Project Final Documentation

Alzheimer's Disease Prediction

- Ashish Benny 21BPS1361
- Angad Singh 21BPS1420
- Sidharth Dinesh 21BAI1167

Index

1. **Introduction:**

1.1 Project Overview:

The project under review represents a significant advancement in the field of healthcare technology, focusing on the development of a deep learning-based AI model and an accompanying user-friendly website tailored for timely Alzheimer's stage prediction. Leveraging the power of deep learning, the project team successfully employed the state-of-the-art Xception model, widely recognized for its exceptional accuracy in image recognition tasks and its effectiveness in transfer learning scenarios. By combining cutting-edge technology with intuitive user interaction through the website interface, the project not only demonstrates technical prowess but also places a strong emphasis on user accessibility and experience.

1.2 Purpose:

The core objective of the project is to enable early and accurate prediction of Alzheimer's disease stages, addressing a crucial need in the medical field and allowing timely intervention by health professionals for a proper treatment plan based on the respective stage of Alzheimer’s detected. This initiative represents a commendable effort in the realm of medical AI, with the potential to significantly impact the lives of individuals affected by Alzheimer's disease.

2. LITERATURE SURVEY

2.1 Existing problem

Alzheimer's disease, a progressive neurodegenerative disorder, poses a significant challenge in healthcare due to limitations in its detection methods. Despite extensive research, several critical issues persist, hindering early and accurate diagnosis.

Firstly, the lack of a definitive diagnostic test for Alzheimer's remains a pressing concern. Current diagnostic methods primarily rely on clinical evaluations and cognitive assessments, which are inherently subjective and prone to interrater variability. This subjectivity results in delayed diagnosis, reducing the effectiveness of potential interventions.

Secondly, while neuroimaging techniques like MRI and PET scans offer valuable insights, they are costly, resource-intensive, and inaccessible to many patients. Furthermore, the interpretation of these images can be complex, requiring specialized expertise.

The heterogeneity of Alzheimer's disease further complicates detection efforts. The disease manifests differently among individuals, making it challenging to establish a one-size-fits-all diagnostic criterion. Subtypes of Alzheimer's, such as early-onset or atypical variants, require tailored detection methods, which adds complexity to the diagnosis process.

Addressing these challenges requires concerted efforts to develop more objective, affordable, and widely accessible diagnostic tools. Emerging technologies like machine learning and biomarker research hold promise in improving Alzheimer's disease detection. However, overcoming these hurdles is crucial to enable early intervention, optimize patient care, and advance research in the quest for a cure.

2.2 References

1. **Deep Learning Approach for Early Detection of Alzheimer's Disease**
2. **Diagnosis of Early Alzheimer's Disease: Clinical Practice in 2021**
3. **Novel Trends in Electrochemical Biosensors for Early Diagnosis**
4. **Systematic Literature Review of Automatic Alzheimer's Disease Detection**
5. **Single and Combined Neuroimaging Techniques for Alzheimer's Disease**
6. **Machine Learning Models for Alzheimer's Disease Detection Using OASIS Data:**
7. **A Comprehensive Review and Current Methods for Classifying Alzheimer's Disease Using Feature Extraction and Machine Learning Techniques:**

2.3 Problem Statement Definition

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. After Prediction of the Stage of disease, the WebApp also recommends the suitable prognosis for the patient based on the Current Disease Progression and Health Circumstances

3. Ideation and Proposed Solution

3.1 Empathy Map Canvas:

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

It is a useful tool to helps 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.

Detection of Alzheimer's disease using deep learning is important because early detection can lead to earlier interventions, such as lifestyle changes and medications, which may help to slow down the progression of the disease. Our project aims at classifying the progression of the disease based on severity. Additionally, early detection can help families and caregivers to prepare for the future. It also allows patients to make their own decisions about their care, instead of having those decisions made for them by others.

Reference:

https://app.mural.co/t/angad20029321/m/angad20029321/1697085904956/ecf6d7ac68bf92a9a3c456b4bee0d9b400163551?sender=uea8530e24312095208ca6299

Empathy Map Canvas Image:

Template

Empathy map canvas

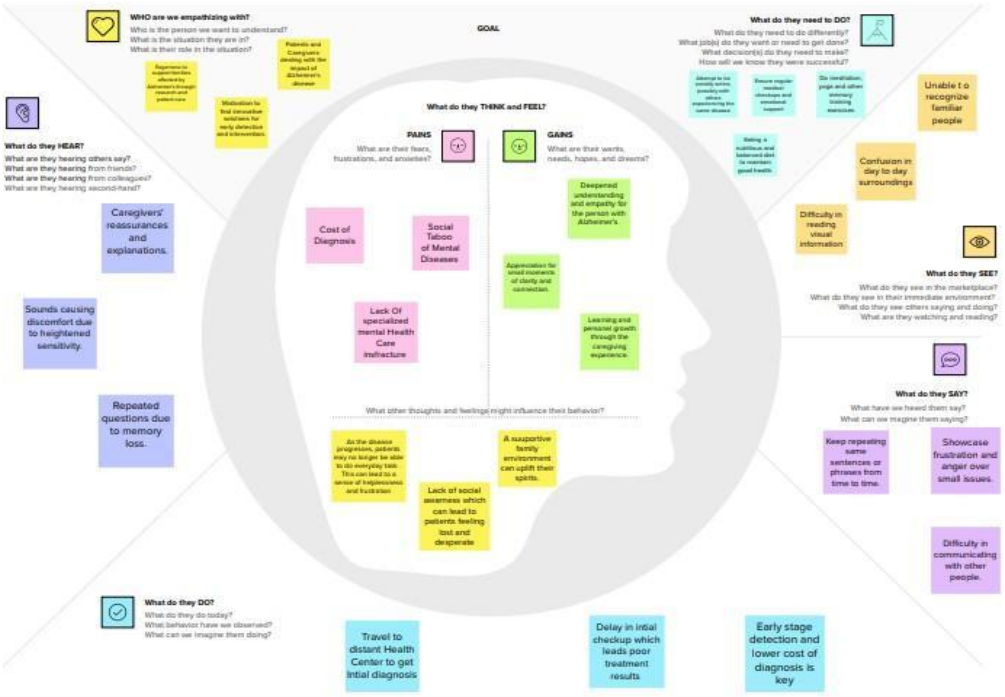
Use this framework to empathize with a customer, user, or any person who is affected by a team's work. Document and discuss your observations and note your assumptions to gain more empathy for the people you serve.

Originally created by Dave Gray at

Share template feedback

Alzheimer Disease Prediction

Detection of Alzheimer's disease using deep learning is important because early detection can lead to earlier interventions, such as lifestyle changes and medications, which may help to slow down the progression of the disease. Our project aims at classifying the progression of the disease based on severity. Additionally, early detection can help families and caregivers to prepare for the future.



Need some inspiration?

See a finished version of this template to kickstart your work.

Open example

3.2 Ideation and Brainstorming

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

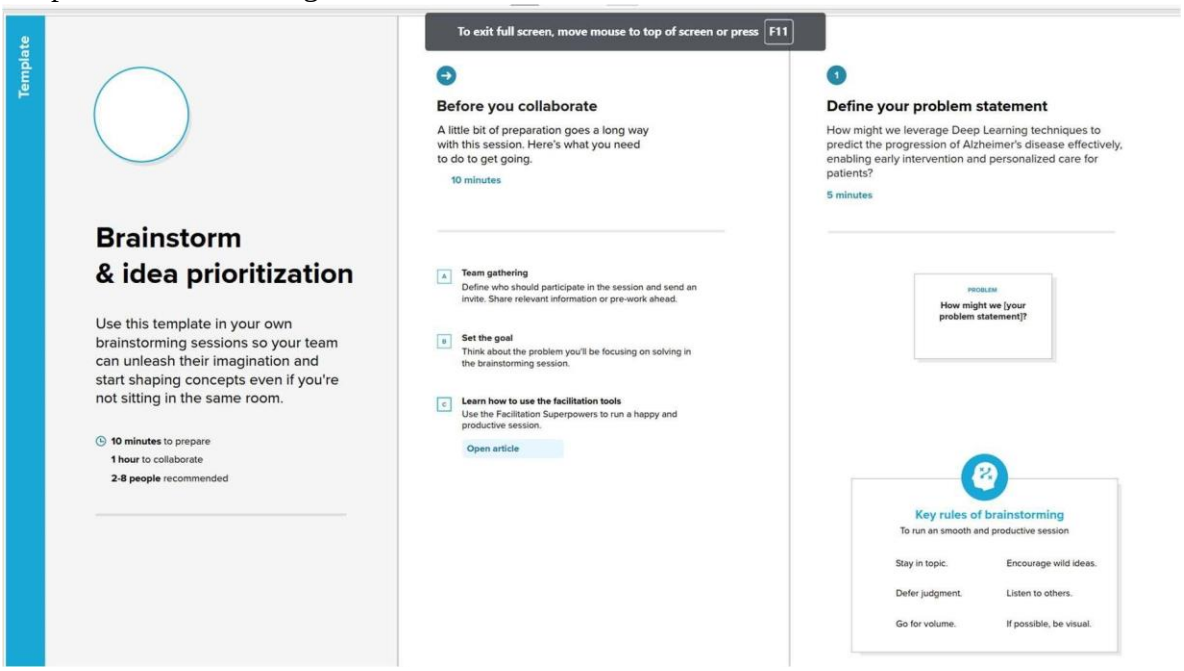
Mural Link:

h
ps://app.mural.co/t/angad20029321/m/angad20029321/1697171919867/14c94eab01f0de5741
4 6bc835eb53c746fc4880c?sender=uff1521e2631a09f255c59105

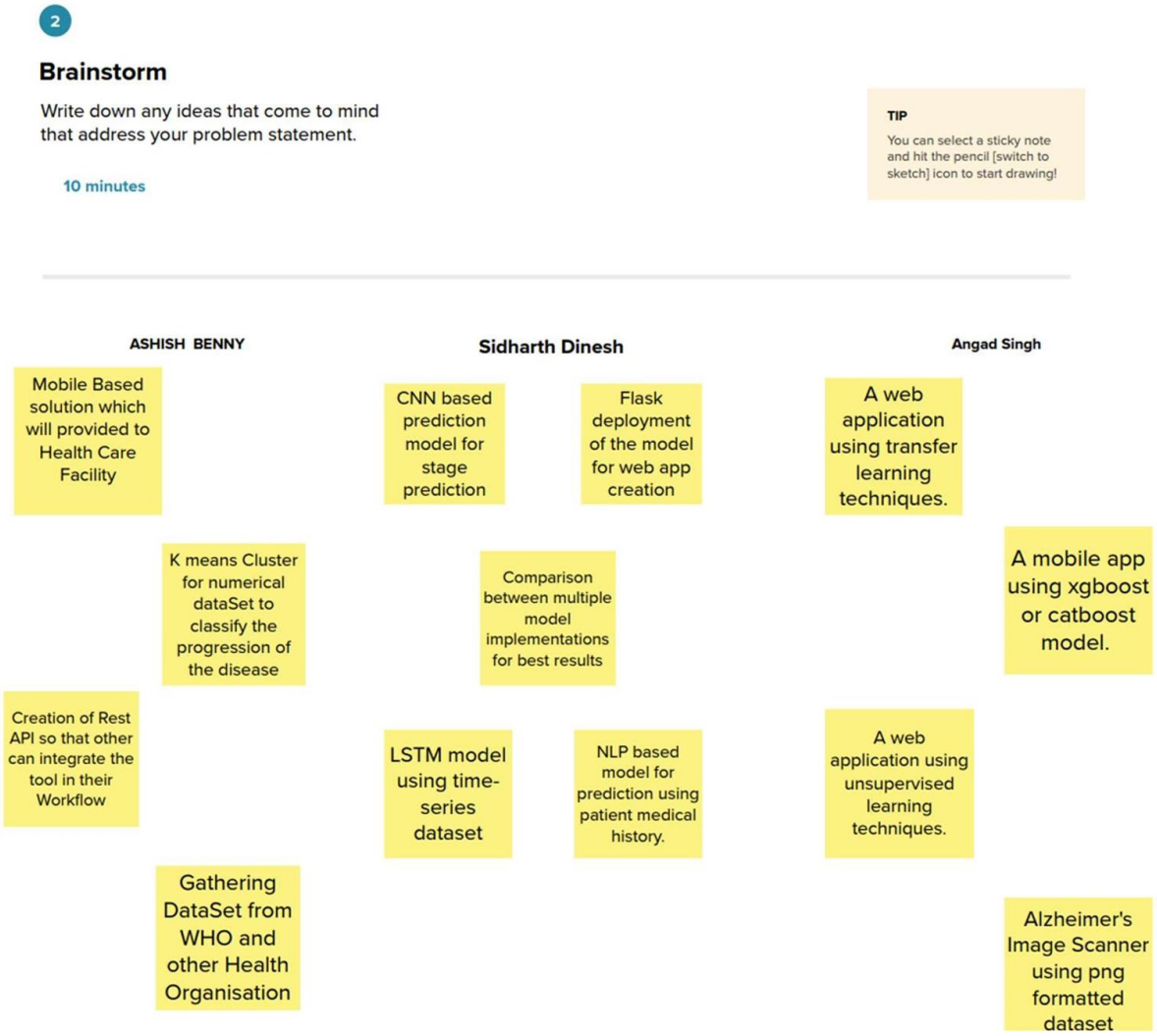
Problem Statement For Brainstorming:

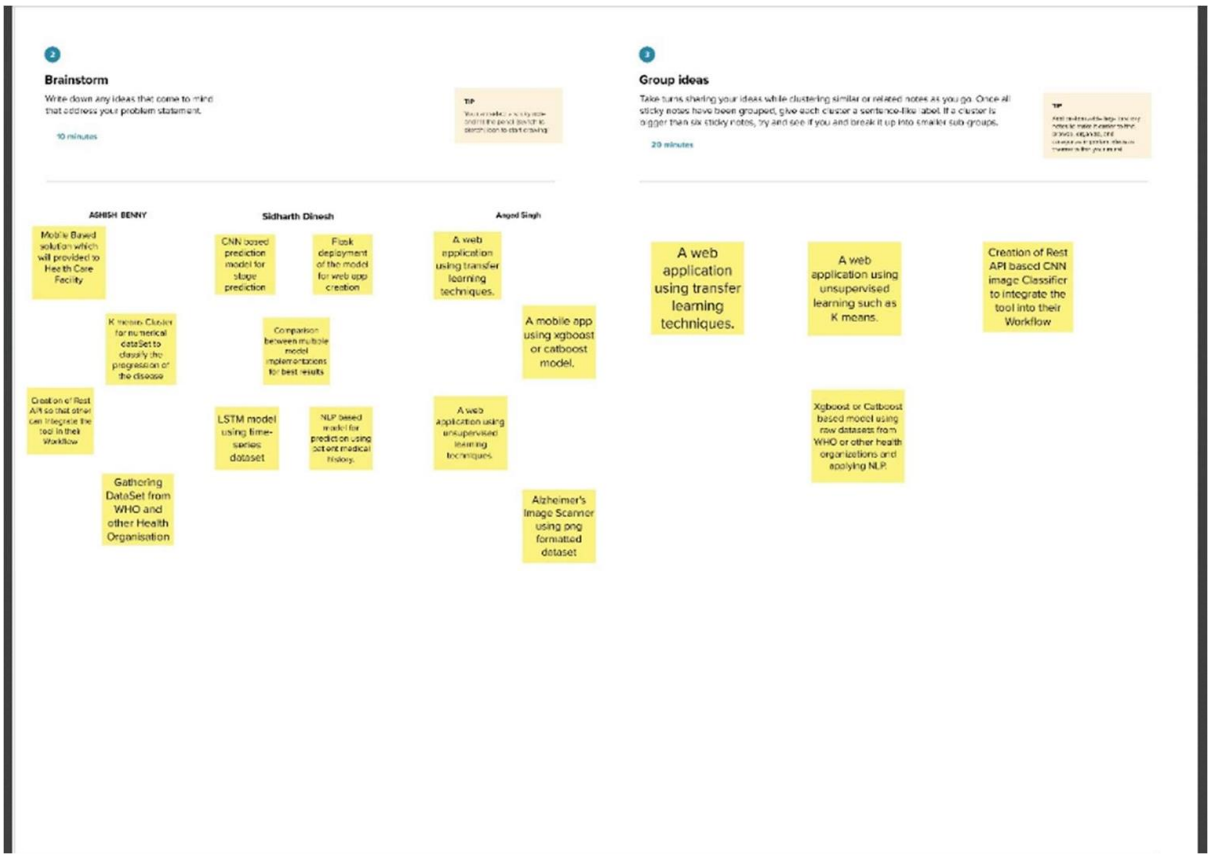
To discuss and collect ideas regarding how to collect the data, how to preprocess it, how to create the model and integrate with a website. Then prioritize the ideas of importance.

Step-1: Team Gathering, Collaboration and Select



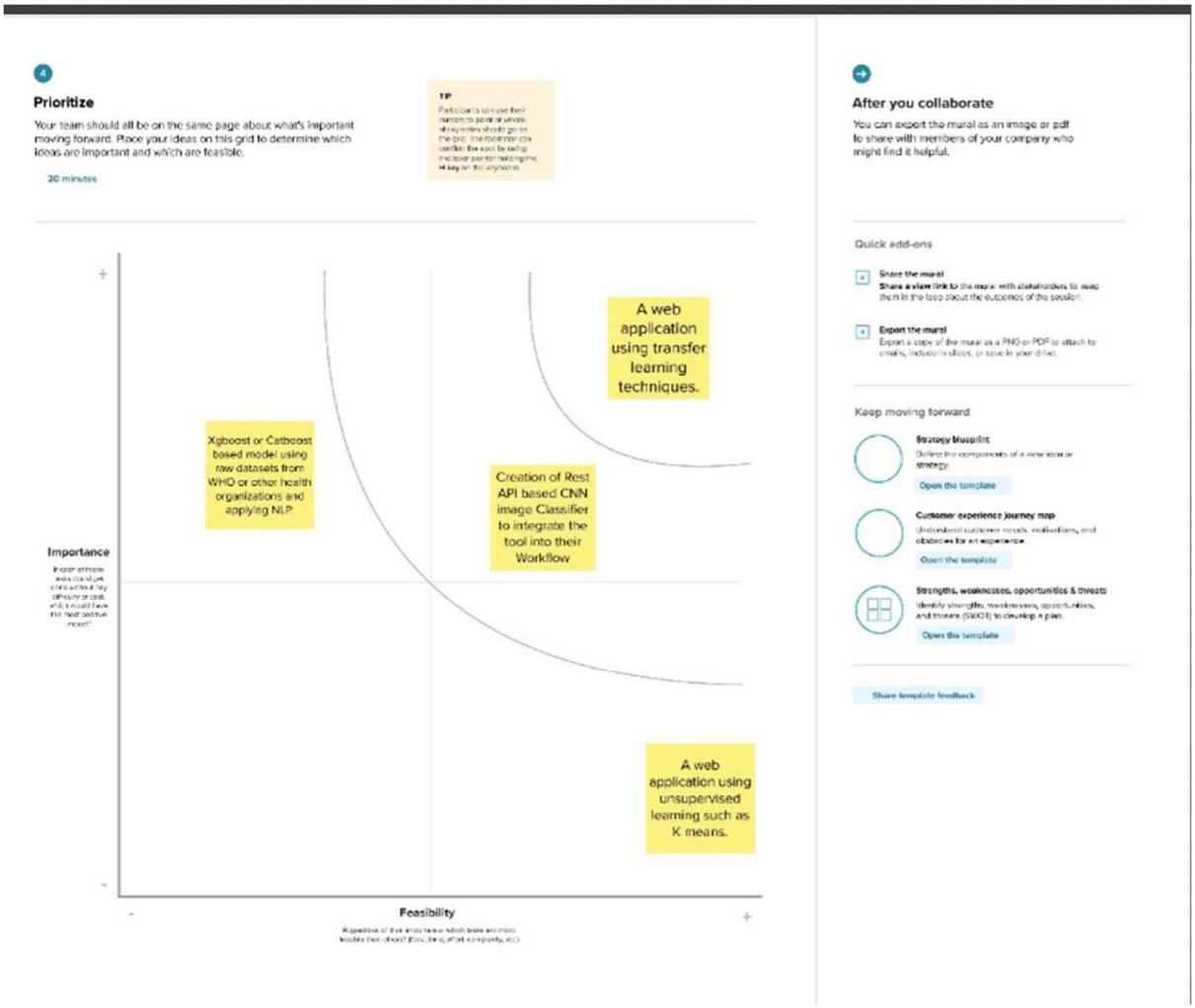
Step-2: Brainstorm, Ideation, Listing and Grouping





Step-3: Idea Prioritization

df


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
TP
A web application using transfer learning techniques.
Xgboost or Catboost based model using raw datasets from WHO or other health organizations and applying NLP.
Creation of Rest API based CNN image Classifier to integrate the tool into their Workflow
A web application using unsupervised learning such as K means.
Importance
Feasibility
After you collaborate
You can export the mural as an image or pdf to share with members of your company who might find it helpful.
Quick add-ons
Share the mural
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.
Export the mural
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.
Keep moving forward
Strategy blueprint
Open the template
Customer experience journey map
Open the template
Strengths, weaknesses, opportunities & threats
Open the template
Share template feedback

df

Proposed Solution:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The current challenge lies in the early detection and prediction of Alzheimer's disease, which significantly impacts the effectiveness of treatment and care. Existing diagnostic methods often fail to identify the disease at its initial stages, leading to delayed interventions and suboptimal patient outcomes. This delay also contributes to increased healthcare costs and places additional emotional and financial burdens on patients and their families. The goal of the project is to predict the current Alzheimer's stage of the patient using an efficient model for early intervention and diagnosis.

df

2.	Idea / Solution description	The idea is to use the four pre-trained transfer learning models VGG16, ResNet50, Inception and Xception on an image dataset which shows various scanned images of the brain to predict the state of Alzheimer's progression. Then a comparative study is performed for the four models to determine which among them provides most accurate results. The parameters used in the CNN networks within the transfer learning models have to be fine-tuned for best results. After the best model prediction is chosen, a website is created to display the appropriate treatment plan according to the prediction made.

df

3.	Novelty / Uniqueness	Use of Convolutional Neural Networks (CNNs) for Alzheimer's detection ensures precision and efficiency, reducing the burden on medical staff. It promotes early intervention, protects patients' health, and contributes to early diagnosis and treatment leading to better Patient treatment and making the project a holistic and innovative approach to healthcare.
4.	Social Impact / Customer Satisfaction	Early detection of a disease goes beyond the medical setting. It affects people's lives in a positive way, by detecting the disease early and enabling treatment to begin sooner. This can lead to a better quality of life for patients and their families. In addition, it has a positive impact on the healthcare industry, as it can reduce costs and improve outcomes. Finally, the project has an economic impact, as it can lead to increased productivity and reduced healthcare costs.

df

5	Business Model (Revenue Model)	We intend to collaborate with healthcare providers and research institutions, offering them subscription-based access to our predictive model platform. Additionally, we plan to provide data analysis services for medical research and partner with pharmaceutical companies for the development of targeted treatments, creating a sustainable revenue stream. Furthermore, we will offer customizable solutions tailored to the specific needs of healthcare organizations, enabling us to establish long-term partnerships and ensure the continuous growth of our business.
6.	Scalability of the Solution	Our solution is designed to accommodate a vast amount of data from various sources, ensuring seamless integration with different healthcare systems. With a scalable infrastructure in place, we aim to deploy the model across multiple healthcare facilities and research institutions,

df

		continually updating it based on the latest research findings and data insights. Additionally, we will invest in a robust data management system and implement stringent security measures to maintain data integrity and protect patient confidentiality, thereby fostering trust and enabling the widespread adoption of our solution.

4. REQUIREMENT ANALYSIS

4.1 **Functional Requirements**

- **Data Preprocessing**: The system should be capable of preprocessing and cleaning large-scale neuroimaging data, including MRI and PET scans, to extract relevant features for Alzheimer's disease prediction.
- **Model Architecture Implementation:** Implement the Xception deep learning model for image classification to accurately predict Alzheimer's disease based on neuroimaging data.
- **Training and Evaluation:** The system must be able to train the Xception model using the preprocessed data and evaluate its performance using appropriate metrics such as accuracy, precision, recall, and F1 score.
- **User Interface:** Develop a user-friendly interface to allow researchers and medical professionals to easily upload, process, and predict Alzheimer's disease from new neuroimaging data.

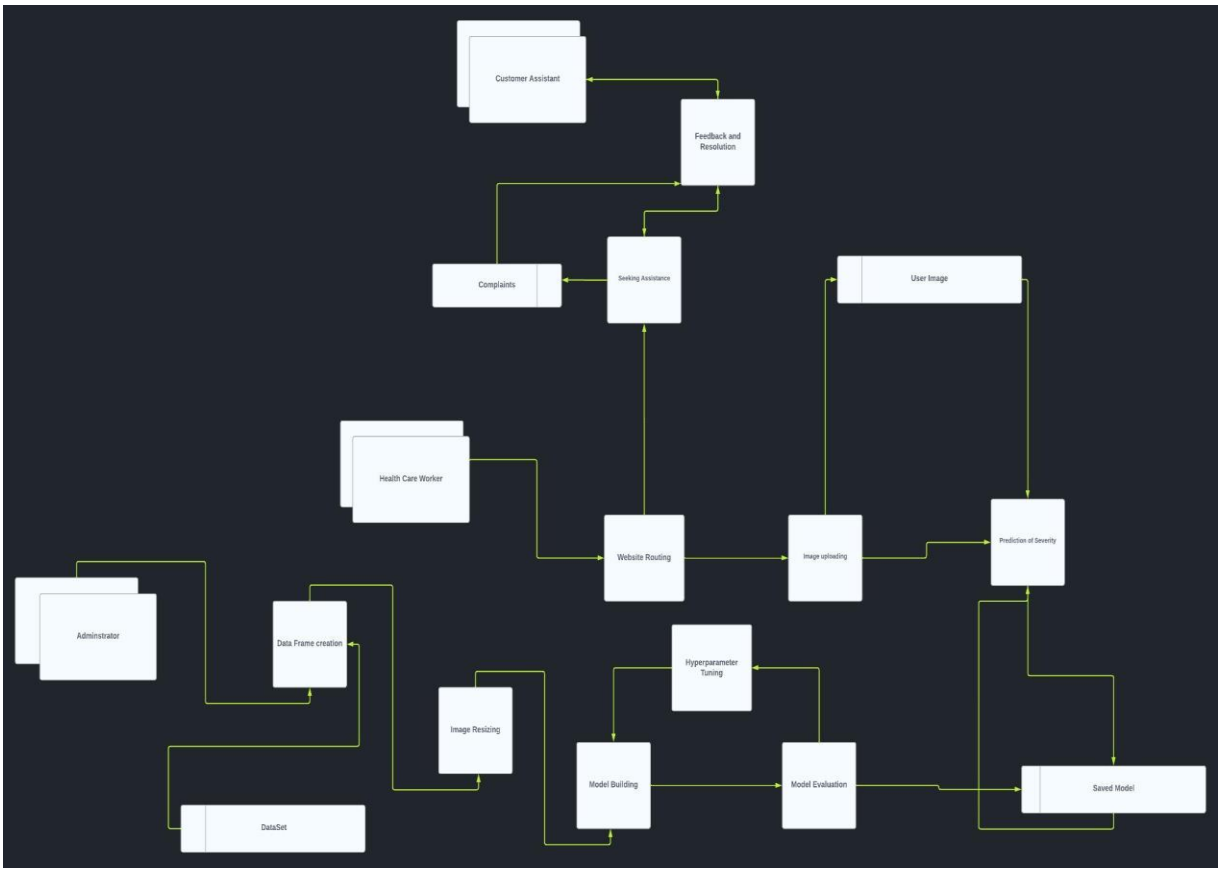
4.2 **Non-Functional Requirements**

- Performance: The model should achieve a high accuracy rate (above 90%) and be capable of making predictions within a reasonable timeframe to ensure its practicality for clinical applications.
- **Scalability:** The system should be scalable to handle a large volume of neuroimaging data and be able to accommodate future expansions and updates in the dataset.
- Security: Ensure data security and privacy compliance by implementing appropriate measures such as encryption, access controls, and user authentication to protect sensitive patient information.
- **Robustness**: The system should be robust enough to handle variations in input data quality and size, ensuring consistent and reliable predictions even in the presence of noise or artifacts in the neuroimaging data.
- **Usability**: Develop an intuitive and user-friendly interface that requires minimal training for users to upload data, interpret results, and understand the model's predictions

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

df


Customer Assistant
Feedback and Resolution
Complaints
Seeking Assistance
User Image
Health Care Worker
Website Routing
Image uploading
Prediction of Severity
Adminstrator
Data Frame creation
Hyperparameter Tuning
Image Resizing
Model Building
Model Evaluation
Saved Model
DataSet

df

User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Health Care Professional (Web User)	Website	USN-1	As a user,I need a website where I can view the different information available about the service	I can access the website with simple Components	High	Sprint-1
		USN-2	As a user, I can check alzheimer Progression of my disease on the WebApp	I can upload the image in the website	High	Sprint-1
		USN-3	As a user, I can get the recommended course of Action for the disease	I can get a prognosis for the stage of alzheimer disease	Medium	Sprint-2
	WebSite Design	USN-4	As a user, I can view about Us page about the service	I can view the about section of the website	Low	Sprint-3
		USN-5	As a User, I can view images and video about the alzheimer disease to know more about the disease	I can view image and video on the APP	Low	Sprint-3

Customer Care Executive	Expert assistance	USN-6	As a user, I get enough information and detailing on the Alzheimer prediction tool to help in providing assistance and clearing doubts of patients and their family seeking help	Successful provision of assistance and clearing of patient doubts	High	Sprint-3

	Seamless Communication	USN-7	As a user, I need an efficient contact system comprising of phone calls as well as emails, avoiding excessive load for smooth and consistent communication and ensuring prompt responses	Successful call connection and email forwarding without server overload	High	Sprint-3
	Outreach to customers	USN-8	As a user, I need to be able to proactively be able to contact customers and give them updates regarding developments or results	Successful transmission of recent updates on a real-time basis	Medium	Sprint-3
Administrator (Backend)	Model Creation	USN-9	As a user, I have to download the dataset and perform data preprocessing on it.	Successfully downloading the dataset and preprocessing its features.	High	Sprint 2
		USN-10	As a user, I have to create a model, and train it on the preprocessed dataset.	Successfully creating and training the model.	High	Sprint 2
		USN-11	As a user, I have to save the successfully trained model for future use.	Successfully saving the trained model.	High	Sprint 2
	Flask Framework	USN-12	As a user, I have to create a flask file for routing of web pages.	Successfully create a flask framework for routing	High	Sprint 2

5.2 Solution Architecture

This solution uses pre-trained CNN models like VGG16, ResNet50, Inception-v3 and Xception to predict the current Alzheimer's stage of the patient accurately. The images in the dataset have to be resized to a 224 * 224 dimension for using in VGG16 and ResNet50 models, whereas they have to be resized to a 299*299 dimension for using in the Inception-v3 and Xception models. The four models have different layers and structures. The VGG16 model has 13 convolution layers, 5 max pooling layers and 3 fully connected dense layers. ResNet50 has 48 convolutional layers, 1 max pool layer and 1 average pool layer. Inception-v3 has 48 layers in total. Xception has 71 layers in total. In addition we add a Flattening layer to each model and set the activation function of output layer as 'softmax' for multi classification scenario. The difference in number of layers indicates a possibility of getting different predictions for each model based on accuracy. A comparative study between the models would help in determining the best model for Alzheimer's stage prediction which helps in early intervention and diagnosis. After model building and determining which of the four models is the best, save the model and then use the prediction of the model as a form of input to a website created via Flask deployment, which provides the resultant strategy for treatment as a result. The multiple layers of each CNN model and the comparisons help in training the model well. Our solution leverages deep learning strategy which uses pre-trained CNN models to address the Alzheimer disease prediction problem effectively.

● Data Gathering: We are using an image Dataset that we obtained from Kaggle ,which contains the different stages of the Alzheimer's disease progression.

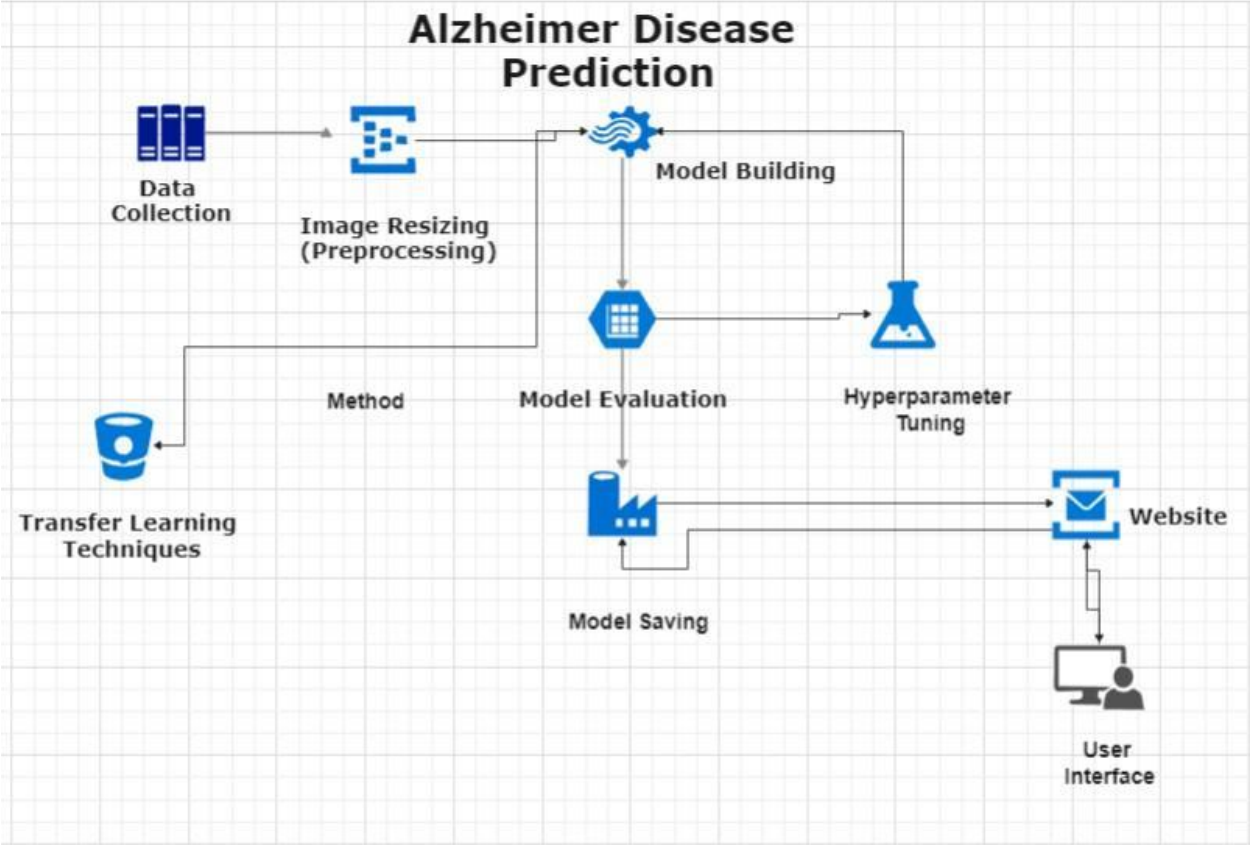
● Image Preprocessing: We are going to resize the input images based on the individual model. For example for VGG16 and ResNet50 we are going to resize to (224,224,3), whereas for Inception-v3 and Xception we are going to resize to (299,299,3) . The next step will be to flatten the image so as to input the pooling features into the input layer of the Neural Network.

● Model Building: Deep learning strategy of Transfer Learning is used, which employs pre-trained CNN models for training and prediction. The four types of Transfer Learning used in this project are: VGG16, ResNet50, Inception-v3 and Xception.

● Alzheimer's Disease Prediction: After testing all the models for their accuracy and errors by doing hyperparameter tuning of the models, we will do a Comparative Study of each of the models to select the best model for our Use Case and the Dataset.

● Real Time Analysis: Finally, we will download the model and Create a WebApp using Flask and create a User Interface for the Users to use our Model and check whether they are having Alzheimer's by uploading their EMR image in the Website.

Solution Architecture Diagram:



6. Project Planning and Scheduling

6.1 Technical Stack:

Technical Architecture:

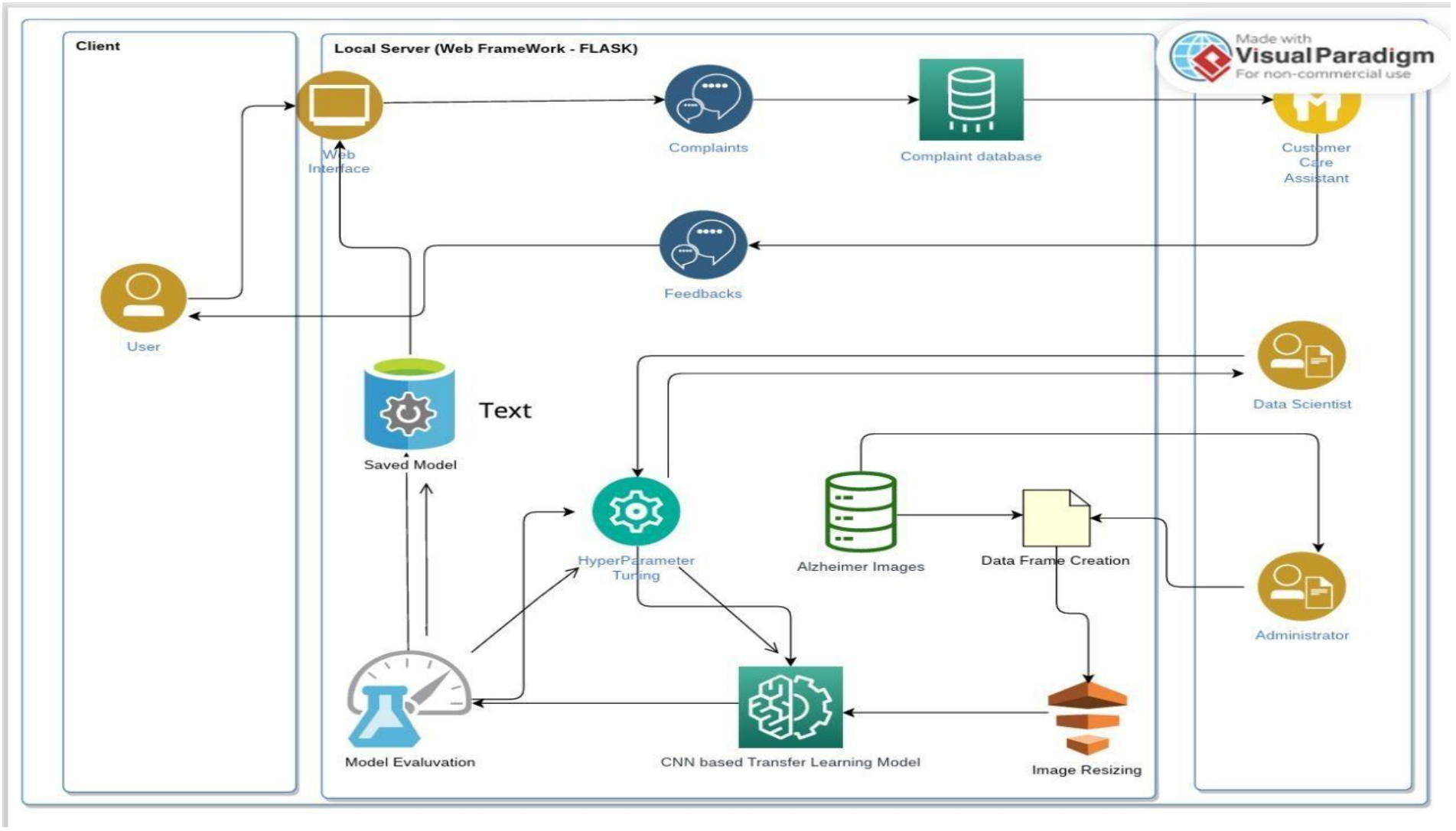


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Users can upload input images to the website and use the predict option in the website to predict the current stage of Alzheimers. They can also request assistance or file complaints which are stored in a database and further directed to the customer care assistant who provides them with the respective feedback.	HTML, CSS, JavaScript.
2.	Application Logic-1	Downloading the dataset, dataframe creation, preprocessing.	Python (pandas, numpy), Kaggle
3.	Application Logic-2	Model building, Model evaluation and Hyperparameter Tuning.	Python (Tensorflow)
4.	Application Logic-3	Model Saving.	Python (Pickle)
5.	Database	Image Database. (jpeg)	Kaggle
6.	File Storage	All files saved locally in VSCode.	VSCode, GitHub
7.	External API-1	Kaggle API used to import Data From using API tokens.	Kaggle

8.	External API-2	Flask has an API that allows developers to easily build RESTful web services such as Routes and Request Objects. Access information about the incoming HTTP request, such as the URL, request method, and request data.	Flask
9.	Machine Learning Model	The transfer learning model uses deep learning	Transfer Learning models - VGG16,
		techniques which reads image datasets and trains them based on pre - defined CNN layers for accurate prediction of result.	ResNet50, Inception and Xception.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Flask is a lightweight Python web framework known for its simplicity and ease of use a great choice for building small to medium-sized web applications.	Flask FrameWork Used
2.	Scalable Architecture	3 tier Client-Server Architecture using a local host.	Visual Paradigm, IBM tools
3.	Availability	The application will be available on the local environment.	Local Host Used

4.	Performance	Used by a single user at a time.	Command Line

6.2 Sprint Planning & Estimation

Project Planning Phase

Schedule, and Estimation

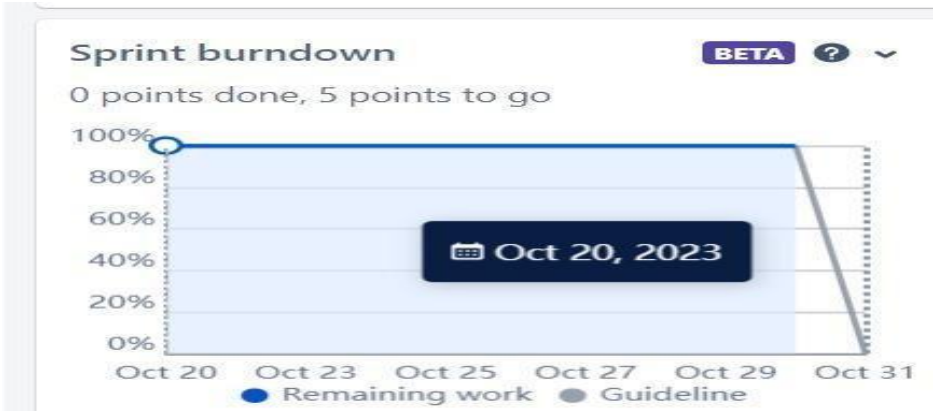
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team members
Sprint-1	Website	USN-1	As a user,I need a website where I can view the different information available about the service	3	High	Ashish Benny
Sprint-1	Website	USN-2	As a user, I can check alzheimer Progression of my disease on the WebApp	2	High	Ashish Benny
Sprint-2	Website	USN-3	As a user, I can get the recommended course of Action for the disease	2	Medium	Ashish Benny
Sprint-3	WebSite Design	USN-4	As a user, I can view about Us page about the service	1	Low	Ashish Benny
Sprint-3	WebSite Design	USN-5	As a User, I can view images and video about the alzheimer disease to know more about the disease	2	Low	Ashish Benny

Sprint-3	Expert assistance	USN-6	As a user, I get enough information and detailing on the Alzheimer prediction tool to help in providing assistance and clearing doubts of patients and their family seeking help	3	High	Sidharth Dinesh
Sprint-3	Seamless Communication	USN-7	As a user, I need an efficient contact system comprising of phone calls as well as emails, avoiding excessive load for smooth and consistent communication and ensuring prompt responses	2	High	Sidharth Dinesh
Sprint-3	Outreach to customers	USN-8	As a user, I need to be able to proactively be able to contact customers and give them updates regarding developments or results	1	Medium	Sidharth Dinesh
Sprint 2	Model Creation	USN-9	As a user, I have to download the dataset and perform data preprocessing on it.	2	High	Angad Singh
Sprint 2	Model Creation	USN-10	As a user, I have to create a model, and train it on the preprocessed dataset.	3	High	Angad Singh
Sprint 2	Model Creation	USN-11	As a user, I have to save the successfully	1	High	Angad Singh
			trained model for future use.			
Sprint 2	Flask Framework	USN-12	As a user, I have to create a flask file for routing of web pages.	2	High	Angad Singh

Project Tracker, Velocity & Burndown Chart:

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	5	2 Days	30 Oct 2023	31 Oct 2023	5	31 Oct 2023
Sprint-2	10	3 Days	01 Nov 2023	03 Nov 2023	15	03 Nov 2023
Sprint-3	9	3 Days	04 Nov 2022	06 Nov 2023	24	06 Nov 2023

Velocity:

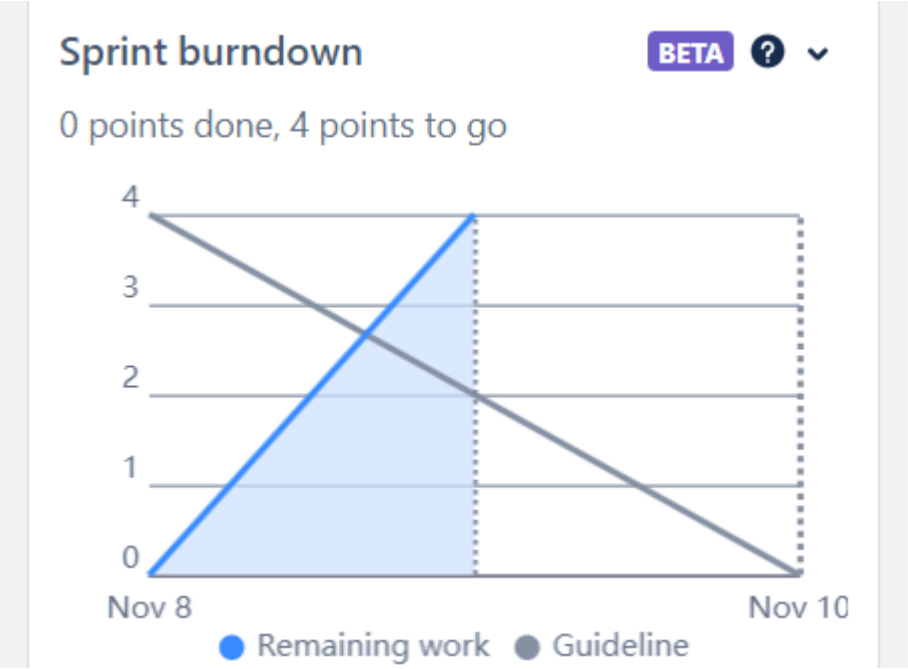


Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

V1(**Sprint 1**) = 2 / 5 = 0.40
V2 (**Sprint 2**) = 3 / 10 = 0.30
V3 (**Sprint 3**)= 3 / 9 = 0.33

Total Sprint Duration= 8 days
Total Velocity = 24 points

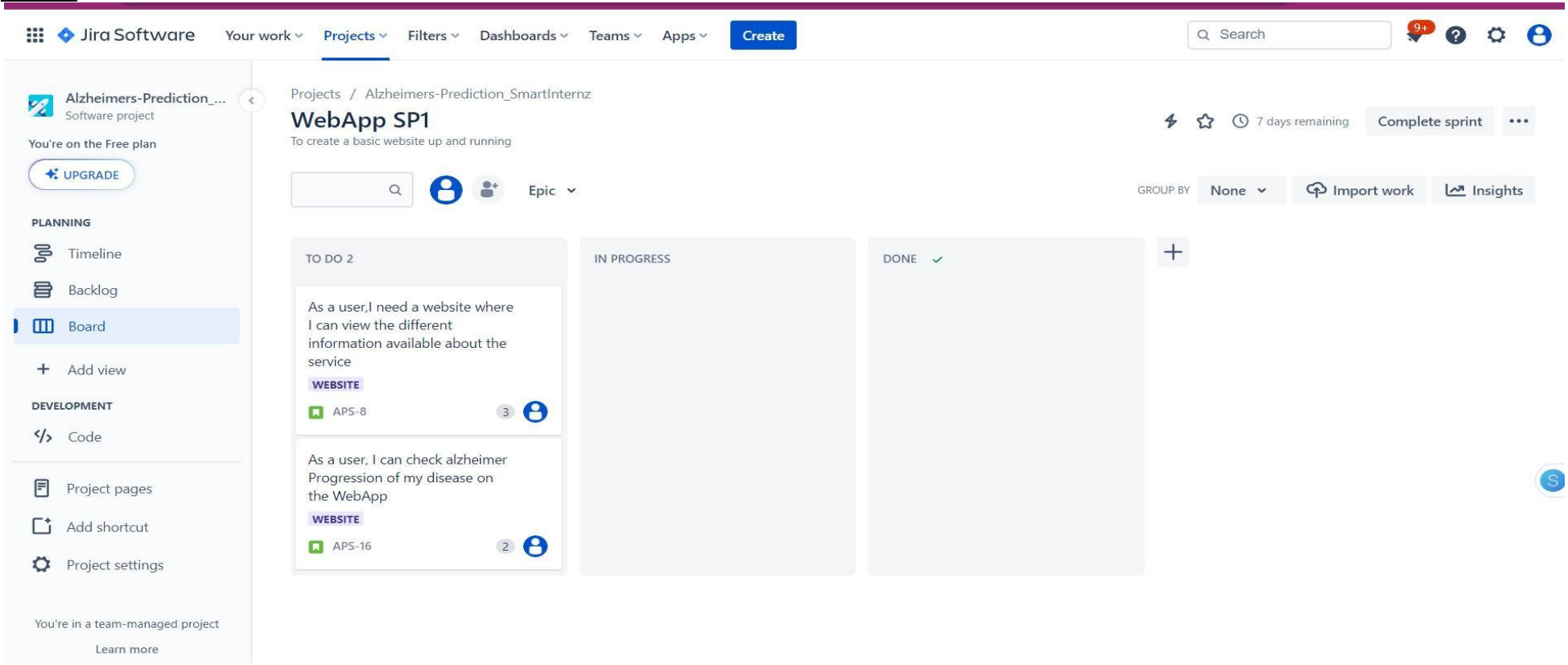
Burndown Chart:



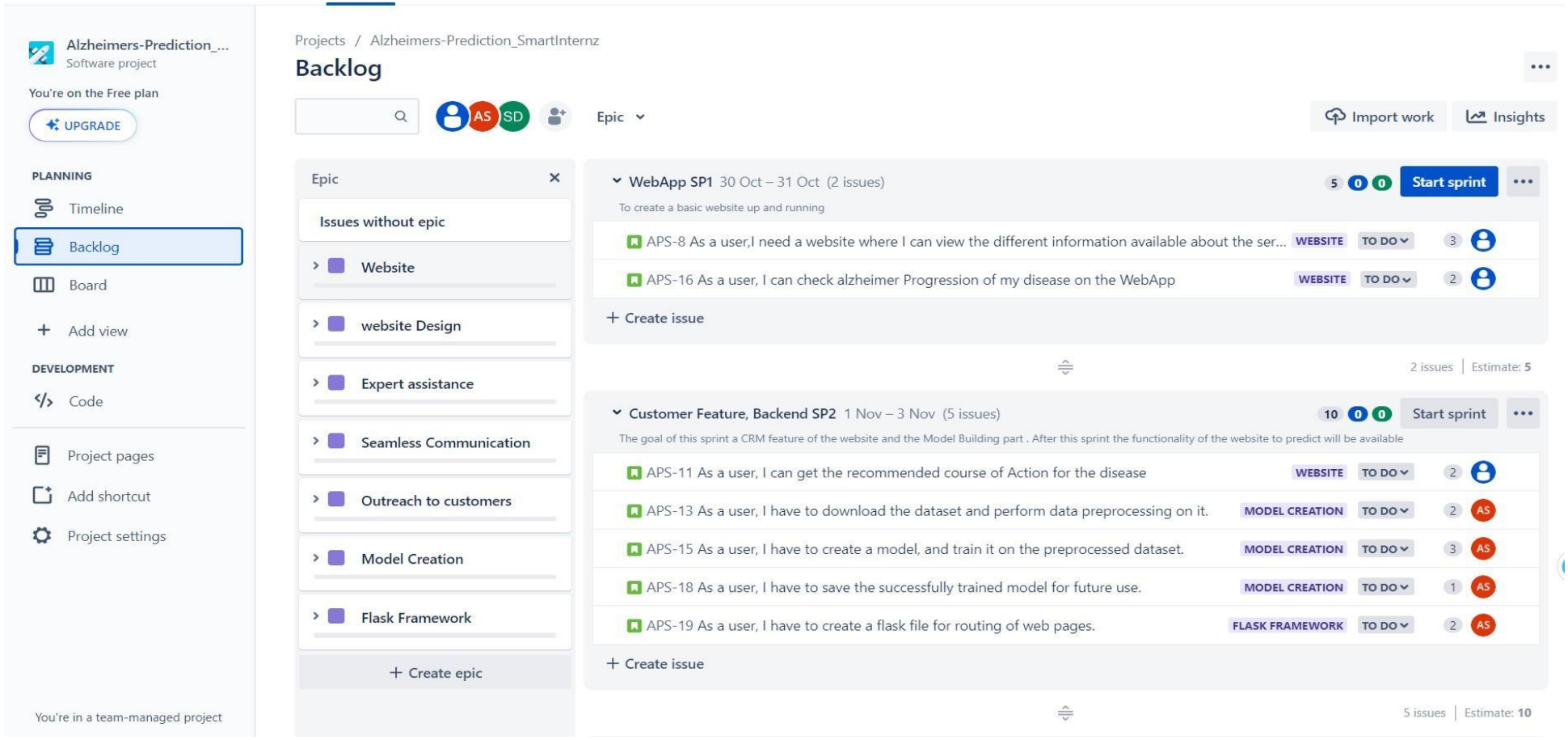
Average Velocity (AV) = **Sprint Duration / Velocity** = (2+3+3) / (5+10+9) = 8 / 24 = **0.33**

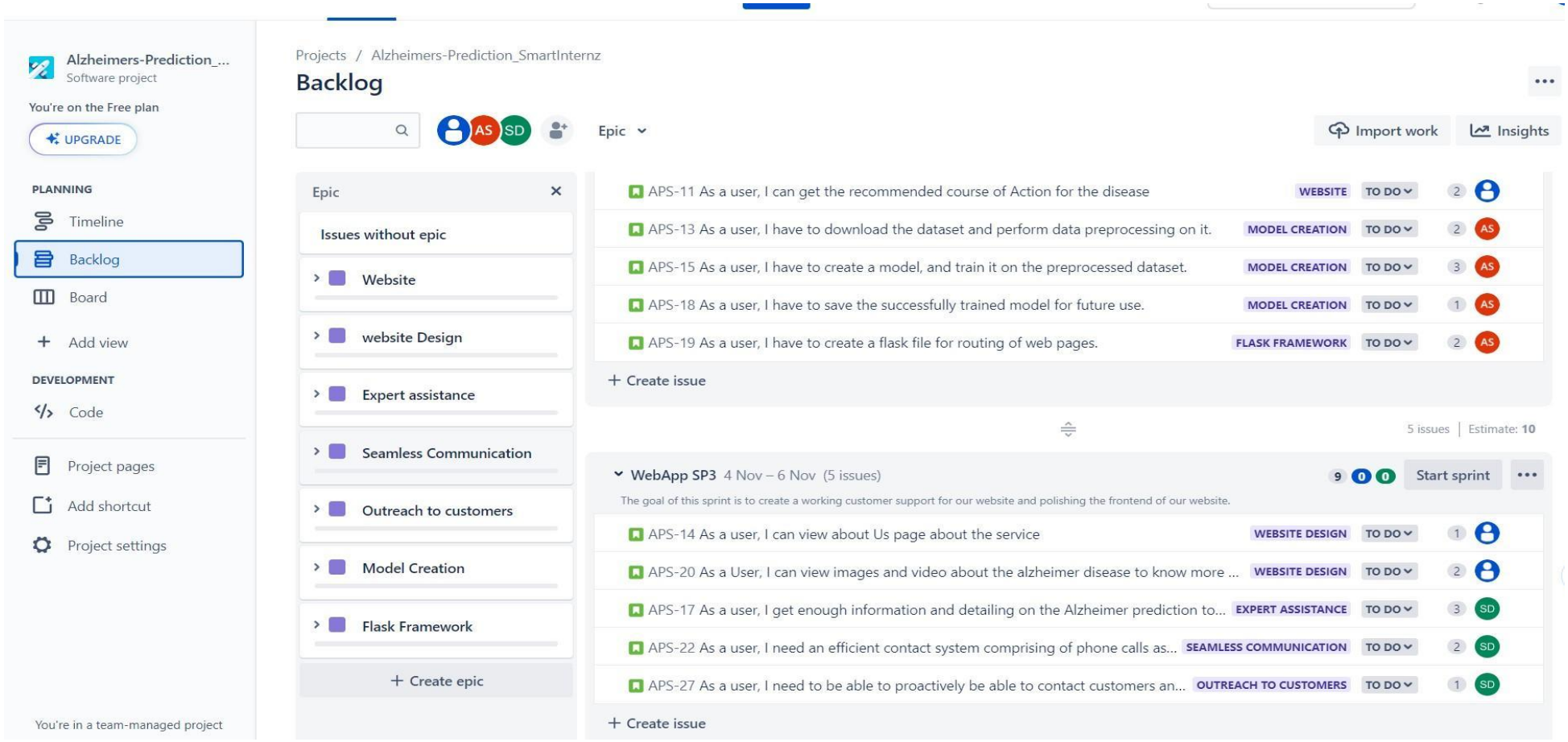
6.3 Sprint Delivery Schedule:

Board:

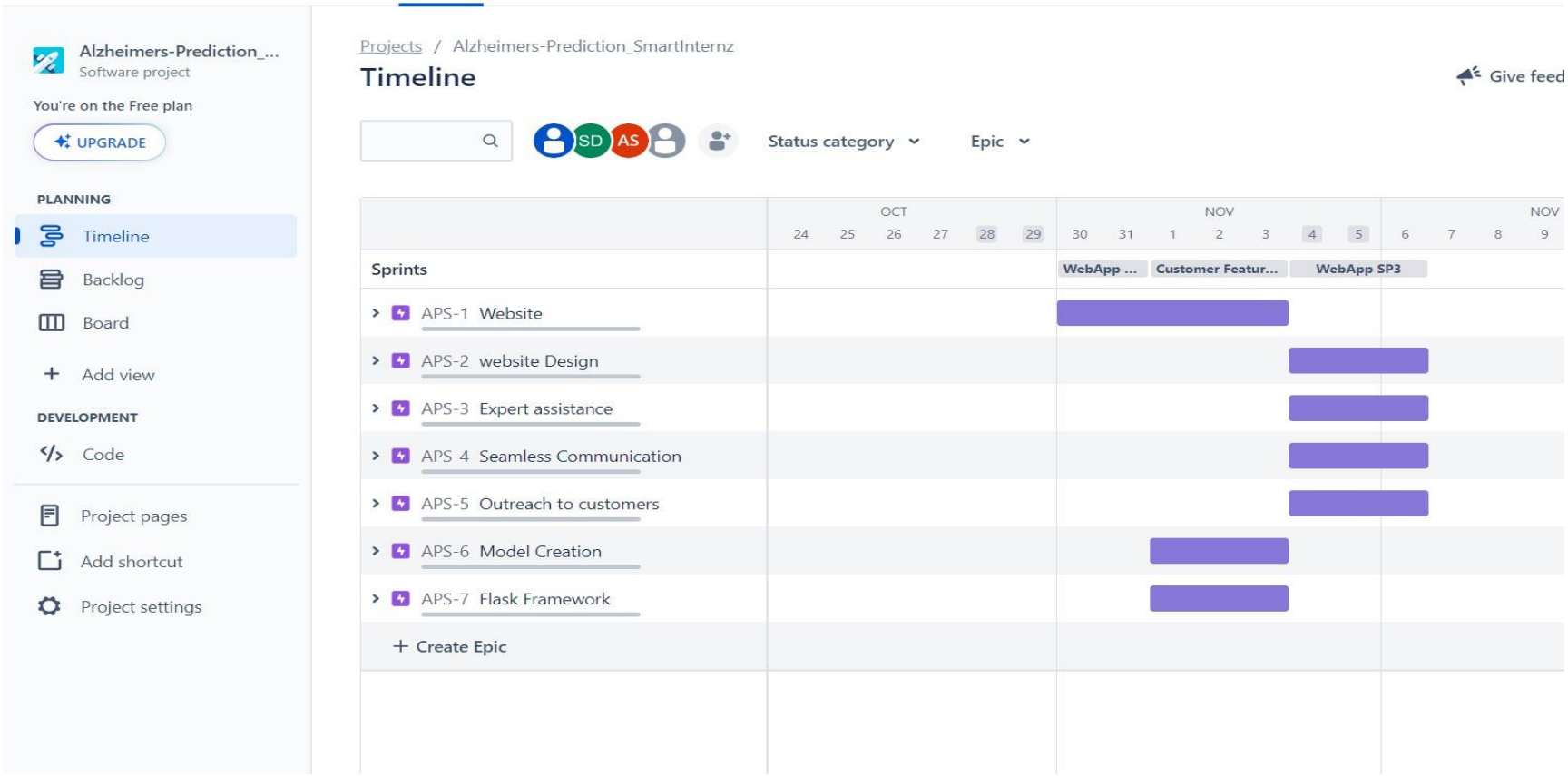


Backlog


Alzheimers-Prediction_...
Software project
You're on the Free plan
UPGRADE
PLANNING
Timeline
Backlog
Board
Add view
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Projects / Alzheimers-Prediction_SmartInternz
Backlog
AS
SD
Epic
Import work
Insights
Epic
Issues without epic
Website
website Design
Expert assistance
Seamless Communication
Outreach to customers
Model Creation
Flask Framework
Create epic
WebApp SP1 30 Oct – 31 Oct (2 issues)
5 0 0
Start sprint
To create a basic website up and running
APS-8 As a user,I need a website where I can view the different information available about the ser... WEBSITE TO DO 3
APS-16 As a user, I can check alzheimer Progression of my disease on the WebApp WEBSITE TO DO 2
Create issue
2 issues | Estimate: 5
Customer Feature, Backend SP2 1 Nov – 3 Nov (5 issues)
10 0 0
Start sprint
The goal of this sprint a CRM feature of the website and the Model Building part . After this sprint the functionality of the website to predict will be available
APS-11 As a user, I can get the recommended course of Action for the disease WEBSITE TO DO 2
APS-13 As a user, I have to download the dataset and perform data preprocessing on it. MODEL CREATION TO DO 2 AS
APS-15 As a user, I have to create a model, and train it on the preprocessed dataset. MODEL CREATION TO DO 3 AS
APS-18 As a user, I have to save the successfully trained model for future use. MODEL CREATION TO DO 1 AS
APS-19 As a user, I have to create a flask file for routing of web pages. FLASK FRAMEWORK TO DO 2 AS
Create issue
5 issues | Estimate: 10


Alzheimers-Prediction_...
Software project
You're on the Free plan
UPGRADE
PLANNING
Timeline
Backlog
Board
Add view
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Projects / Alzheimers-Prediction_SmartInternz
Backlog
AS
SD
Epic
Import work
Insights
Epic
Issues without epic
Website
website Design
Expert assistance
Seamless Communication
Outreach to customers
Model Creation
Flask Framework
Create epic
APS-11 As a user, I can get the recommended course of Action for the disease WEBSITE TO DO 2
APS-13 As a user, I have to download the dataset and perform data preprocessing on it. MODEL CREATION TO DO 2 AS
APS-15 As a user, I have to create a model, and train it on the preprocessed dataset. MODEL CREATION TO DO 3 AS
APS-18 As a user, I have to save the successfully trained model for future use. MODEL CREATION TO DO 1 AS
APS-19 As a user, I have to create a flask file for routing of web pages. FLASK FRAMEWORK TO DO 2 AS
Create issue
5 issues | Estimate: 10
WebApp SP3 4 Nov – 6 Nov (5 issues)
9 0 0
Start sprint
The goal of this sprint is to create a working customer support for our website and polishing the frontend of our website.
APS-14 As a user, I can view about Us page about the service WEBSITE DESIGN TO DO 1
APS-20 As a User, I can view images and video about the alzheimer disease to know more ... WEBSITE DESIGN TO DO 2
APS-17 As a user, I get enough information and detailing on the Alzheimer prediction to... EXPERT ASSISTANCE TO DO 3 SD
APS-22 As a user, I need an efficient contact system comprising of phone calls as... SEAMLESS COMMUNICATION TO DO 2 SD
APS-27 As a user, I need to be able to proactively be able to contact customers an... OUTREACH TO CUSTOMERS TO DO 1 SD
Create issue

TimeLine


Alzheimers-Prediction_...
Software project
You're on the Free plan
UPGRADE
PLANNING
Timeline
Backlog
Board
Add view
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
Projects / Alzheimers-Prediction_SmartInternz
Timeline
Give feed
SD
AS
Status category
Epic
OCT
NOV
NOV
24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9
Sprints
WebApp ...
Customer Featur...
WebApp SP3
APS-1 Website
APS-2 website Design
APS-3 Expert assistance
APS-4 Seamless Communication
APS-5 Outreach to customers
APS-6 Model Creation
APS-7 Flask Framework
Create Epic

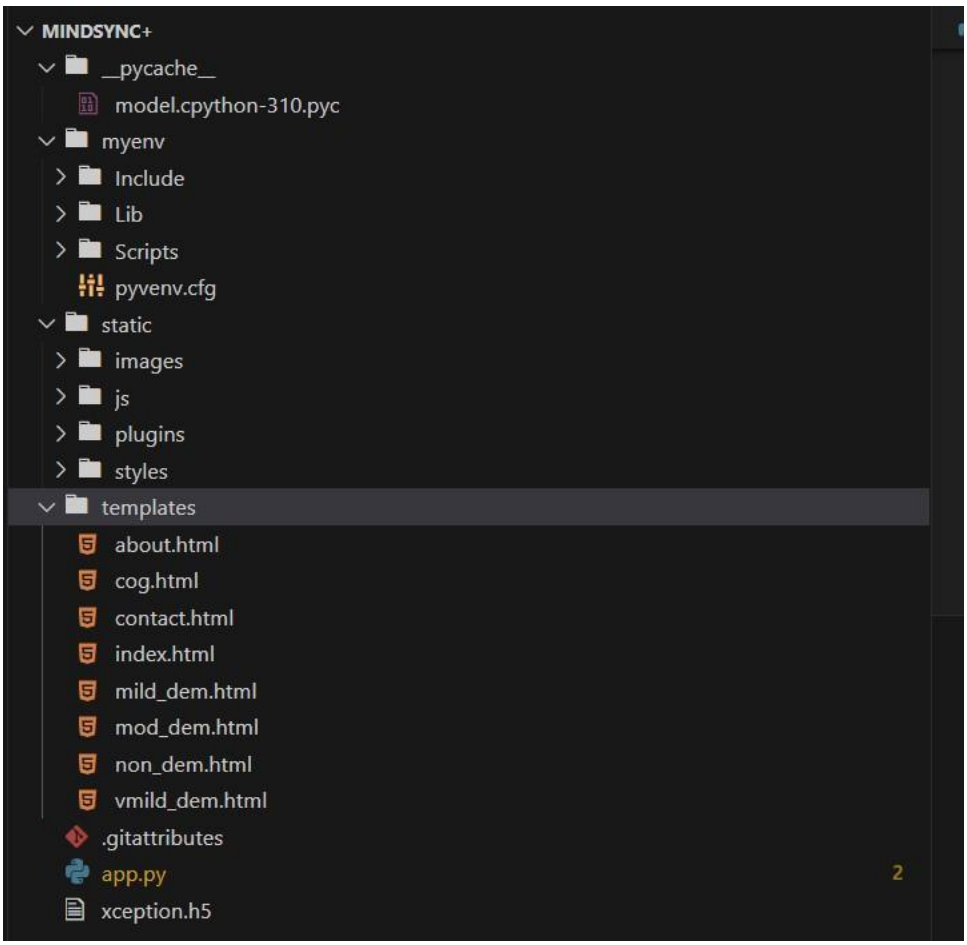
7. Coding and Solutioning

Project Flow:

- The user uploads an image to the website.
- The uploaded 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 of the current stage of Alzheimers.
- 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.

1. **Data Collection.**
    - A Kaggle image dataset consisting of scanned images of the brain is used.
    - Divide it into train and test paths.
2. **Image Pre-processing.**
    - Import the required libraries.
    - Perform image augmentation.
    - Set input image size as (150,150) for Xception model.
    - Set the suitable batch size, zoom, bright_range, horizontal_flip, fill_mode, data_format and so on.
    - Perform oversampling using SMOTE for balancing the dataset.
3. **Model Building**
    - Use a customized Xception model.
    - Add layers like dropout, global_average_pooling_2D, Flatten, BatchNormalization and Dense layers to customize the pre-built CNN model of Xception.
    - Among them there will be 4 Dense layers with activation function as ‘relu’ with 512, 256, 128 and 64 filters respectively.
    - The final output layer will be another Dense layer with 4 possible outcomes and a softmax activation function for multi output.
    - Configure the training process, setting the optimizer as ‘Adam’, loss as ‘categorical_crossentropy’ and metrics as ‘Accuracy’, ‘Precision’, ‘Recall’ and ‘AUC’.
    - Train the model using 20 epochs.
    - Save the model.
    - Test and evaluate the model.
4. **Application Building o Create an HTML file**
    - Create CSS files for Styling the Div for the website and for animation
    - oUsing JavaScript to provide functionality to the Websites Component such as Uploading Images
    - Build Python Flask Code for creating and accessing the prediction website to obtain accurate diagnosis of the Alzheimers disease upon request.
    - Run the application and test its efficiency.

Project Structure:



- The xception.h5 contains the model trained in Google Colab python notebook which has a customized **Xception** trained 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.
- There are Four main HTML file in this project which are cog.html, contact.html, about.html, index.html.
- Remaining HTML files are for the result based on the prediction.
- Static contains the style folder which contains the **CSS** file for the HTML page. Each page has static and dynamic CSS.
- In Flask, **myenv** typically refers to the virtual environment or Python environment where you install and manage the dependencies for your Flask web application. It's not a built-in concept in Flask itself but rather a common practice when developing Flask applications (or any Python applications) to isolate your project-specific dependencies from the system-wide Python environment.
- In the app,py file, we are importing modules like Flask, render_template, request, load_model, image and Image from libraries like flask, keras, PIL and tensorflow. We also import the numpy library.
- The render_template() is used for routing the python file to the html pages like index.html, about.html, cog.html and contact.html.
- This is followed by the python code for the customized Xception model for prediction of stage and intensity of Alzheimer's disease.

7.1 Transfer Learning Model Training:

Milestone 1: Data Collection

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.
In this project, we are using Kaggle to obtain the dataset of scanned images of the brain. The data consists of MRI images.
The data has four classes of images both in training as well as a testing set

Activity 1: Import the required libraries

Import important libraries like TensorFlow, keras, NumPy, pandas, seaborn, matplotlib, os, PIL, random, imblearn, sklearn and so on. Also import the necessary modules.

```
!pip install tensorflow_addons

import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf
import matplotlib.pyplot as plt

import os
from distutils.dir_util import copy_tree, remove_tree

from PIL import Image
from random import randint

from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.metrics import matthews_corrcoef as MCC
from sklearn.metrics import balanced_accuracy_score as BAS
from sklearn.metrics import classification_report, confusion_matrix

import tensorflow_addons as tfa
from keras.utils import plot_model
from tensorflow.keras import Sequential, Input
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import Conv2D, Flatten
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.preprocessing.image import ImageDataGenerator as IDG
from tensorflow.keras.layers import SeparableConv2D, BatchNormalization, GlobalAveragePooling2D
```

Activity 2: Download the dataset

Collect images of brain MRI then split into train and test set. Then within these folders, organize them 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 ‘MildDemented’, ‘ModerateDemented’, ‘NonDemented’ and ‘VeryMildDemented’ 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

We are going to build our training model on Google colab. We upload kaggle.json and acquire the respective input image files from Kaggle using their API token link. The zip folder uploaded is unzipped to get folders for different sub-categories of image dataset we need. The following steps are followed to achieve the same:

- 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 kaggle.json file from the respective directory.
- Wait for the upload to complete. You should see the file appear in the "Files" section.
- Now use the API token link created for the respective dataset to upload the zip file of the dataset. ● To unzip the file, you can use the following command:

```
!pip install -q kaggle

!mkdir ~/.kaggle

!cp kaggle.json ~/.kaggle

!kaggle datasets download -d tourist55/alzheimers-dataset-4-class-of-images
!unzip alzheimers-dataset-4-class-of-images
```

Activity 3: Create training and testing dataset

To build a Deep Learning 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.

```
base_dir = "/content/Alzheimer_s Dataset/"
root_dir = "./"
test_dir = base_dir + "test/"
train_dir = base_dir + "train/"
work_dir = root_dir + "dataset/"

if os.path.exists(work_dir):
    remove_tree(work_dir)

os.mkdir(work_dir)
copy_tree(train_dir, work_dir)
copy_tree(test_dir, work_dir)
print("Working Directory Contents:", os.listdir(work_dir))
```

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, zooming etc. The preprocessing steps performed include image augmentation, dataset balancing using SMOTE and so on.

Activity 1: Importing the libraries

Import the necessary libraries as shown in the image

```
import tensorflow_addons as tfa
from keras.utils import plot_model
from tensorflow.keras import Sequential, Input
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import Conv2D, Flatten
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.preprocessing.image import ImageDataGenerator as IDG
from tensorflow.keras.layers import SeparableConv2D, BatchNormalization, GlobalAveragePooling2D
```

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 provides support for working with large, multi-dimensional arrays and matrices of numerical data, as well as a wide range of mathematical functions to operate on these arrays
- **tensorflow_addons** as **tfa**: tensorflow_addons is an additional library for TensorFlow that provides extra functionality, operations, and layers not available in the core TensorFlow package. It includes various addons for machine learning and deep learning tasks.
- **keras.utils**: This module provides utility functions, including plot_model, which allows you to generate a plot of the neural network model. This can be helpful for visualizing the architecture of the model.
- **Sequential:** It is a linear stack of layers in Keras. It allows you to create models layer-by-layer in a step-by-step fashion. Each layer has weights that correspond to the layer that follows it in the model.
- **Conv2D**: Conv2D is a 2D convolutional layer in Keras. It performs 2D convolutional operations on the input data, typically used for image processing tasks.
- **ReduceLROnPlateau:** It is a callback in Keras. It monitors a specified metric and reduces the learning rate when the metric has stopped improving. This is helpful for improving training when the model's performance plateaus.
- **SeparableConv2D**: SeparableConv2D is a depthwise separable 2D convolutional layer in Keras. It performs a spatial convolution on each channel of the input data independently and then combines the results.

- **BatchNormalization**: BatchNormalization is a layer in Keras that normalizes and scales the activations of a neural network, reducing internal covariate shift and improving training stability.

- **GlobalAveragePooling2D:** GlobalAveragePooling2D is a layer in Keras that performs global average pooling on the spatial dimensions of the input data. It reduces each feature map to a single value, effectively creating a global average of the features.

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. This additional set of modified images provides additional batches of input for training which improves the capability of the model. The image size is set to (150,150) before being augmented. 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.
- fill_mode='constant': 'fill_mode' is a parameter used to specify the strategy for filling in newly created pixels that might appear after a transformation. When it is 'constant', any newly created pixels outside the boundaries of the original image are filled by a constant value specified by the 'cval' parameter which is 0 by default.
- data_format = 'channels_last': 'data_format' is a parameter that determines the ordering of the dimensions in the input data. It specifies how the data is structured in multi-dimensional arrays.When it is set as 'chennels_last', the order for 2D case is (samples, height, width, channels) and for 3D case is (samples, depth, height, width, channels).
- Also set the batch_size as 6500 and shuffle as False and assign the correct directories for train and test.
- 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_data_gen 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.

```
WORK_DIR = './dataset/'

CLASSES = [ 'NonDemented',
            'VeryMildDemented',
            'MildDemented',
            'ModerateDemented']

IMG_SIZE = 150
IMAGE_SIZE = [150, 150]
DIM = (IMG_SIZE, IMG_SIZE)

# Performing Image Augmentation to have more data samples

ZOOM = [.99, 1.01]
BRIGHT_RANGE = [0.8, 1.2]
HORZ_FLIP = True
FILL_MODE = "constant"
DATA_FORMAT = "channels_last"

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)

# Retrieving the data from the ImageDataGenerator iterator

train_data, train_labels = train_data_gen.next()

# Getting to know the dimensions of our dataset

print(train_data.shape, train_labels.shape)
```

Activity 3: Handling imbalance data (SMOTE):

```
# Retrieving the data from the ImageDataGenerator iterator

train_data, train_labels = train_data_gen.next()

# Getting to know the dimensions of our dataset

print(train_data.shape, train_labels.shape)
```

```
Found 6400 images belonging to 4 classes.
(6400, 150, 150, 3) (6400, 4)
```

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

Checking the data before oversampling. As we can see, there are 5121 total images in the train data and 1279 images in the test data.

The SMOTE() is used to apply the SMOTE oversampling technique to balance the dataset. The random_state value provided is 42.

The reshape(-1, IMG_SIZE*IMG_SIZE*3) is used to reshape the 2D matrix shape of the input image of (150,150,3) dimension to a 1D array of length 150*150*3.

The sm.fit_resample technique is used to apply the SMOTE technique on the reshape train_dataset. After the sampling is applied the images are again reshaped back to a 4D array of shape (batch_size, IMG_size, IMG_size,3).

```
# Performing over-sampling of the data since the classes are imbalanced

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)
```

After oversampling the data which had a shape of (6400, 150, 150, 3) has now doubled to (12,800, 150, 150, 3). The data has been doubled due to oversampling. As we can observe, from 5121 to 10,242.

Activity 4: Splitting into train test split

```
# Splitting the data into train, test, and validation sets

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 and assigning them to train_data, train_labels and test_data, test_lables. We also do the same for the validation test and assign to train_data, train_labels and val_data, val_

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. We perform some customization to the pre-built Xception model to make it more accurate and for better training purpose.

Deep understanding on the Xception model is required.

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

Here, we have considered images of dimension (150,150,3).

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).

```
IMG_SIZE = 150
IMAGE_SIZE = [150, 150]
DIM = (IMG_SIZE, IMG_SIZE)
```

Different Transfer Learning models are used in our project such as VGG16, ResNet50, InceptionV3 and Xception, and the best model (Xception) is selected as it gives best metrics out of all of them. The image input size of Xception model is 150, 150.

```
# Change the model to Xception
from tensorflow.keras.applications.xception import Xception

xception_model = Xception(input_shape=(150, 150, 3), include_top=False, weights="imagenet")

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

Activity 2: Creating Sequential layers

```
custom_xception_model = Sequential([
    xception_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="xception_cnn_model")
```

Now we customize the Xception CNN model to make the training process better and more efficient.

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.

We first input the pre-built Xception model. Then we add a Dropout layer with parameter as 0.5. Dropout rate is set to 0.5 as a form of regularization technique to prevent overfitting.

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

Then we add a Flattening layer where a multi-dimensional array can be converted into a one-dimensional array.

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

BatchNormalization : Then a Batch Normalization 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.

This is followed by 4 alternating sets of Dense layer, Batch Normalization layer and Dropout layer. These 4 Dense layers have a 'ReLu' activation function (Rectified Linear Unit). It has 512, 256, 128 and 64 filters respectively.

Finally add another Dense layer with 4 filters and an activation function 'softmax' as the output layer.

ReLu activation function has been taken in the first four 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 custom_xception_model with inputs as xception.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. In this case, we are using the 'adam' optimizer.

Metrics are used to evaluate the performance of your model. They provide insights into how well the model is performing but are not used during the training process. In our code, we have defined a set of custom metrics using TensorFlow's Keras metrics, including Binary Accuracy, Precision, Recall, and AUC. These metrics will be used to measure various aspects of the model's performance.

```
METRICS = [
    tf.keras.metrics.BinaryAccuracy(name='accuracy'),
    tf.keras.metrics.Precision(name='precision'),
    tf.keras.metrics.Recall(name='recall'),
    tf.keras.metrics.AUC(name='auc')
]
```

Activity 4: Train the model

Now, we will train our model with our image dataset. The model is trained for 20 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 and training accuracy increases.

The **fit** function is used to train a deep learning neural network. The custom_xception_model.compile() takes the optimizer, loss and metrics as its parameters whereas the custom_xception_model.fit() takes the training and validation datasets, labels and the number of epochs as its parameters.

```
custom_xception_model.compile(optimizer='adam',
                              loss=tf.losses.CategoricalCrossentropy(),
                              metrics=METRICS)

custom_xception_model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
xception (Functional)        (None, 5, 5, 2048)        20861480

dropout (Dropout)            (None, 5, 5, 2048)        0

global_average_pooling2d (   (None, 2048)              0
GlobalAveragePooling2D)

flatten (Flatten)            (None, 2048)              0

batch_normalization_8 (Bat   (None, 2048)              8192
chNormalization)

dense (Dense)                (None, 512)               1049088

batch_normalization_9 (Bat   (None, 512)               2048
chNormalization)

dropout_1 (Dropout)          (None, 512)               0

dense_1 (Dense)              (None, 256)               131328

batch_normalization_10 (Ba   (None, 256)               1024
tchNormalization)

dropout_2 (Dropout)          (None, 256)               0

dense_2 (Dense)              (None, 128)               32896

batch_normalization_11 (Ba   (None, 128)               512
tchNormalization)

dropout_3 (Dropout)          (None, 128)               0

dense_3 (Dense)              (None, 64)                8256

dropout_4 (Dropout)          (None, 64)                0

batch_normalization_12 (Ba   (None, 64)                256
tchNormalization)

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)
_________________________________________________________________
```

```
[ ]
# Fit the training data to the model and validate it using the validation data
EPOCHS = 20

history = custom_xception_model.fit(train_data, train_labels, validation_data=(val_data, val_labels), epochs=EPOCHS)
```

Arguments:

- Epochs: an integer and number of epochs we want to train our model for. (20 epochs is used here)
- 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_xception_model.fit(train_data, train_labels, validation_data=(val_data, val_labels), epochs=EPOCHS)
```

```
Epoch 1/20
256/256 [==============================] - 41s 91ms/step - loss: 1.4895 - accuracy: 0.7276 - precision: 0.4157 - recall: 0.2211 - auc: 0.6274 - val_loss: 1.0269 - val_accuracy: 0.7915
Epoch 2/20
256/256 [==============================] - 21s 82ms/step - loss: 1.0417 - accuracy: 0.7921 - precision: 0.6591 - recall: 0.3491 - auc: 0.8001 - val_loss: 0.7444 - val_accuracy: 0.8373
Epoch 3/20
256/256 [==============================] - 21s 84ms/step - loss: 0.8685 - accuracy: 0.8172 - precision: 0.7404 - recall: 0.4141 - auc: 0.8600 - val_loss: 0.6784 - val_accuracy: 0.8473
Epoch 4/20
256/256 [==============================] - 21s 82ms/step - loss: 0.7836 - accuracy: 0.8324 - precision: 0.7766 - recall: 0.4625 - auc: 0.8864 - val_loss: 0.6488 - val_accuracy: 0.8519
Epoch 5/20
256/256 [==============================] - 21s 82ms/step - loss: 0.7592 - accuracy: 0.8382 - precision: 0.7813 - recall: 0.4901 - auc: 0.8950 - val_loss: 0.6200 - val_accuracy: 0.8588
Epoch 6/20
256/256 [==============================] - 21s 82ms/step - loss: 0.7262 - accuracy: 0.8442 - precision: 0.7874 - recall: 0.5160 - auc: 0.9042 - val_loss: 0.5861 - val_accuracy: 0.8639
Epoch 7/20
256/256 [==============================] - 20s 78ms/step - loss: 0.6785 - accuracy: 0.8528 - precision: 0.7913 - recall: 0.5587 - auc: 0.9157 - val_loss: 0.5777 - val_accuracy: 0.8662
Epoch 8/20
256/256 [==============================] - 21s 82ms/step - loss: 0.6705 - accuracy: 0.8562 - precision: 0.7864 - recall: 0.5830 - auc: 0.9192 - val_loss: 0.5566 - val_accuracy: 0.8677
Epoch 9/20
256/256 [==============================] - 20s 77ms/step - loss: 0.6453 - accuracy: 0.8623 - precision: 0.7892 - recall: 0.6129 - auc: 0.9249 - val_loss: 0.5346 - val_accuracy: 0.8806
Epoch 10/20
256/256 [==============================] - 20s 77ms/step - loss: 0.6413 - accuracy: 0.8634 - precision: 0.7810 - recall: 0.6304 - auc: 0.9265 - val_loss: 0.5327 - val_accuracy: 0.8823
Epoch 11/20
256/256 [==============================] - 20s 78ms/step - loss: 0.6170 - accuracy: 0.8720 - precision: 0.7970 - recall: 0.6550 - auc: 0.9326 - val_loss: 0.5373 - val_accuracy: 0.8789
Epoch 12/20
256/256 [==============================] - 20s 77ms/step - loss: 0.5912 - accuracy: 0.8761 - precision: 0.7992 - recall: 0.6738 - auc: 0.9379 - val_loss: 0.5094 - val_accuracy: 0.8877
Epoch 13/20
256/256 [==============================] - 20s 77ms/step - loss: 0.5822 - accuracy: 0.8795 - precision: 0.7976 - recall: 0.6943 - auc: 0.9398 - val_loss: 0.5003 - val_accuracy: 0.8896
Epoch 14/20
256/256 [==============================] - 21s 82ms/step - loss: 0.5547 - accuracy: 0.8862 - precision: 0.8128 - recall: 0.7076 - auc: 0.9453 - val_loss: 0.4950 - val_accuracy: 0.8939
Epoch 15/20
256/256 [==============================] - 20s 77ms/step - loss: 0.5547 - accuracy: 0.8867 - precision: 0.8077 - recall: 0.7178 - auc: 0.9457 - val_loss: 0.4612 - val_accuracy: 0.9019
```


```
Epoch 15/20
256/256 [==============================] - 20s 77ms/step - loss: 0.5547 - accuracy: 0.8867 - precision: 0.8077 - recall: 0.7178 - auc: 0.9457 - val_loss: 0.4612 - val_accuracy: 0.9019
```


```
Epoch 16/20
256/256 [==============================] - 21s 82ms/step - loss: 0.5190 - accuracy: 0.8965 - precision: 0.8268 - recall: 0.7416 - auc: 0.9523 - val_loss: 0.4730 - val_accuracy: 0.8973
Epoch 17/20
256/256 [==============================] - 20s 77ms/step - loss: 0.5137 - accuracy: 0.8978 - precision: 0.8259 - recall: 0.7493 - auc: 0.9533 - val_loss: 0.4607 - val_accuracy: 0.9016
Epoch 18/20
256/256 [==============================] - 21s 82ms/step - loss: 0.4944 - accuracy: 0.9023 - precision: 0.8337 - recall: 0.7609 - auc: 0.9566 - val_loss: 0.4282 - val_accuracy: 0.9093
Epoch 19/20
256/256 [==============================] - 20s 77ms/step - loss: 0.4791 - accuracy: 0.9055 - precision: 0.8383 - recall: 0.7706 - auc: 0.9596 - val_loss: 0.4393 - val_accuracy: 0.9037
Epoch 20/20
256/256 [==============================] - 20s 78ms/step - loss: 0.4648 - accuracy: 0.9075 - precision: 0.8416 - recall: 0.7759 - auc: 0.9616 - val_loss: 0.4322 - val_accuracy: 0.9091
```

Activity 5: Save the Model

```
custom_xception_model.save('xception.h5')
```

Out of all the models we tried (VGG16, Resnet50, Inception V3 & Xception) Xception gave us the best training and validation accuracies (about 90%). So, we are saving Xception as our final model.

The model is saved with .h5 extension as follows
A .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. Use an image to test the model and augment the image.

```
from tensorflow.keras.preprocessing import image
import numpy as np

# Load and preprocess the image
img_path = '/content/Alzheimer_s Dataset/test/VeryMildDemented/27 (25).jpg'  # Replace with the actual path of your image
img = image.load_img(img_path, target_size=(150, 150))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array /= 255.  # Normalize the image

# Make predictions
predictions = custom_xception_model.predict(img_array)
```

Index positions:

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

Taking an image as input and checking the results

```
from tensorflow.keras.preprocessing import image
import numpy as np

# Load and preprocess the image
img_path = '/content/Alzheimer_s Dataset/test/VeryMildDemented/27 (25).jpg'  # Replace with the actual path of your image
img = image.load_img(img_path, target_size=(150, 150))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array /= 255.  # Normalize the image

# Make predictions
predictions = custom_xception_model.predict(img_array)

# Analyze the results
predicted_class = np.argmax(predictions[0])
confidence = predictions[0][predicted_class]

class_labels = ['NonDemented', 'VeryMildDemented', 'MildDemented', 'ModerateDemented']
predicted_label = class_labels[predicted_class]

print(f"The predicted class is {predicted_label} with a confidence of {confidence:.2f}.")
```

```
1/1 [==============================] - 2s 2s/step
The predicted class is ModerateDemented with a confidence of 0.72.
```

So, our model will give the index position of the label. In this case, the 3rd index is for Moderate Dementia, which has been predicted correctly.

7.2 Web Application Development:

Milestone 4: Application Building

In this section, we will be building an Alzheimer's Prediction web application called 'MindSync+' that is integrated to the Xception model we built. An User Interface is provided for the users where they must upload the image to get predicted results on the stage and intensity of Alzheimer's. The uploaded image is given as input to the saved model and the respective prediction is showcased in a different web page with their Key Metrics about how they can deal with the disease.

This section has the following tasks

- Building HTML Pages.
- Building Flask Python Code.

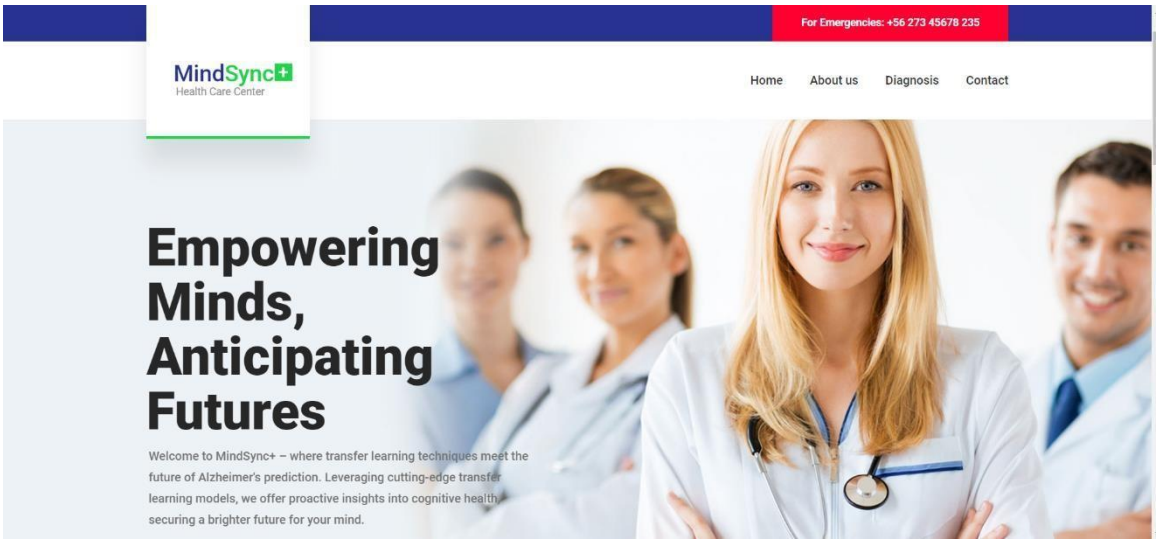
Activity1: Building Html Pages:

For this project create eight HTML files namely

- index.html
- about.html
- cog.html
- contact.html
- mild_dem.html
- mod_dem.html
- non_dem.html
- vmild_dem.html

The last four html pages are output pages. Based on the prediction, the user is redirected to one of these sites.

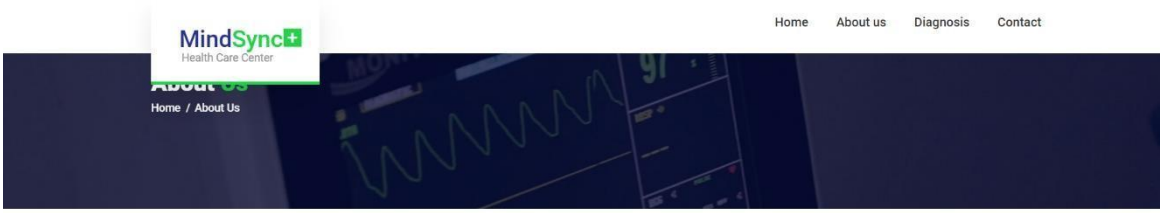
1. Let’s see how our index.html page (Home page) looks like:

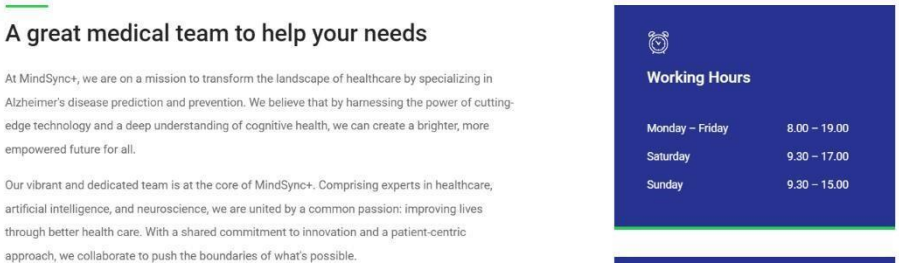


This is the page is our Landing Page and contains all the information about the service. The page has Links to all the four parts of application in the Dashboard ribbon. This contains all the information

- The timing to call our announcement
- Working days
- Useful Links to important website

2. Let’s see how our about.html page (About Us Page) looks like:

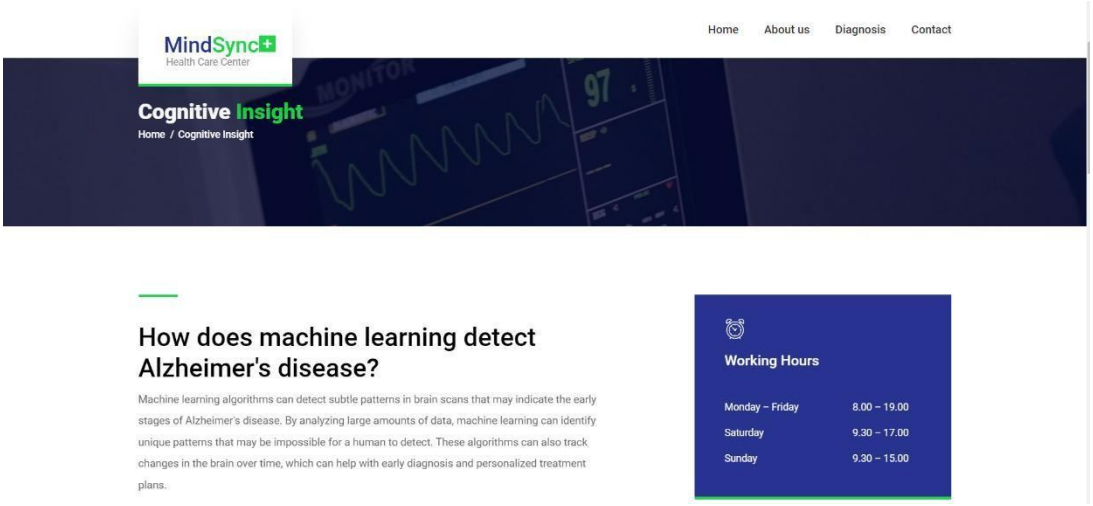




About Us page contain the following elements

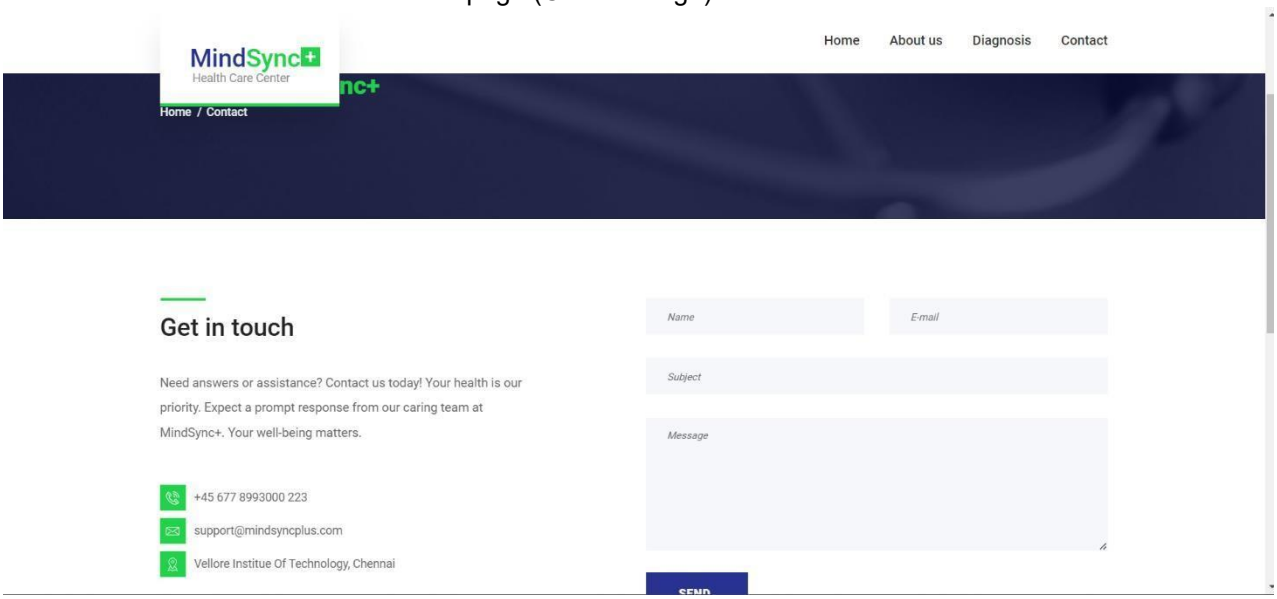
- Working Hours of the services
- Information about the team
- Useful links to the articles about Alzheimers

3. Let’s see how our cog.html page (Diagnosis Page) looks like:



- The 'Diagnosis' page is the page where the prediction process is performed.
- It provides a UI to the user where they can upload a scanned image of the brain by clicking the 'CHOOSE FILE' option.
- The uploaded image can be viewed in the 'Image Preview' section.
- Then the user should click the 'Submit' button.
- The uploaded image is sent to the saved model in the backend, where the prediction takes place.
- Based on the predicted result, the website redirects the user to one of the four result pages, where they can get more
  information about the current stage of their Alzheimer's disease and some guidance related to it.

4. Let's see how our contact.html page (Contact Page) looks like:

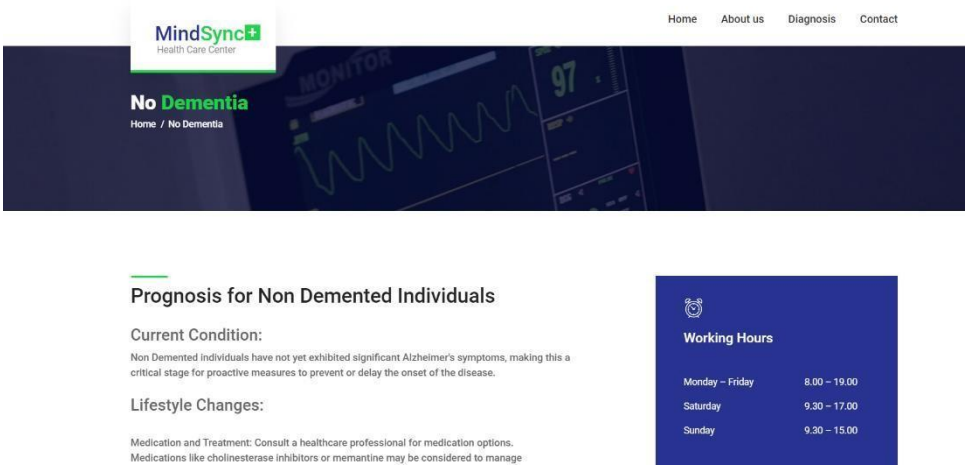


- The Contact page offers a UI to users, where they can provide feedback or register complaints by typing their name, email id, subject of concern, their message and then clicking the 'SEND' button.

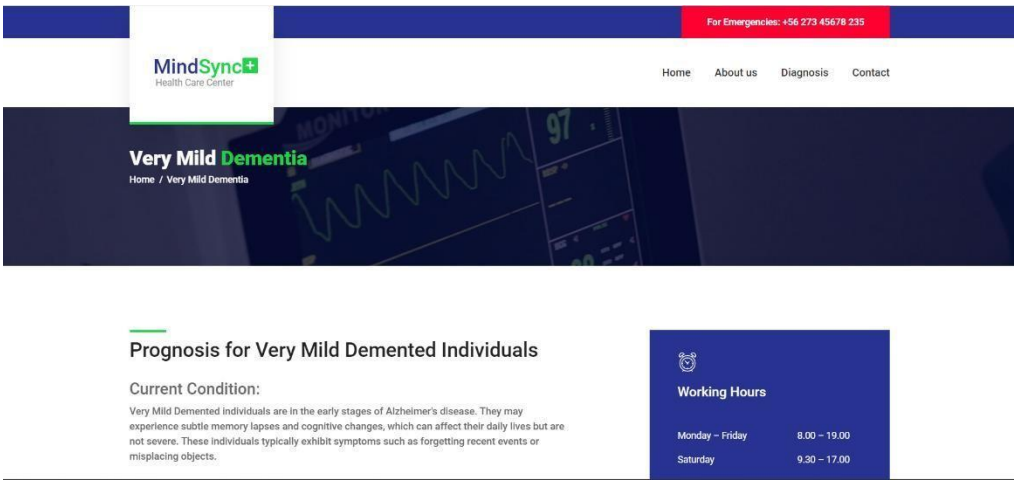
- They can also use the contact details or email id provided on the website for help or guidance purposes.

Result Pages:

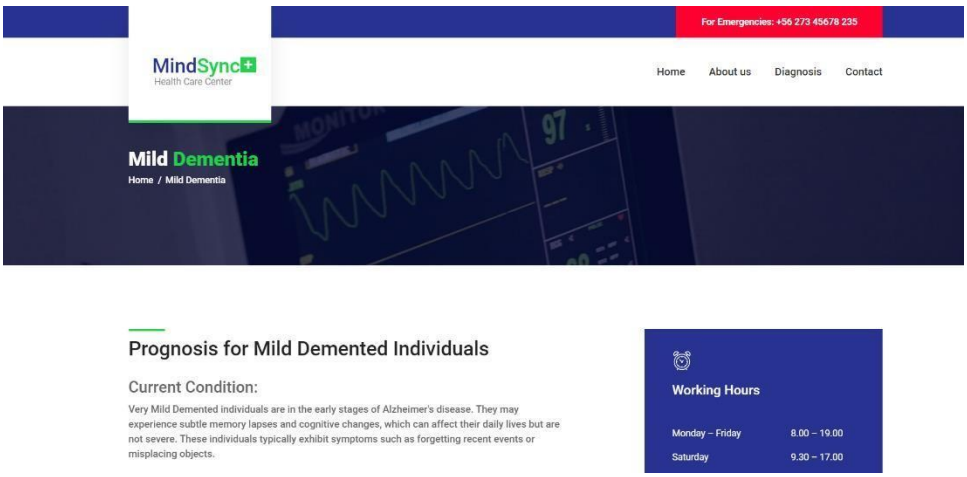
1. NonDemented:



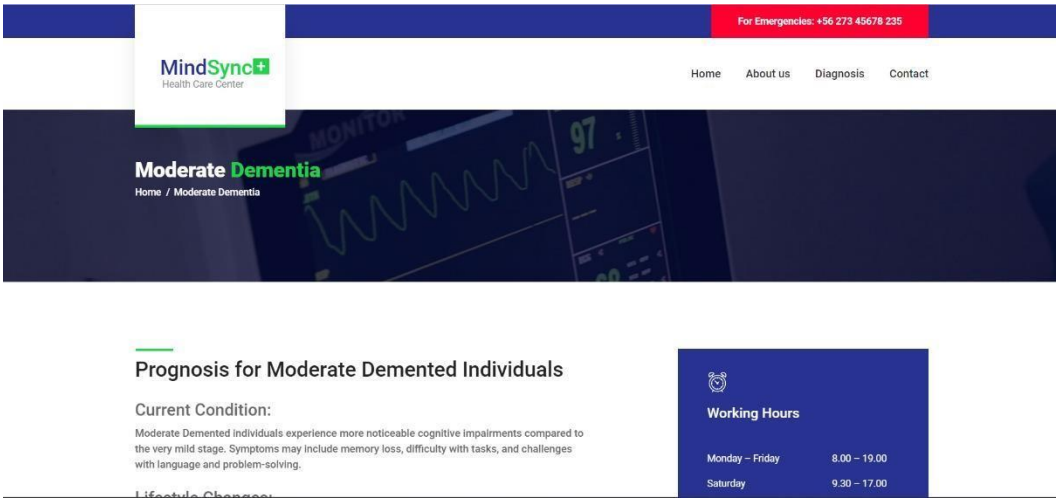
2. VeryMildDemented



3. MildDemented



4. ModerateDemented



Prognosis for Moderate Demented Individuals

Current Condition:

Moderate Demented individuals experience more noticeable cognitive impairments compared to the very mild stage. Symptoms may include memory loss, difficulty with tasks, and challenges with language and problem-solving.

Lifestyle Changes:

Working Hours

Monday – Friday	8.00 – 19.00
Saturday	9.30 – 17.00

Activity 2: Build Python code:

Import the libraries

```python
from flask import Flask, render_template, request
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from PIL import Image
import numpy as np
```

- Import above libraries and modules.

Initializing the flask app

```python
app = Flask(__name__)
```

Render HTML pages:

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

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

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

@app.route('/contact')
def contact():
    return render_template('contact.html')
```

- Set the routing pattern of the website for each of the main html pages – index.html, about.html, cog.html and contact.html using the render_template().
- Since the '/' route is bound to the index.html page, when we first open the website, the 'Home Page' is displayed.
- Since the '/about' route is bound to the about.html, when we click on the 'About Us' option we redirect to the 'About Us' page.
- Since the '/cog' route is bound to the cog.html, when we click the 'Diagnosis' option we redirect to the 'Diagnosis' page.

- Since the '/contact' route is bound to the contact.html, when we click the 'Contact' option we redirect to the 'Contact' page.

```python
@app.route('/predict_image_file', methods=['POST'])
def predict_image_file():
    if request.method == 'POST':
        img_file = request.files['file']
        img = Image.open(img_file)
        img = img.resize((150, 150))
        img = img.convert("RGB")  # Convert to RGB
        img_array = np.array(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array = img_array.astype('float32') / 255.0

        model = load_model("E:/Smartbridge_project/MindSync+/xception.h5")
        predictions = model.predict(img_array)
        predicted_class = np.argmax(predictions[0])
        confidence = predictions[0][predicted_class]

        class_labels = ['NonDemented', 'VeryMildDemented', 'MildDemented', 'ModerateDemented']
        predicted_label = class_labels[predicted_class]

        if (predicted_label == 'VeryMildDemented'):
            return render_template('vmild_dem.html')
        elif (predicted_label == 'MildDemented'):
            return render_template('mild_dem.html')
        elif (predicted_label == 'ModerateDemented'):
            return render_template('mod_dem.html')
        elif (predicted_label == 'NonDemented'):
            return render_template('non_dem.html')
```

- The post request is taken, and the files posted in the post method is open and transformations such as resize, rgb coding and Np array createing is done.
- The model is loaded from the Xception.h5 file and the image is predicted, and class of the image is taken in predicted_class
- The Class is used for redirecting the user to the correct page result page using else if page.

Main Function:

```python
if __name__ == '__main__':
    app.run(debug=True)
```

Activity 3: Run the application

- Open VSCode
- Upload all the required files and folders.
- Now click on the ‘run’ button for ‘app.py’.

```
[Running] python -u "e:\Smartbridge_project\MindSync+\app.py"
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 136-994-668
```

- Visit the website.
- Go to the ‘Diagnosis’ page.
- Click on ‘CHOOSE FILE’ and upload the image that needs to be tested.

- View the preview of the image in the ‘Image Preview’ section.
- Click ‘Submit’ to run the prediction.

8. Performance testing

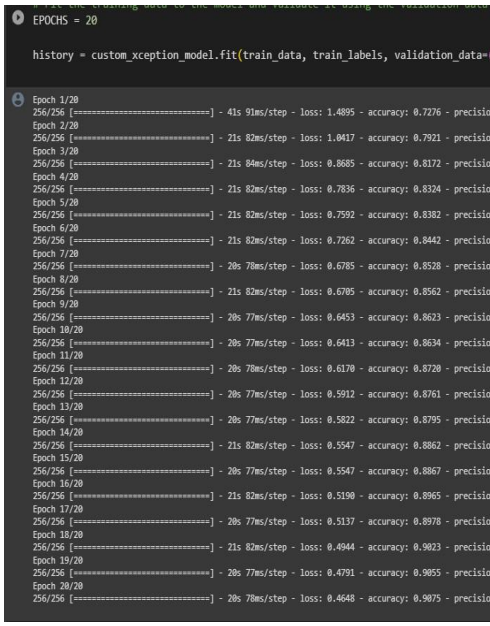
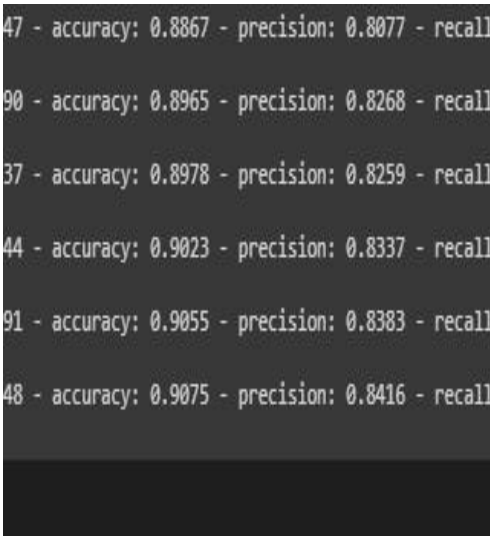
8.1 Performance Metrics:

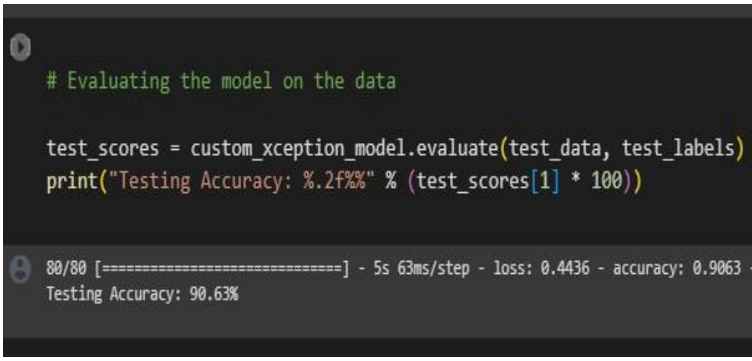
Model Performance Testing:

S.No.	Parameter	Values	Screenshot

1.	Model Summary	**The model used here is a customized version of the Transfer Learning Technique called Xception. It is a deep learning model which uses CNN layers to train its model effectively.**	

```
xception_model = Xception(input_shape=(150, 150, 3), include_top=False, weights="imag

for layer in xception_model.layers:
    layer.trainable = False

custom_xception_model = Sequential([
    xception_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="xception_cnn_model")

METRICS = [
    tf.keras.metrics.BinaryAccuracy(name='accuracy'),
    tf.keras.metrics.Precision(name='precision'),
    tf.keras.metrics.Recall(name='recall'),
    tf.keras.metrics.AUC(name='auc')
]

custom_xception_model.compile(optimizer='adam',
                              loss=tf.losses.CategoricalCrossentropy(),
                              metrics=METRICS)
```

2.	Accuracy	Training Accuracy - 90.75 Validation Accuracy – 90.63	 

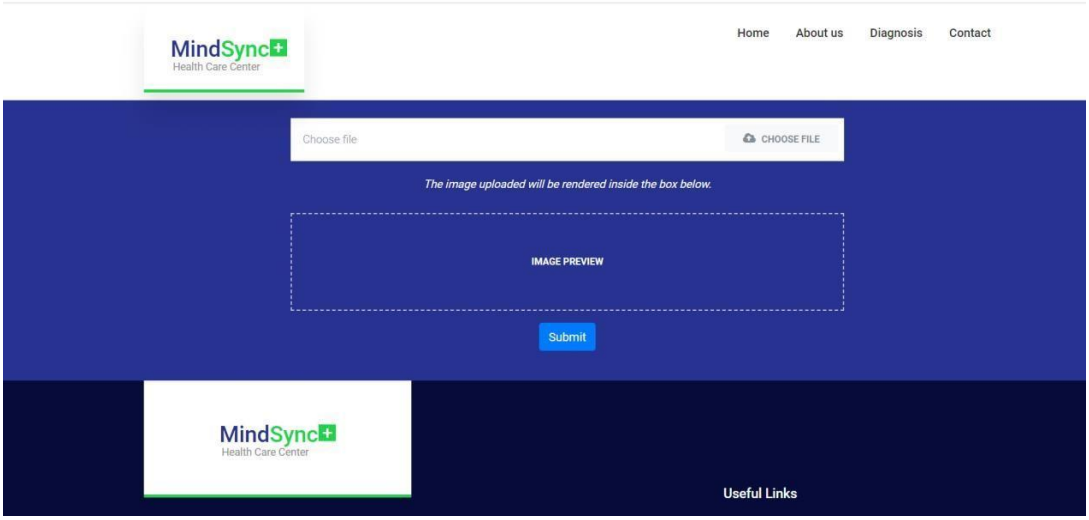
```
# Evaluating the model on the data

test_scores = custom_xception_model.evaluate(test_data, test_labels)
print("Testing Accuracy: %.2f%%" % (test_scores[1] * 100))
```

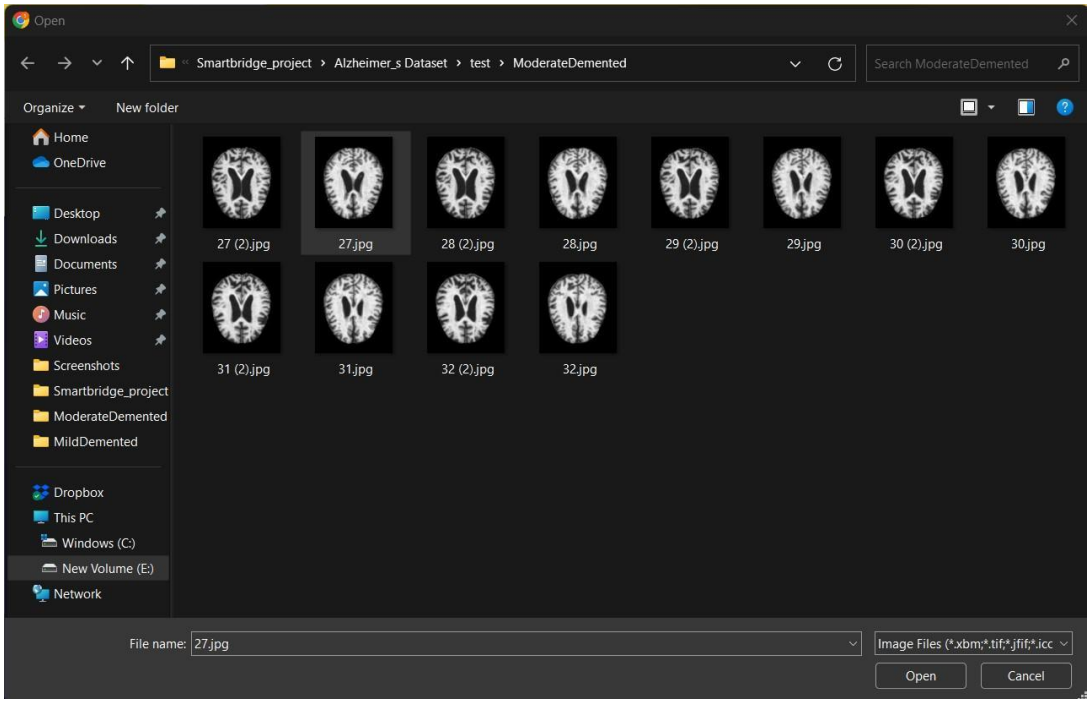
```
80/80 [==============================] - 5s 63ms/step - loss: 0.4436 - accuracy: 0.9063
Testing Accuracy: 90.63%
```

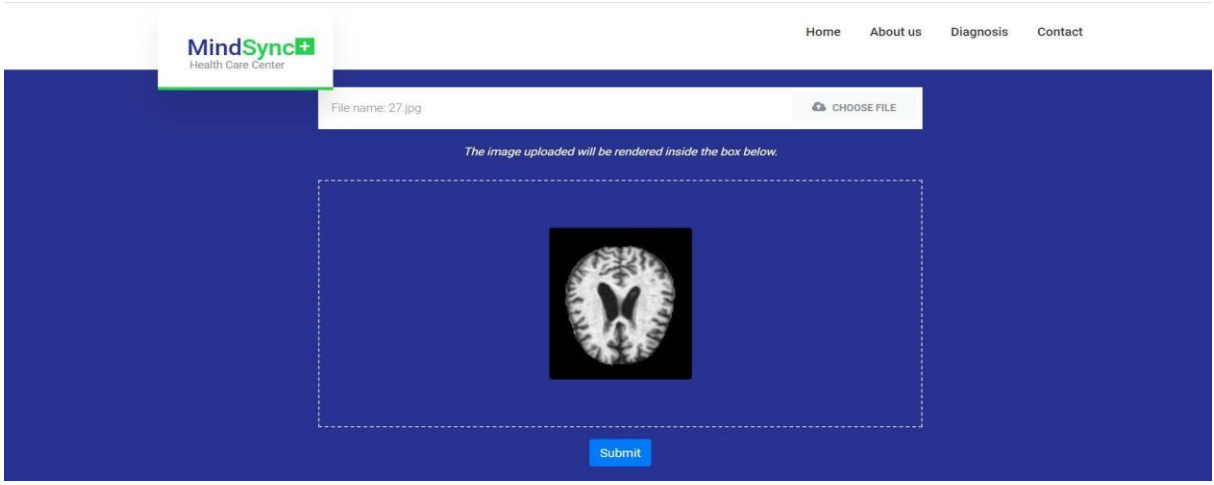

9. Results:

9.1 Output Screenshots:

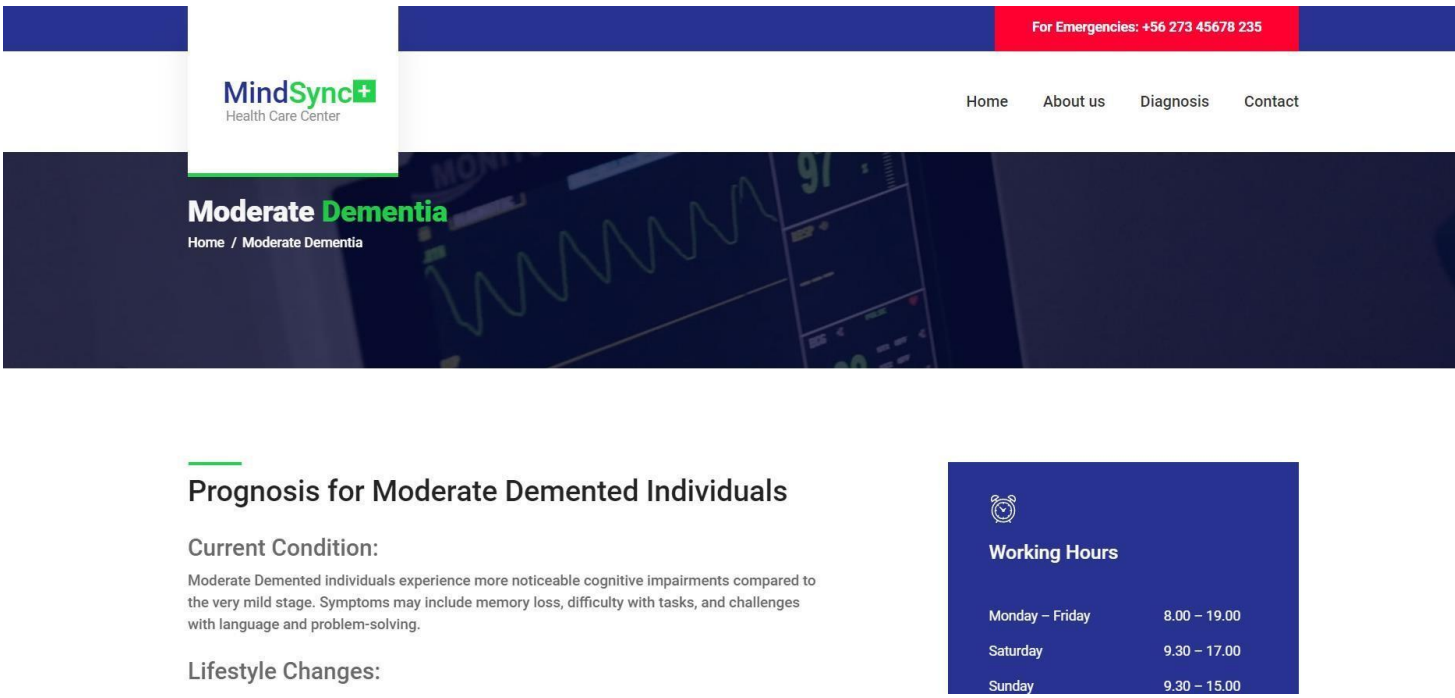
Input:




Open
Smartbridge_project › Alzheimer_s Dataset › test › ModerateDemented
Search ModerateDemented
Organize
New folder
Home
OneDrive
Desktop
Downloads
Documents
Pictures
Music
Videos
Screenshots
Smartbridge_project
ModerateDemented
MildDemented
Dropbox
This PC
Windows (C:)
New Volume (E:)
Network
27 (2).jpg
27.jpg
28 (2).jpg
28.jpg
29 (2).jpg
29.jpg
30 (2).jpg
30.jpg
31 (2).jpg
31.jpg
32 (2).jpg
32.jpg
File name: 27.jpg
Image Files (*.xbm;*.tif;*.jfif;*.icc
Open
Cancel


MindSync
Health Care Center
Home
About us
Diagnosis
Contact
File name: 27.jpg
CHOOSE FILE
The image uploaded will be rendered inside the box below.
Submit

Output:



10. ADVANTAGES & DISADVANTAGES

Advantages

High Accuracy: The Xception model demonstrates superior accuracy in predicting Alzheimer's disease based on neuroimaging data, providing reliable results for early diagnosis and intervention.

Robust Feature Extraction: The model's deep learning architecture allows for the extraction of intricate features from neuroimaging data, enabling the identification of subtle patterns associated with the progression of Alzheimer's disease.

Scalability: The system is scalable and can efficiently handle a large volume of neuroimaging data, making it suitable for use in large-scale clinical studies and real-world healthcare applications.

Disadvantages

Data Requirement: The Xception model requires a substantial amount of labeled data for training, which may be a limitation in cases where the availability of annotated neuroimaging data is limited.

Computational Complexity: Training the Xception model can be computationally intensive, requiring significant processing power and time, which could be a potential drawback for resource-constrained environments.

Interpretability: The inherent complexity of deep learning models such as Xception can make it challenging to interpret the model's decision-making process, potentially limiting the understanding of the underlying biological mechanisms of Alzheimer's disease.

11. CONCLUSION

In conclusion, the application of the Xception model in Alzheimer's disease prediction demonstrates promising results, with high accuracy and robust feature extraction capabilities. Despite the challenges related to data requirements and computational complexity, the model's scalability and potential for early diagnosis make it a valuable tool in the ongoing efforts to combat Alzheimer's disease.

12. FUTURE SCOPE

The future scope of this research lies in further enhancing the interpretability of the Xception model, exploring techniques to reduce the data requirements for training, and optimizing the computational efficiency to facilitate its practical implementation in diverse healthcare settings. Additionally, integrating multimodal data and exploring the use of transfer learning techniques could enhance the model's performance and broaden its applicability in the field of neurodegenerative disease research and clinical practice. Ongoing collaborations with neuroscientists and healthcare professionals can pave the way for the integration of this technology into routine clinical practice for early Alzheimer's disease detection and personalized treatment planning.

13. APPENDIX

smartinternz02/SI-GuidedProject-589116-1697035252