.append('image', file)

    const data = fetch(url, {
      method: 'POST',
      body: formData
    })
      .then(res => res.json())
      .then(res => {
        console.log(res)
        return res
      })
      .catch((err) => {
        console.log(`Error: ${err}`)
      })

    return data
  }

  const generateReportPieChart = (labels, data, idSelector, colours) =>
  {
    const reportCard = new Chart(idSelector, {
      type: "bar",
      data: {
        labels: labels,
        datasets: [{
          label: "Alzheimer's Report",
          data: data,
          backgroundColor: colours,
          hoverOffset: 4
        }]
      },
      options: {
        scales: {
          y: {
            beginAtZero: true,
            max: 100,
            ticks: {
              callback: function (value) {
                return value + "%";
              }
            }
          }
        }
      }
    });
    return reportCard;
  }

```

```

};

form.addEventListener('submit', async (evt) => {
  const url = "http://localhost:5000/get-result"
  evt.preventDefault()
  const file = imageScan.files[0]
  let result = await uploadScan(file, url).then(data => { return
data })

  result = result.class
  const keys = Object.keys(result)
  const values = Object.values(result)
  const colours = ["#ee9b00", "#d00000", "#99d98c", "#94d2bd"]
  const resultIndex = values.indexOf(Math.max(...values))
  const reportCard = generateReportPieChart(keys, values,
"reportCard", colours)

  if (result) {
    const text = document.querySelector("#reportSummary")
    const h2 = document.createElement("h2")
    const p = document.createElement("p")
    p.classList.add(["lead", "text-center"])
    h2.classList.add("h2")
    let msg = ""
    switch (keys[resultIndex]) {
      case keys[0]:
        //generate report for mild demented
        msg = "Caring for someone with mild dementia involves
creating a stable routine and using clear communication. Providing support
with daily tasks while ensuring a safe environment is essential. Encouraging
the use of memory aids and engaging in suitable activities helps maintain a
sense of normalcy. Social interaction is important, and patience is key when
adapting to changes in behavior. Seeking professional guidance and joining
support groups for caregivers can provide valuable insights. Lastly,
prioritizing self-care is crucial for effectively navigating the challenges of
caregiving."

        h2.innerText = "You have mild dementia"
        p.innerText = msg
        text.append(h2, p)
        break
      case keys[1]:
        //generate report for moderate demented
        msg = "Caring for someone with moderate dementia
requires increased patience and adaptability. Establishing a clear routine
remains important, but flexibility becomes crucial as cognitive challenges
intensify. Providing more hands-on assistance with daily activities becomes
necessary, and creating a safe environment becomes paramount to prevent
accidents. Communication may become more challenging, requiring simplified
language and increased support. Memory aids and engaging activities should be

```

tailored to the individual's evolving abilities. Social interaction remains valuable, though the caregiver may need to take a more active role. Seeking regular guidance from healthcare professionals becomes even more critical, and joining support groups for caregivers can offer additional insights and emotional support. As the caregiving demands escalate, maintaining one's well-being through self-care becomes an even more vital aspect of the caregiving journey."

```
        h2.innerText = "You have moderate dementia"
        p.innerText = msg
        text.append(h2, p)
        break
    case keys[2]:
        //generate report for non demented
        msg = "You do not have Dementia"
        h2.innerText = "You have moderate dementia"
        p.innerText = msg
        text.append(h2, p)
        break
    case keys[3]:
        //generate report for very mild demented
        msg = "Caring for someone with very mild dementia
involves subtle adjustments to support their cognitive well-being.
Establishing a consistent routine can help create a sense of stability. Clear
and simple communication remains important, along with providing gentle
assistance with daily tasks as needed. Creating a safe environment involves
minimal adjustments at this stage. Memory aids and engaging activities should
be tailored to maintain cognitive stimulation. Social interaction remains
beneficial, and patience is key as the person may occasionally struggle with
memory. Regular check-ins with healthcare professionals help monitor and
manage any changes. Caregivers should still prioritize self-care to navigate
the challenges of supporting someone with very mild dementia effectively."
        h2.innerText = "You have moderate dementia"
        p.innerText = msg
        text.append(h2, p)
        break
    }

    }
    console.log(finalResult)
})
</script>
```

Links:

- [Github Repo](#)
- [Youtube demo](#)

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

Date	17 NOVEMBER 2023
Team ID	PNT2022TMIDxxxxxx
Project Name	Project - xxx
Maximum Marks	4 Marks

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

Example: Order processing during pandemics for offline mode

Reference: <https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/>

Guidelines:

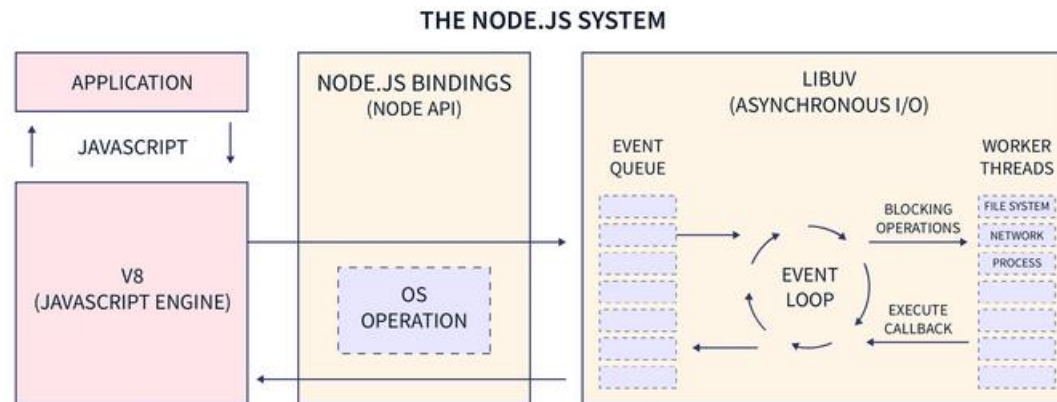


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	User can access our frontend and upload an image of the scan and get the results	HTML(EJS), Bootstrap, JavaScript, ChartJS
2.	NodeJS	User's webpages (i.e., templates) are handled and routed by using expressJS and nodeJS	JavaScript, expressJS, nodeJS
3.	Flask	Model predictions are done on flaskJS server	Flask, TensorflowJS
4.	chartJS API	Charts to render the results	chartJS
5.	CNN	Image pattern detection among trained scans using keras layers and SMOTE principle to increase efficiency	Pattern Recognition model

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Bootstrap	Bootstrap is a frontend styling library which maximises the efficiency and minimises the CSS coding	Bootstrap v5.0
2.	Security Implementations	Uses https protocol for more secure information transfer	https
3.	Scalable Architecture	This can be scaled as an external independent API as it does not depend on Database which makes it easily accessible	Flask and nodeJS server

S.No	Characteristics	Description	Technology
4.	EJS	EJS is a html templating engine which increases the efficiency and reduces the cluster and repetition of html blocks and uses Javascript logic to template the HTML	EJS

References:

<https://getbootstrap.com/docs/5.1/getting-started/introduction/>

<https://ejs.co/#docs>

<https://expressjs.com/en/5x/api.html>

<https://flask.palletsprojects.com/en/3.0.x/>

https://www.tensorflow.org/api_docs