# Cognitive Care: Early Intervention For Alzheimer's Disease

# Final Project Report

**Team Leader : Rehan Suhail**

**Team member : Riman Ghosh**

**Team member : Kanish Khetarpal**

**Team member : Siddharth Nadimetla**

# 1. Introduction

## 1.1. Project Overview

The project aims to develop and optimize deep learning models for the early diagnosis of Alzheimer's Disease using MRI scans. Leveraging advanced machine learning techniques, the project focuses on enhancing accuracy and reliability in identifying different stages of Alzheimer's progression from medical imaging data. By integrating state-of-the-art models and rigorous preprocessing techniques, the goal is to create a robust diagnostic tool that can assist medical professionals in early intervention and treatment planning.

## 1.2. Objectives

The main objectives of the project include:

1. **Model Development**: Build and optimize deep learning models capable of accurately classifying MRI scans into various stages of Alzheimer's Disease.
2. **Data Preprocessing**: Implement comprehensive data preprocessing techniques including normalization, augmentation, and handling of class imbalances to improve model performance.
3. **Validation and Optimization**: Validate model efficacy through rigorous testing and optimization, ensuring high accuracy and generalization on unseen data.
4. **Deployment Readiness**: Prepare the final model for deployment, ensuring it meets operational requirements and can be integrated into clinical settings effectively.
5. **Impact Assessment**: Evaluate the potential impact of the developed model on clinical practices, particularly in enhancing early detection and intervention for Alzheimer's Disease.

By achieving these objectives, the project aims to contribute significantly to the field of medical imaging and AI-driven healthcare solutions, specifically in the context of Alzheimer's Disease diagnosis and management.

# 2. Project Initialization and Planning Phase

## 2.1. Define Problem Statement

**Alzheimer's Disease (AD)** affects millions of people, with early symptoms often overlooked or misdiagnosed, leading to delays in critical intervention. This delay is particularly problematic in underserved areas, where there is a lack of awareness and resources for early detection and treatment. Stigma and misconceptions about AD further hinder timely diagnosis.

Effective early intervention strategies are essential to slow the progression of the disease, enhance cognitive function, and improve the quality of life for those at risk of or in the early stages of AD.

**Problem Statements:**

**Customer**: 65-year-old male engineer

- **I am trying to**: Maintain independence
- **But**: I have a lack of awareness
- **Because**: There is social stigma
- **Which makes me feel**: Anxious

## 2.2. Project Proposal (Proposed Solution)

### Project Overview

**Objectives:**

- Early Diagnosis
- Awareness
- Personalized Care Plans
- Support Services
- Monitoring and Feedback

**Scope:** The project focuses on developing online tools, educational materials, and personalized care plans for early Alzheimer's intervention, excluding direct medical treatment and in-person services.

### Problem Statement

The project addresses the lack of early detection and personalized intervention for Alzheimer's

Disease to improve patient outcomes and quality of life.

### Impact

Solving the problem could significantly delay disease progression, preserve cognitive function, and enhance overall quality of life for individuals affected by Alzheimer's Disease.

### Proposed Solution

**Approach:** Utilize agile development methodologies for iterative refinement of online tools, educational materials, care plans, support service integration, and monitoring systems in collaboration with stakeholders and healthcare experts.

**Key Features:** The proposed solution uniquely integrates advanced technology with personalized care plans and community engagement to enhance early intervention and support for Alzheimer's Disease.

### Resource Requirements

| Resource Type | Description | Specification/Allocation |
|---|---|---|
| Hardware | Computing Resources | CPU/GPU specifications, number of cores T4 GPU |
| Memory | RAM specifications | 16 GB |
| Storage | Disk space for data, models, and logs | 512 GB SSD |
| Software | Frameworks | Python frameworks Flask, Django, Pandas, Numpy |
| Libraries | Additional libraries | TensorFlow, Scikit-learn |
| Development Environment | IDE, version control | Jupyter Notebook, Git, Google Collab, Spyder |
| Data | Data Source, size, format | Kaggle dataset, 6400 images |

## 2.3. Initial Project Planning

**Sprint-1: Data Collection**

- **Functional Requirement (Epic)**: Data Collection
- **User Story Number**: USN-1

- **User Story / Task**: Collect images of brain MRI and organize into sub directories based on their respective names. Create folders of types of Alzheimer.
- **Story Points**: 8
- **Priority**: Medium
- **Team Members**: Rehan
- **Sprint Start Date**: 2024-07-01
- **Sprint End Date (Planned)**: 2024-07-02

**Sprint-1: Image Preprocessing**

- **Functional Requirement (Epic)**: Image Preprocessing
- **User Story Number**: USN-2
- **User Story / Task**: Importing required libraries, configuration of the images, handling imbalanced data.
- **Story Points**: 5
- **Priority**: Medium
- **Team Members**: Rehan
- **Sprint Start Date**: 2024-07-02
- **Sprint End Date (Planned)**: 2024-07-03

**Sprint-2: Model Development**

- **Functional Requirement (Epic)**: Model Development
- **User Story Number**: USN-3
- **User Story / Task**: Developing and training the model.
- **Story Points**: 5
- **Priority**: Medium
- **Team Members**: Rehan
- **Sprint Start Date**: 2024-07-03
- **Sprint End Date (Planned)**: 2024-07-05

**Sprint-3: Model Tuning and Testing**

- **Functional Requirement (Epic)**: Model Tuning and Testing
- **User Story Number**: USN-4
- **User Story / Task**: Model testing with different datasets and checking for any possible errors.
- **Story Points**: 8
- **Priority**: High
- **Team Members**: Rehan
- **Sprint Start Date**: 2024-07-05

- **Sprint End Date (Planned)**: 2024-07-07

**Sprint-4: Application Building**

- **Functional Requirement (Epic)**: Application Building
- **User Story Number**: USN-5
- **User Story / Task**: Building HTML pages, building Flask code, and running the application.
- **Story Points**: 5
- **Priority**: High
- **Team Members**: Rehan, Siddharth
- **Sprint Start Date**: 2024-07-08
- **Sprint End Date (Planned)**: 2024-07-09

**Sprint-5: Documentation and Report**

- **Functional Requirement (Epic)**: Documentation and Report
- **User Story Number**: USN-6
- **User Story / Task**: Documenting appropriate templates and making a report.
- **Story Points**: 8
- **Priority**: High
- **Team Members**: Kanish, Riman, Rehan
- **Sprint Start Date**: 2024-07-09
- **Sprint End Date (Planned)**: 2024-07-10

# 3. Data Collection and Preprocessing Phase

## 3.1. Data Collection Plan and Raw Data Sources Identified

The Data Collection Plan and Raw Data Sources Identification are crucial for effective data training strategies. These components involve strategic planning to identify and gather relevant data sources systematically. By ensuring the integrity and comprehensive coverage of data, organizations can enhance the accuracy and reliability of their training datasets, leading to more informed decision-making and robust model performance in data-driven analyses and applications.

## Data Collection Plan

### Project Overview

The machine learning project aims to develop robust Alzheimer's Disease diagnostic models using MRI scans. Its objective is to leverage comprehensive data collection and rigorous training strategies to enhance model accuracy and support informed medical decision-making.

### Data Collection Plan

The data for this project was collected from Kaggle, a popular platform for datasets and machine learning competitions.

### Raw Data Sources Identified

The raw data sources for this project include JPEG images sourced from Kaggle, categorized into four classes. These images represent various stages of Alzheimer's Disease progression.

## Raw Data Sources Report

**Source Name**: Kaggle Dataset

**Description**: The data source comprises JPEG images from Kaggle representing MRI scans categorized into four classes: MildDemented, ModerateDemented, NonDemented, and VeryMildDemented, each depicting different stages of Alzheimer's Disease progression.

**Location/URL**: Kaggle Alzheimer's Dataset

**Format**: JPEG

**Size**: 33.0 MB

**Access Permissions**: Public

# 3.2. Data Quality Report

The Data Quality Report summarizes data quality issues from the selected source, including severity levels and resolution plans. It aids in systematically identifying and rectifying data discrepancies.

### Data Source: Kaggle Dataset

**Data Quality Issues:**

1. **Class imbalance among Alzheimer's Disease classes**
   - **Severity**: Moderate
   - **Resolution Plan**: Implement SMOTE (Synthetic Minority Over-sampling Technique) to generate synthetic samples for minority classes, ensuring balanced representation during model training.
2. **Inconsistent image quality and resolution across MRI scans**
   - **Severity**: High
   - **Resolution Plan**: Standardize image preprocessing techniques such as resizing to a uniform resolution, applying denoising filters, and ensuring consistent contrast adjustment to enhance image clarity and consistency.

## 3.3. Data Preprocessing

The data exploration and preprocessing approach for Alzheimer's Disease involves a series of steps aimed at enhancing data quality, facilitating model generalization, and optimizing neural network training through advanced computer vision techniques.

### Data Overview

The project utilizes Alzheimer's MRI scans sourced from Kaggle, categorized into four classes.

### Resizing

Resizing to 180x180 pixels ensures consistent input dimensions for the Xception model as per the code's preprocessing requirements.

### Normalization

Normalization involves scaling pixel values to a range between 0 and 1, ensuring a consistent data range across images to facilitate effective model training and convergence.

### Data Augmentation

Data augmentation techniques such as brightness adjustment, zooming, and horizontal flipping diversify training data, thereby boosting model robustness in Alzheimer's Disease pattern recognition.

### Data Balancing

To address class imbalance, SMOTE (Synthetic Minority Over-sampling Technique) is employed to generate synthetic samples, improving model accuracy across all Alzheimer's Disease categories.

### Transfer Learning

Transfer learning utilizes a pre-trained Xception model with frozen layers to enhance Alzheimer's Disease classification performance, leveraging knowledge from previously trained models.

### Batch Normalization

Batch normalization stabilizes training by normalizing input batches, enhancing model convergence and generalization.

# 4. Model Development Phase

## 4.1. Model Selection Report

#### Xception

The Xception (Extreme Inception) model is a deep Convolutional neural network architecture that builds upon the Inception model. It replaces the standard Inception modules with depth-wise separable convolutions, significantly reducing the number of parameters and computations while maintaining high accuracy.

#### VGG19

VGG19 is a deep Convolutional neural network architecture consisting of 19 layers. It was introduced by the Visual Geometry Group at the University of Oxford. Known for its simplicity and depth, VGG19 uses small 3x3 convolution filters throughout the network, enhancing its ability to learn complex features.

#### Inception V3

Inception V3 is a deep Convolutional neural network architecture introduced by Google as part of the Inception family. It incorporates advanced techniques such as factorized convolutions, aggressive regularization, and label smoothing to enhance model performance and reduce computational cost.

## 4.2. Initial Model Training Code, Model Validation and Evaluation Report

### Xception Model

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

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_dim_ordering_tf_kernels_notop.h5
83683744/83683744 [==============================] - 5s 0us/step

[8] for layer in xcep_model.layers:
        layer.trainable = False
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SeparableConv2D, BatchNormalization, GlobalAveragePooling2D , Dropout
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")
```

```
[10] custom_inception_model.compile(
         loss='categorical_crossentropy',
         optimizer='adam',
         metrics=['accuracy']
     )

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

### VGG19 Model

```
[6] from tensorflow.keras.applications import VGG19

    vgg_model = VGG19(weights='imagenet', include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, 3))

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80134624/80134624 [==============================] - 4s 0us/step
```

```
[7] for layer in vgg_model.layers:
        layer.trainable = False
```

```
model = Sequential([
    vgg_model,
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dense(4, activation='softmax')
])
```

```
[7] model.compile(
        optimizer='adam',
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
```

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

## InceptionV3 model

```
from tensorflow.keras.applications import InceptionV3

inception_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, 3))

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87910968/87910968 [==============================] - 5s 0us/step
```

```
[5] for layer in inception_model.layers:
        layer.trainable = False
```

```
[6] model = Sequential([
        inception_model,
        Flatten(),
        Dense(512, activation='relu'),
        Dropout(0.5),
        Dense(256, activation='relu'),
        Dropout(0.5),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(64, activation='relu'),
        Dense(4, activation='softmax')
    ])
```

```
[7] model.compile(
        optimizer='adam',
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
```

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

## Model Validation and Evaluation Report :

## Xception Model

```
Model: "inception_cnn_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 xception (Functional)       (None, 6, 6, 2048)        20861480

 dropout (Dropout)           (None, 6, 6, 2048)        0

 global_average_pooling2d (  (None, 2048)              0
 GlobalAveragePooling2D)

 flatten (Flatten)           (None, 2048)              0

 batch_normalization_4 (Bat  (None, 2048)              8192
 chNormalization)

 dense (Dense)               (None, 512)               1049088

 batch_normalization_5 (Bat  (None, 512)               2048
 chNormalization)

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 batch_normalization_6 (Bat  (None, 256)               1024
 chNormalization)

 dropout_2 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 128)               32896

 batch_normalization_7 (Bat  (None, 128)               512
```

```
 dropout_3 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 64)                8256

 dropout_4 (Dropout)         (None, 64)                0

 batch_normalization_8 (Bat  (None, 64)                256
 chNormalization)

 dense_4 (Dense)             (None, 4)                 260

=================================================================
Total params: 22095340 (84.29 MB)
Trainable params: 1227844 (4.68 MB)
Non-trainable params: 20867496 (79.60 MB)
_________________________________________________________________
```

```
Epoch 15/30
205/205 [==============================] - 23s 112ms/step - loss: 0.4927 - accuracy: 0.7987 - val_loss: 0.4612 - val_accuracy: 0.7980
Epoch 16/30
205/205 [==============================] - 24s 116ms/step - loss: 0.4938 - accuracy: 0.8051 - val_loss: 0.4533 - val_accuracy: 0.8121
Epoch 17/30
205/205 [==============================] - 24s 117ms/step - loss: 0.4521 - accuracy: 0.8186 - val_loss: 0.4391 - val_accuracy: 0.8109
Epoch 18/30
205/205 [==============================] - 24s 117ms/step - loss: 0.4542 - accuracy: 0.8222 - val_loss: 0.4384 - val_accuracy: 0.8164
Epoch 19/30
205/205 [==============================] - 24s 117ms/step - loss: 0.4260 - accuracy: 0.8375 - val_loss: 0.4230 - val_accuracy: 0.8237
Epoch 20/30
205/205 [==============================] - 24s 117ms/step - loss: 0.4227 - accuracy: 0.8321 - val_loss: 0.4026 - val_accuracy: 0.8359
Epoch 21/30
205/205 [==============================] - 23s 112ms/step - loss: 0.3990 - accuracy: 0.8443 - val_loss: 0.3936 - val_accuracy: 0.8347
Epoch 22/30
205/205 [==============================] - 23s 112ms/step - loss: 0.4108 - accuracy: 0.8431 - val_loss: 0.3816 - val_accuracy: 0.8456
Epoch 23/30
205/205 [==============================] - 23s 112ms/step - loss: 0.3809 - accuracy: 0.8575 - val_loss: 0.3914 - val_accuracy: 0.8371
Epoch 24/30
205/205 [==============================] - 23s 113ms/step - loss: 0.3702 - accuracy: 0.8599 - val_loss: 0.4047 - val_accuracy: 0.8328
Epoch 25/30
205/205 [==============================] - 24s 117ms/step - loss: 0.3513 - accuracy: 0.8669 - val_loss: 0.3854 - val_accuracy: 0.8426
Epoch 26/30
205/205 [==============================] - 23s 112ms/step - loss: 0.3519 - accuracy: 0.8726 - val_loss: 0.4065 - val_accuracy: 0.8353
Epoch 27/30
205/205 [==============================] - 24s 116ms/step - loss: 0.3281 - accuracy: 0.8772 - val_loss: 0.4077 - val_accuracy: 0.8328
Epoch 28/30
205/205 [==============================] - 23s 113ms/step - loss: 0.3166 - accuracy: 0.8808 - val_loss: 0.3696 - val_accuracy: 0.8530
Epoch 29/30
205/205 [==============================] - 23s 112ms/step - loss: 0.3191 - accuracy: 0.8828 - val_loss: 0.3741 - val_accuracy: 0.8475
Epoch 30/30
205/205 [==============================] - 24s 117ms/step - loss: 0.3252 - accuracy: 0.8797 - val_loss: 0.3685 - val_accuracy: 0.8536
```

## VGG19 Model

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg19 (Functional)          (None, 5, 5, 512)         20024384

 flatten (Flatten)           (None, 12800)             0

 dense (Dense)               (None, 512)               6554112

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 128)               32896

 dropout_2 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 64)                8256

 dense_4 (Dense)             (None, 4)                 260

=================================================================
Total params: 26751236 (102.05 MB)
Trainable params: 6726852 (25.66 MB)
Non-trainable params: 20024384 (76.39 MB)
_________________________________________________________________
```

```
Epoch 18/30
205/205 [==============================] - 28s 137ms/step - loss: 1.3864 - accuracy: 0.2472 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 19/30
205/205 [==============================] - 28s 136ms/step - loss: 1.3866 - accuracy: 0.2492 - val_loss: 1.3863 - val_accuracy: 0.2502
Epoch 20/30
205/205 [==============================] - 28s 136ms/step - loss: 1.3865 - accuracy: 0.2504 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 21/30
205/205 [==============================] - 28s 135ms/step - loss: 1.3864 - accuracy: 0.2532 - val_loss: 1.3863 - val_accuracy: 0.2502
Epoch 22/30
205/205 [==============================] - 33s 159ms/step - loss: 1.3864 - accuracy: 0.2550 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 23/30
205/205 [==============================] - 33s 159ms/step - loss: 1.3866 - accuracy: 0.2440 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 24/30
205/205 [==============================] - 33s 160ms/step - loss: 1.3865 - accuracy: 0.2454 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 25/30
205/205 [==============================] - 28s 136ms/step - loss: 1.3865 - accuracy: 0.2440 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 26/30
205/205 [==============================] - 33s 161ms/step - loss: 1.3863 - accuracy: 0.2510 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 27/30
205/205 [==============================] - 28s 137ms/step - loss: 1.3866 - accuracy: 0.2468 - val_loss: 1.3864 - val_accuracy: 0.2495
Epoch 28/30
205/205 [==============================] - 33s 160ms/step - loss: 1.3864 - accuracy: 0.2503 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 29/30
205/205 [==============================] - 33s 159ms/step - loss: 1.3865 - accuracy: 0.2472 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 30/30
205/205 [==============================] - 33s 159ms/step - loss: 1.3864 - accuracy: 0.2532 - val_loss: 1.3863 - val_accuracy: 0.2495
```

## InceptionV3 model

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 4, 4, 2048)        21802784

 flatten (Flatten)           (None, 32768)             0

 dense (Dense)               (None, 512)               16777728

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 128)               32896

 dropout_2 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 64)                8256

 dense_4 (Dense)             (None, 4)                 260

=================================================================
Total params: 38753252 (147.83 MB)
Trainable params: 16950468 (64.66 MB)
Non-trainable params: 21802784 (83.17 MB)
_________________________________________________________________
```

```
Epoch 18/30
205/205 [==============================] - 13s 64ms/step - loss: 1.3864 - accuracy: 0.2489 - val_loss: 1.3864 - val_accuracy: 0.2495
Epoch 19/30
205/205 [==============================] - 13s 64ms/step - loss: 1.3864 - accuracy: 0.2472 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 20/30
205/205 [==============================] - 13s 64ms/step - loss: 1.3864 - accuracy: 0.2483 - val_loss: 1.3863 - val_accuracy: 0.2502
Epoch 21/30
205/205 [==============================] - 13s 64ms/step - loss: 1.3867 - accuracy: 0.2371 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 22/30
205/205 [==============================] - 13s 63ms/step - loss: 1.3865 - accuracy: 0.2463 - val_loss: 1.3864 - val_accuracy: 0.2502
Epoch 23/30
205/205 [==============================] - 13s 64ms/step - loss: 1.3867 - accuracy: 0.2446 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 24/30
205/205 [==============================] - 13s 63ms/step - loss: 1.3866 - accuracy: 0.2524 - val_loss: 1.3863 - val_accuracy: 0.2502
Epoch 25/30
205/205 [==============================] - 13s 65ms/step - loss: 1.3864 - accuracy: 0.2408 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 26/30
205/205 [==============================] - 13s 64ms/step - loss: 1.3865 - accuracy: 0.2483 - val_loss: 1.3864 - val_accuracy: 0.2495
Epoch 27/30
205/205 [==============================] - 13s 65ms/step - loss: 1.3866 - accuracy: 0.2407 - val_loss: 1.3863 - val_accuracy: 0.2495
Epoch 28/30
205/205 [==============================] - 13s 66ms/step - loss: 1.3866 - accuracy: 0.2455 - val_loss: 1.3864 - val_accuracy: 0.2495
Epoch 29/30
205/205 [==============================] - 13s 64ms/step - loss: 1.3862 - accuracy: 0.2556 - val_loss: 1.3864 - val_accuracy: 0.2508
Epoch 30/30
205/205 [==============================] - 13s 64ms/step - loss: 1.3866 - accuracy: 0.2439 - val_loss: 1.3864 - val_accuracy: 0.2495
```

# 5. Model Optimization and Tuning Phase

## 5.1. Tuning Documentation

he Model Optimization and Tuning Phase involves refining neural network models for peak performance. It includes optimized model code, fine-tuning hyper parameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

### Hyper parameter Tuning Documentation

**Xception**

- **Learning Rate**: Adam optimizer with a default learning rate of 0.001.
- **Batch Size**: 6500 training samples per iteration.
- **Epochs**: 30 complete passes through the training dataset.
- **Dropout Rate**: 0.5 to prevent over fitting.
- **Zoom Range**: Random zoom between 0.99 and 1.01.
- **Brightness Range**: Random brightness adjustment between 0.8 and 1.2.
- **Rescale**: Data normalized by scaling pixel values to 1./255.
- **Global Average Pooling2D**: A pooling layer to reduce the spatial dimensions of the feature maps.

**VGG19**

- **Learning Rate**: Adam optimizer with a default learning rate of 0.001.
- **Batch Size**: 6500 training samples per iteration.
- **Epochs**: 30 complete passes through the training dataset.
- **Dropout Rate**: 0.5 to prevent over fitting.
- **Zoom Range**: Random zoom between 0.99 and 1.01.
- **Brightness Range**: Random brightness adjustment between 0.8 and 1.2.
- **Rescale**: Data normalized by scaling pixel values to 1./255.
- **Conv Block**: Multiple Convolutional layers with small 3x3 filters.

**Inception V3**

- **Learning Rate**: Adam optimizer with a default learning rate of 0.001.
- **Batch Size**: 6500 training samples per iteration.

- **Epochs**: 30 complete passes through the training dataset.
- **Dropout Rate**: 0.5 to prevent overfitting.
- **Zoom Range**: Random zoom between 0.99 and 1.01.
- **Brightness Range**: Random brightness adjustment between 0.8 and 1.2.
- **Rescale**: Data normalized by scaling pixel values to 1./255.
- **Factorized Convolutions**: Use of smaller convolutions like 1x7 and 7x1 to reduce computational cost.

## 5.2. Final Model Selection Justification

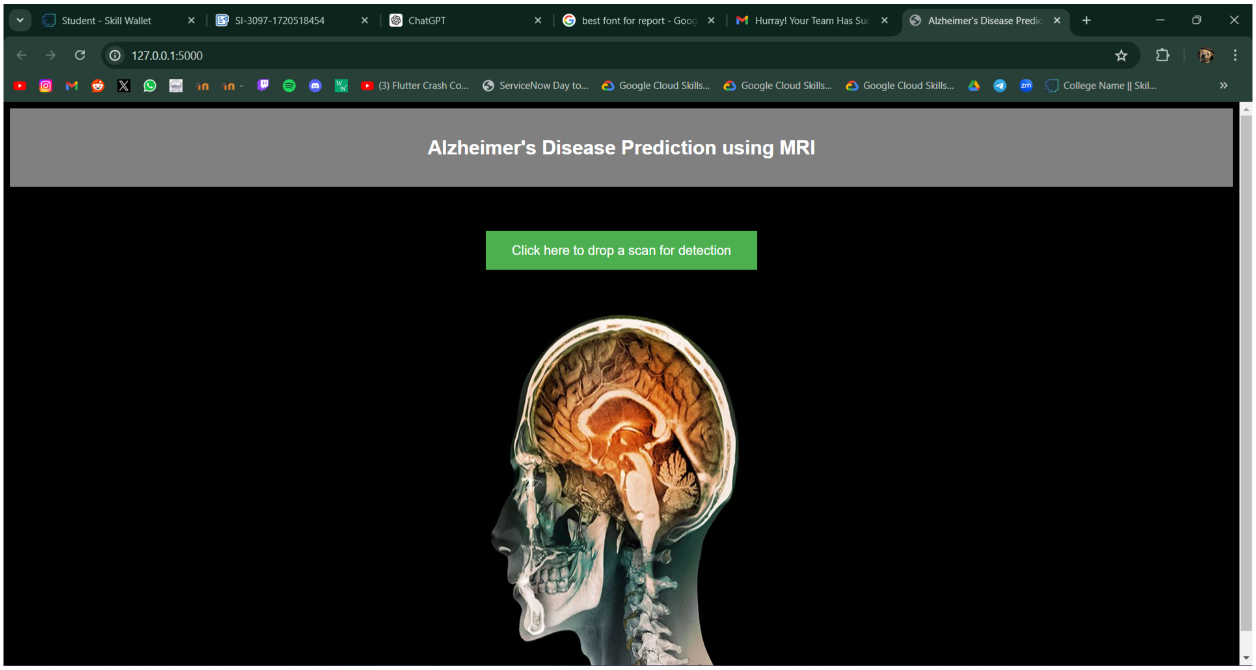
**Xception**

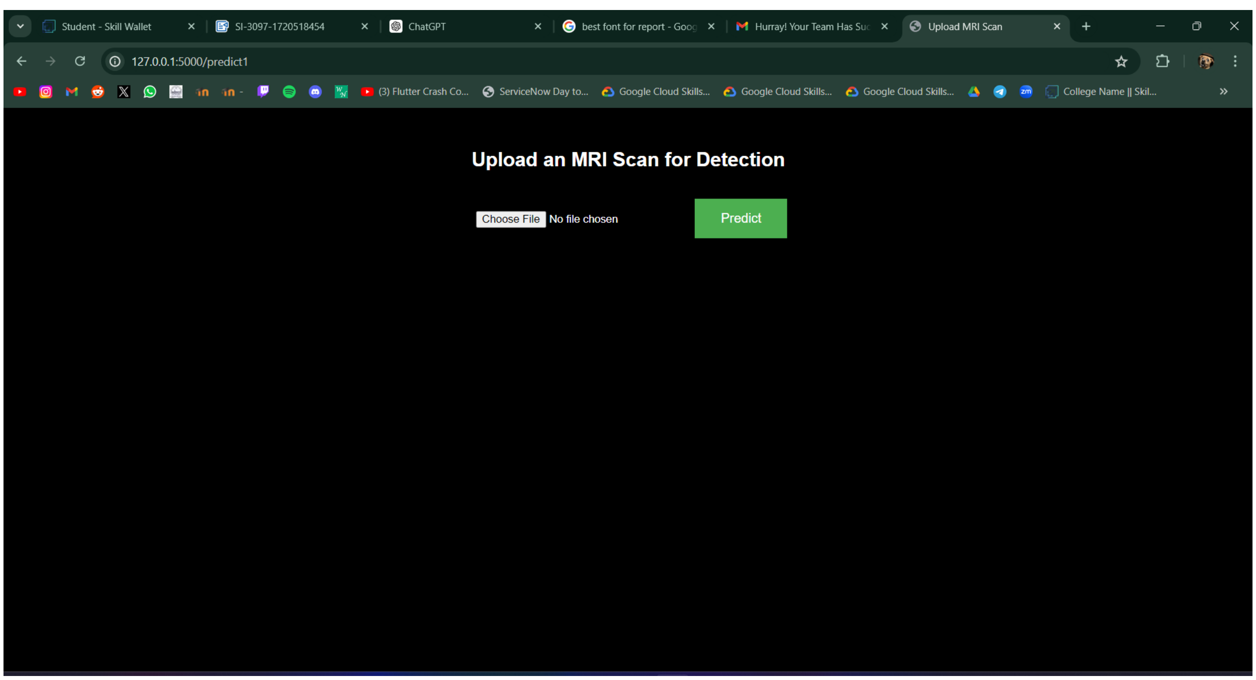
The Xception model was chosen as the final optimized model for several compelling reasons:

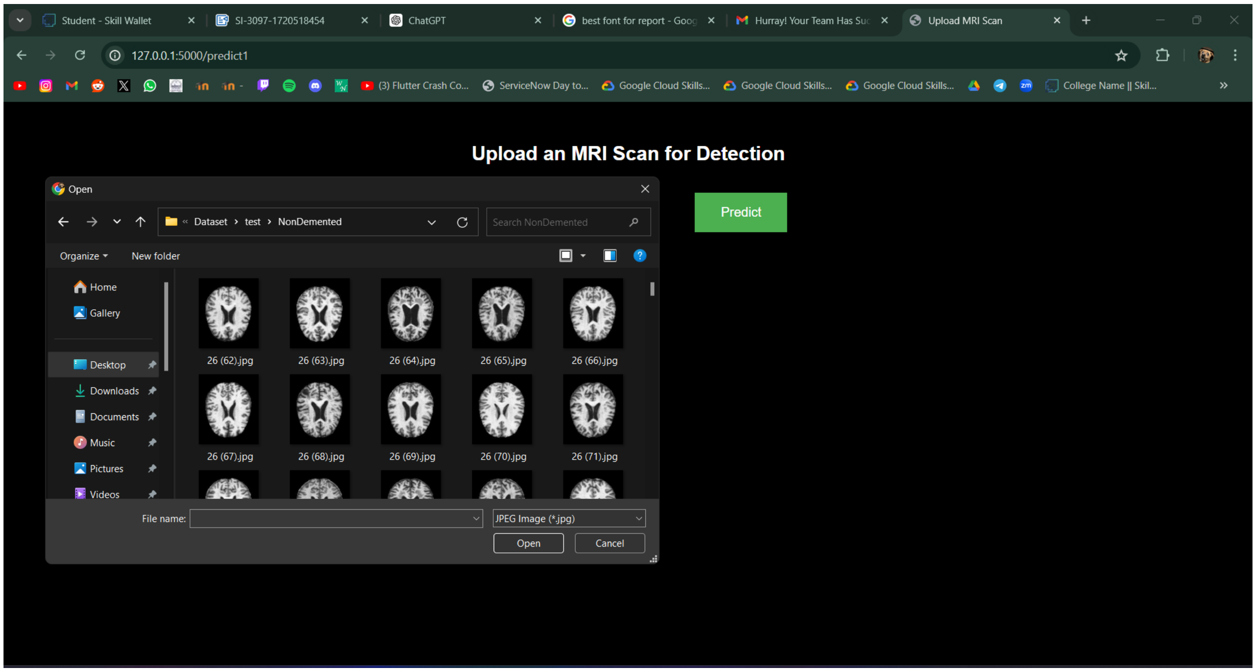
- **Performance**: Throughout 30 epochs of training, the Xception model consistently demonstrated improvement in accuracy and validation metrics.
- **Validation Accuracy**: It achieved a final validation accuracy of 85.36%, indicating strong capability in distinguishing between different classes of Alzheimer's Disease progression.
- **Robust Performance**: The model exhibited robust performance and convergence during training, suggesting it effectively learned relevant features and patterns from the data.
- **Optimization**: Hyperparameter tuning and optimization further enhanced its performance, ensuring it was well-suited for the task of Alzheimer's Disease diagnosis using MRI scans.

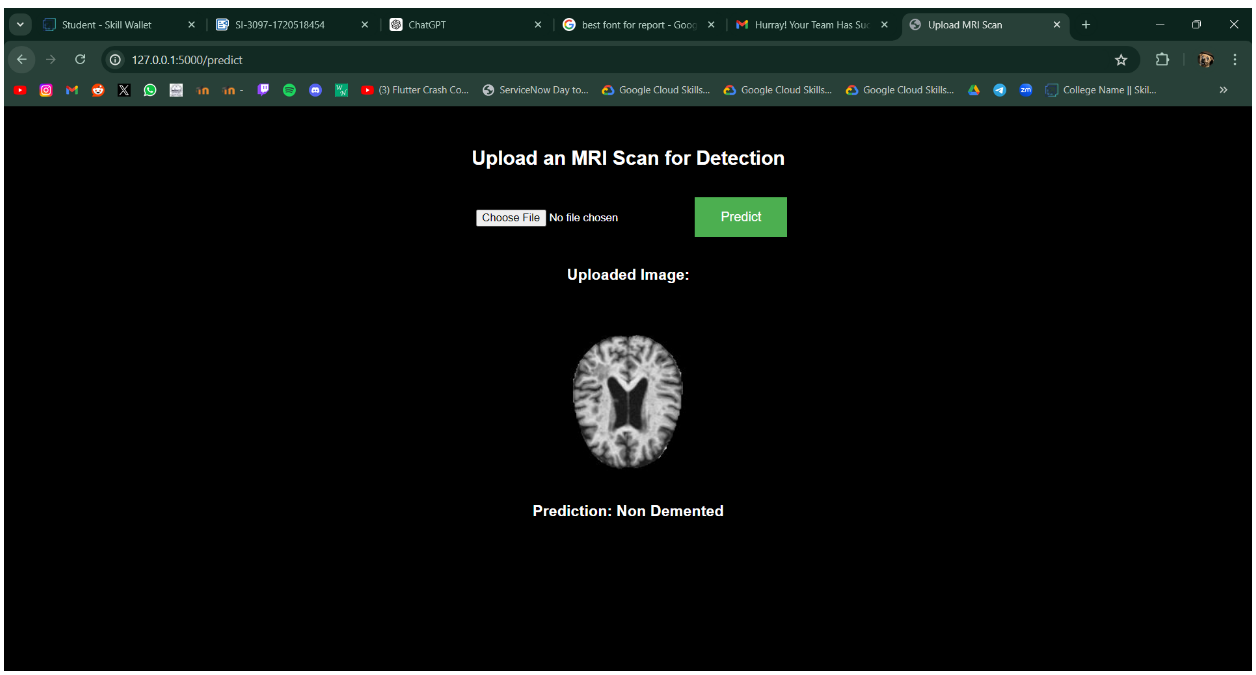
# 6. Results

## 6.1. Output Screenshots




127.0.0.1:5000/predict1
Upload an MRI Scan for Detection
Open
Dataset › test › NonDemented
Organize
New folder
Home
Gallery
Desktop
Downloads
Documents
Music
Pictures
Videos
26 (62).jpg
26 (63).jpg
26 (64).jpg
26 (65).jpg
26 (66).jpg
26 (67).jpg
26 (68).jpg
26 (69).jpg
26 (70).jpg
26 (71).jpg
File name:
JPEG Image (*.jpg)
Open
Cancel
Predict


127.0.0.1:5000/predict
Upload an MRI Scan for Detection
Choose File No file chosen
Predict
Uploaded Image:
Prediction: Non Demented

# 7. Advantages & Disadvantages

## Advantages

1. **High Accuracy**: Achieved a final validation accuracy of 85.36%, indicating superior performance in distinguishing Alzheimer's Disease progression stages.
2. **Effective Learning**: Demonstrated robust learning and convergence during training, suggesting it effectively captured relevant features from the MRI scan data.
3. **Optimized Performance**: Through hyper parameter tuning and optimization, the model's performance was further enhanced, ensuring it meets project requirements effectively.
4. **Efficiency**: Utilizes depth wise separable convolutions to reduce parameters and computational complexity, potentially leading to faster inference times.

## Disadvantages

1. **Complexity**: Implementing and fine-tuning Xception may require more computational resources and expertise compared to simpler models like VGG19.
2. **Overfitting**: Despite dropout regularization (dropout rate of 0.5), there is still a risk of over fitting if not carefully monitored and tuned.
3. **Data Requirements**: Requires sufficient and diverse data for effective training, especially given the complexity of its architecture.

# 8. Conclusion

In conclusion, this project has been dedicated to the development and optimization of deep learning models for diagnosing Alzheimer's Disease using MRI scans.

**Model Selection and Justification**: After thorough evaluation, the Xception model was chosen as the final optimized model due to its consistent performance improvements across 30 epochs, achieving a commendable validation accuracy of 85.36%. It effectively learned to distinguish between different stages of Alzheimer's Disease progression and predictive capability.

**Advantages**: The Xception model stands out for its high accuracy, facilitated by depthwise separable convolutions that reduce computational complexity while maintaining efficacy. Optimized hyperparameters further bolstered its performance, making it a suitable choice for complex diagnostic tasks.

**Challenges**: Implementing and fine-tuning the Xception model required substantial computational resources and expertise. Managing potential over fitting and ensuring sufficient, diverse data were ongoing challenges addressed during the project.

# 9. Future Scope

Looking ahead, there are several avenues for future exploration and enhancement in Alzheimer's Disease diagnosis through deep learning:

1. **Ensemble Methods**: Integrating multiple models or ensemble learning techniques could potentially improve overall diagnostic accuracy and robustness.
2. **Advanced Data Augmentation**: Exploring more sophisticated data augmentation techniques could further enhance model generalization capabilities.
3. **Transfer Learning Variants**: Investigating variations in transfer learning approaches tailored to specific aspects of Alzheimer's Disease progression could yield nuanced insights.
4. **Clinical Integration**: Collaborating with clinical experts to incorporate additional relevant data sources and domain knowledge into model development.
5. **Ethical Considerations**: Continuously addressing ethical implications, such as data privacy and bias mitigation, is essential for deploying AI-driven tools responsibly.

By continuing to innovate and collaborate across disciplines, future advancements in AI for Alzheimer's Disease diagnosis promise to enhance early detection & patient outcomes.

# 10. Appendix

## 10.1. Source Code

### Xception Model Code

```
!unzip '/content/archive.zip'
!pip install Tensorflow
!pip install Keras

import tensorflow as tf
from tensorflow.keras.layers import Dense, Flatten, Dropout, GlobalAveragePooling2D, BatchNormalization
from tensorflow.keras.models import Sequential, Model, load_model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.xception import Xception, preprocess_input
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
import numpy as np
import matplotlib.pyplot as plt

# Paths for training and testing data
trainPath =  r"/content/Alzheimer_s Dataset/train"
testPath = r"/content/Alzheimer_s Dataset/test"

# Image preprocessing parameters
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"

# Data augmentation and loading
work_dr = ImageDataGenerator(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=trainPath, target_size=DIM, batch_size=6500, shuffle=False)
train_data, train_labels = train_data_gen.next()

# SMOTE oversampling for class imbalance
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)

# Train-test-validation 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)

# Xception base model with pretrained weights
xcep_model = Xception(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

```
# Freeze layers in base model
for layer in xcep_model.layers:
    layer.trainable = False

# Custom model on top of Xception
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")

# Compile the model
custom_inception_model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)

# Train the model
history = custom_inception_model.fit(train_data, train_labels, validation_data=(val_data, val_labels), epochs=30)

# Save the model
custom_inception_model.save('/content/adp_xception.h5')

# Load the saved model
model = load_model("/content/adp_xception.h5")

# Function to predict the class of an image
def predict_class(img_path, model):
    img = tf.keras.preprocessing.image.load_img(img_path, target_size=(180, 180))
    x = tf.keras.preprocessing.image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = x / 255.0  # Normalize the image
    preds = model.predict(x)
    pred_class_index = np.argmax(preds, axis=1)
    class_names = ['MildDemented', 'ModerateDemented', 'NonDemented', 'VeryMildDemented']
    predicted_class = class_names[pred_class_index[0]]
    return predicted_class

# Example usage:
img_path = r"/content/Alzheimer_s Dataset/test/MildDemented/26.jpg"
```

```
predicted_class = predict_class(img_path, model)
print(f"Predicted Class: {predicted_class}")
```

# APP.PY

```
import numpy as np
import os
from keras.preprocessing import image
import tensorflow.compat.v1 as tf
from flask import Flask, request, render_template
from werkzeug.utils import secure_filename
from tensorflow.python.keras.backend import set_session
from tensorflow.python.keras.models import load_model

tf.disable_eager_execution()
sess = tf.Session()
tf.disable_v2_behavior()
graph = tf.get_default_graph()

app = Flask(__name__)
set_session(sess)

# Load the model
model = load_model('adp.h5')

@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('alzheimers.html')

@app.route('/predict1', methods=['GET'])
def predict1():
    # Prediction page
    return render_template('alzpre.html')

@app.route('/predict', methods=['POST'])
def upload():
    if request.method == 'POST':
        # Get the file from the post request
        f = request.files['image']

        # Save the file to /uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(basepath, 'static', 'uploads', secure_filename(f.filename))
        f.save(file_path)

        # Preprocess the image to the size expected by the model
        img = image.load_img(file_path, target_size=(128, 128))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)

        # Make prediction
        with graph.as_default():
            set_session(sess)
            prediction = model.predict(x)[0]
            print(prediction)
```

```python
        # Determine prediction text based on your model's output
        prediction_class = np.argmax(prediction)
        if prediction_class == 0:
            text = "Mild Demented"
        elif prediction_class == 1:
            text = "Moderate Demented"
        elif prediction_class == 2:
            text = "Non Demented"
        else:
            text = "Very Mild Demented"

        return render_template('alzpre.html', result=text, image_path='static/uploads/' + secure_filename(f.filename))

if __name__ == "__main__":
    if not os.path.exists('static/uploads'):
        os.makedirs('static/uploads')
    app.run(debug=True)
```

## alzheimers.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Alzheimer's Disease Prediction using MRI</title>
    <style>
        body {
            background-color: black;
            color: white;
            font-family: Arial, sans-serif;
        }
        nav {
            background-color: gray;
            padding: 15px;
            text-align: center;
        }
        .container {
            text-align: center;
            margin-top: 50px;
        }
        .button {
            background-color: #4CAF50;
            border: none;
            color: white;
            padding: 15px 32px;
            text-align: center;
            text-decoration: none;
            display: inline-block;
            font-size: 16px;
            margin: 4px 2px;
            cursor: pointer;
        }
        .image-container {
            margin-top: 20px;
        }
    </style>
```

```
</head>
<body>
  <nav>
    <h1>Alzheimer's Disease Prediction using MRI</h1>
  </nav>
  <div class="container">
    <a href="/predict1" class="button">Click here to drop a scan for detection</a>
    <div class="image-container">
      <img src="/static/background.jpg" alt="Background Image" width="640" height="480">
    </div>
  </div>
</body>
</html>
```

## alzpre.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Upload MRI Scan</title>
  <style>
    body {
      background-color: black;
      color: white;
      font-family: Arial, sans-serif;
    }
    .upload-container {
      margin-top: 50px;
      text-align: center;
    }
    .upload-container input[type="file"] {
      margin: 20px 0;
      padding: 10px;
    }
    .upload-container button {
      background-color: #4CAF50;
      border: none;
      color: white;
      padding: 15px 32px;
      text-align: center;
      text-decoration: none;
      display: inline-block;
      font-size: 16px;
      margin: 4px 2px;
      cursor: pointer;
    }
    .uploaded-image {
      margin-top: 20px;
      max-width: 300px;
      max-height: 300px;
    }
  </style>
</head>
<body>
  <div class="upload-container">
```

```
    <h2>Upload an MRI Scan for Detection</h2>
    <form id="uploadForm" action="/predict" method="post" enctype="multipart/form-data" onsubmit="return validateForm()">
      <input type="file" name="image" accept=".jpg">
      <button type="submit">Predict</button>
    </form>
    {% if image_path %}
      <h3>Uploaded Image:</h3>
      <img src="{{ url_for('static', filename='uploads/' + image_path.split('/')[-1]) }}" class="uploaded-image" alt="Uploaded MRI
Scan">
    {% endif %}
    {% if result %}
      <h3>Prediction: {{ result }}</h3>
    {% endif %}
  </div>

  <script>
    function validateForm() {
      var fileInput = document.querySelector('input[type="file"]');
      if (!fileInput.value) {
        alert('Please select an image file.');
        return false; // Prevent form submission
      }
      return true; // Allow form submission
    }
  </script>
</body>
</html>
```

## 10.2. GitHub & Project Demo Link

**GitHub**: https://github.com/RehanSuhail/Cognitive-Care-Early-Intervention-For-Alzheimer-s-Disease

**Project Demo : https://www.youtube.com/embed/mtIQoXgj6m0**