Prithivi Narayan Campus, Pokhara

Tribhuvan University

Institute of Science & Technology

Final Year Project Report

On

"SECOND OPINION" – AN AI BRAIN TUMOR CLASSIFIER

In partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science and Information Technology

Submitted By;

Srijan Basnet (30077/78)

Suraj Dhakal (30079/78)

Yojjal Sharma (30084/78)

Supervisor:

Tara Bahadur Thapa

Date:

2082/07/14

LETTER OF APPROVAL

This is to certify that this project by Srijan Basnet (**30077/78**), Suraj Dhakal (**30079/78**) and Yojjal Sharma (**30084/78**) titled "SECOND OPINION" in partial fulfilment of the requirement for the degree of BSc. In Computer Science and Information Technology has been well studied. In our opinion it is satisfactory in scope and quality as a project for the required degree.

……………………………….. Chudamani Subedi BSc. CSIT Program Coordinator Prithvi Narayan Campus	…………………………. Tara Bahadur Thapa Supervisor Prithvi Narayan Campus
………………………………… External Examiner IOST, Tribhuvan University	…………………………….. Internal Examiner Prithvi Narayan Campus
………………………………….. Prof. Dr. Hari Prasad Pathak Campus Chief Prithvi Narayan Campus	

SUPERVISOR RECOMMENDATION

I hereby recommend that this project be prepared under my supervision entitled "Second Opinion" in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology be processed for the evaluation.

……………………………

Tara Bahadur Thapa

Supervisor

Prithvi Narayan Campus

ACKNOWLEDGEMENT

We hereby declare that the project work entitled “Second Opinion” submitted to Institute of Science and Technology, Tribhuvan University, Kathmandu is an original piece of work under the supervision of Mr. Tara Bahadur Thapa and faculty members, Prithvi Narayan Campus, Pokhara and is submitted in partial fulfillment of the requirements for the award of degree of Bachelor of Science in Computer Science and Information Technology (B.Sc. CSIT). This project report has not been submitted to any other university or institution for the award of any degree.

Members:

Srijan Basnet (30074/78)

Suraj Dhakal (30079/78)

Yojjal Sharma (30084/78)

B.Sc.CSIT, 4th year

Prithvi Narayan Campus, Pokhara

ABSTRACT

This project aims to develop an automated brain tumor classification system that improves diagnostic accuracy and supports effective treatment planning by classifying tumors from Magnetic Resonance Imaging (MRI) scans. Brain tumors, which can be benign or malignant and commonly include gliomas, meningiomas, and pituitary tumors, affect approximately 11,700 people annually. This system provides the ViT architecture to predict the tumor class from the MRI Scan. This project employs the ViT-B/16 model with transfer learning for detecting robust image features for stable tumor classification. The paper demonstrates the potential of Vision Transformers in computer-assisted pathology and medical image processing to correctly classify tumors into glioma, meningioma, and types of pituitary tumors. The paper addresses the challenge of useful, reliable diagnostic tools in brain tumor imaging and paves the way for future studies on computer-aided medical diagnosis.

Keywords: ViT-B/16, MRI, Brain Tumor Classification, Classifier Model, Vision Transformer, Glioma, Meningioma,

LIST OF ABBREVIATIONS:

AI: Artificial Intelligence

AUC: Area Under Curve

CNN: Convolutional Neural Network

EHR: Electronic Health Records

ER Diagram: Entity Relationship Diagram

FPR: False Positive Rate

KNN: K Nearest Neighbor

MRI: Magnetic Resonance Imaging

ORM: Object Relational Mapper

PACS: Picture Archiving and Communication Systems

ROC: Receiver Operating Characteristics

SVM: Support Vector Machine

TPR: True Positive Rate

ViT: Vision Transformer

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Introduction

Brain tumors must be diagnosed as early as possible to be effectively treated. Radiologists typically use magnetic resonance imaging (MRI) to identify and categorize brain tumors. Manual MRI scan interpretation can, nevertheless, be time-consuming, subjective, and prone to human error, particularly in resource-scarce environments [1].

Current progress in deep learning and computer vision have potential for the computer-automated analysis of medical images to enhance quality and efficiency of the diagnosis of MRI Scan [2].This project suggests the construction of an artificial intelligence-based tool using the Vision Transformer (ViT) models to classify brain MRI images with classes Glioma Tumor, Meningioma Tumor, Pituitary Tumor, and Normal Brain. The aim is to develop an accessible web application usable by clinicians to obtain an automated second opinion on MRI scans for quick and trustworthy decision-making. This project consists of two classifier models, one for determining whether the input image is Brain MRI Scan or not and another for classifying the type of tumor among 3 classes (Glioma, Meningioma and Pituitary tumor) or 4th class normal brain.

The Second Opinion system will utilize transfer learning with transformer models, adapting them to the specific task of brain tumor classification. This system can help to reduce the workload and minimize human errors by providing faster and efficient reports for the MRI scan.

1.2 Problem Statement

The burden of brain tumors has been increasing in Nepal bu the current infrastructures face challenge in providing timely and accurate diagnosis. The lack of infrastructures leads to the delayed diagnosis of the MRI scan of the brain for any tumor. There is a requirement for a cost-effective and easily accessible solution to improve brain tumor diagnosis across the country. In Nepal, 39% of the brain tumor is identified to be glioma, 29% are meningioma and remaining 6% are pituitary according to Nepal Brain Tumor Center. Despite such expansion of brain tumor patients, there is still lack of doctors and infrastructures for proper diagnosis and treatment [3].

Brain tumors in Nepal are a severe health issue, with increasing incidence rates and limited availability of specialized diagnosis centers, particularly in rural and poor regions. In Nepal, the diagnosis of brain tumors is dependent on MRI scans interpreted by a limited number of specialist radiologists who are based in urban areas, leading to delayed treatment and diagnosis especially in rural and remote areas where expertise and imaging facilities are limited [4].

The absence of automated diagnostic tools creates these problems, particularly in remote locations with weak and poor healthcare infrastructure. An AI-based brain tumor classifier through deep learning and machine learning techniques can be capable of resolving these problems by providing a cost-effective and accurate way for early detection and diagnosis of brain tumors from MRI scans [5] [6].

1.3. Objectives

Second Opinion aims to provide an interactive web app for uploading the MRI scan so that it can classify the types of tumors find on the MRI scan or if not found any kind of tumor. This helps in improving the diagnosis efficiency and speed.

The major objectives and goals of the project are as follows:

1. To develop a classifier model that can classify if the image is a Brain MRI or not.
2. To develop a classifier model that can early detect 4 class of tumors from the MRI scan using a ViT model.
3. To create the user-friendly interactive web app to upload the MRI Scan and get the result for the MRI scan efficiently with minimal guidance.

1.4. Scope and Limitation:

1.4.1 Scope

The scope of this project is to develop an AI powered medical analysis tool that can assist doctors and medical specialists for early and efficient diagnosis of the MRI scan.

The system provides:

1. Automated Analysis of uploaded MRI Scan to find the class of the brain tumor if found.
2. Results with confidence of the AI model along with downloadable PDF format.
3. Web-based interface for the doctors to upload review results and manage patient data
4. Sharing of the report through email or downloading the PDF.
5. Secure User Authentication and Prediction History Management.
6. Error message when a non-MRI scan is uploaded.

1.4.2 Limitations

The limitations that this model suffers from

1. The AI models trained on specific datasets may not show consisted result to some rare conditions.
2. Currently The Model Supports only Image upload and doesn’t integrate with the Hospitals PACS or EHR systems.
3. The system is currently limited to a web application only.
4. The system works like a black box and does not clarify the reasoning behind predictions.

1.5 Methodology

For the development of this system, we have adopted a modified version of the waterfall model know as iterative waterfall model. In this approach, the tasks can be performed in parallel with other tasks and we can loop back to the previous phase if some feedback is received. This approach makes sure that feedback is well heard at any phase and so that we can move back to the phase requiring changes. Since we can also overlap different phases, we can divide the tasks to different team members for different tasks for each team member. This helps in faster development on a team-based project by running the overlapping phases and have the advantage of feedback and loops to the previous phases.

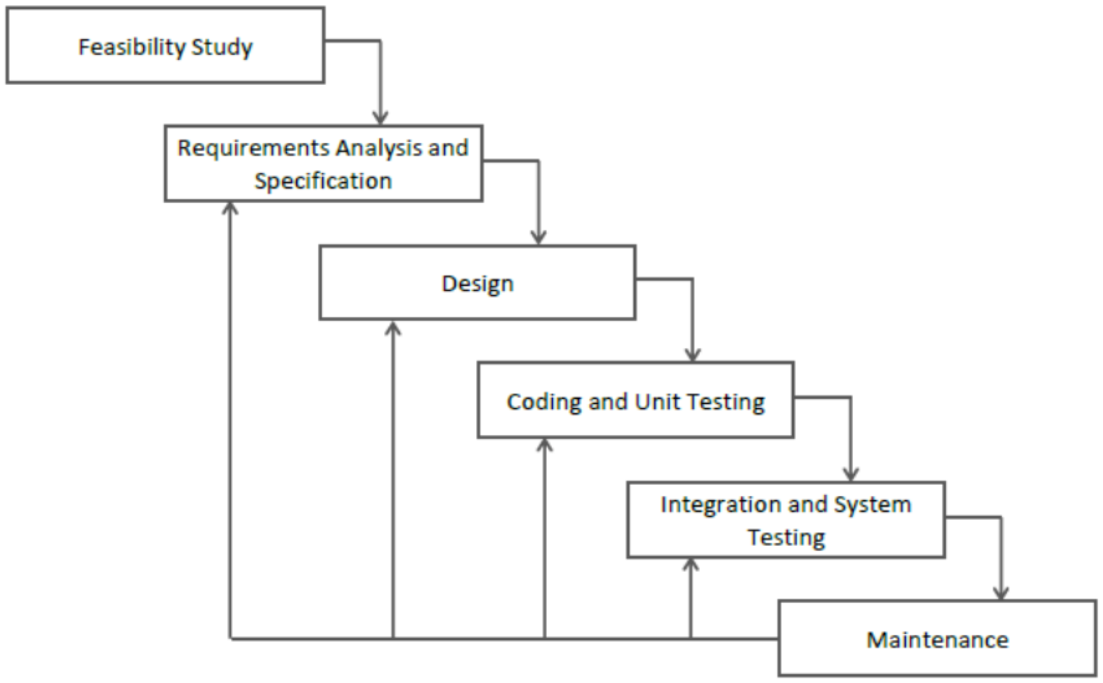

Feasibility Study
Requirements Analysis and Specification
Design
Coding and Unit Testing
Integration and System Testing
Maintenance

Figure 1.1:Block Diagram of Iterative Waterfall method

1.6 Report Organizations

Chapter 1 provides an overview of the project, the problem it tries to solve, the main objectives, scope areas and limitations and chosen Development methodology for completing this project and finally provides the overview of how the project report is organized.

Chapter 2 consists of the Background Study of the Project like fundamental theories, concepts and terminologies related to the project. The Literature Review in this chapter provides the overview of the similar research projects, papers, theories and findings by other researchers working on similar project for developing a context for the project.

Chapter 3 consists of System analysis, which tells about the in-depth working of the system and how the system does what it does. It provides various functional and non-functional requirements for the project and provides feasibility study based on different factors like technical, operational, schedule and economic aspects.

Chapter 4 shows the design of the overall system and how various parts of the system are related to each other. This chapter also provides the user interface design and detailed insights into algorithms used in the project.

Chapter 5 defines the implementation and testing of the project. It provides detailed explanation of the tools, frameworks, IDE's, database platforms, programming language used for the development process. This chapter also outlines the major testing performed in the project development phase. It lists the test cases done and the result analysis of the test. It also includes the trained model performance along with all the charts.

Chapter 6, The Final Chapter compacts the major conclusions and findings from the outcomes of the project. It also points out the potential future enhancements that can be done in the project for sustainability of the project.

Chapter 2: Background Study and Literature Review

2.1 Background Study

A brain tumor is the irregular growth of the brain cells and adjacent tissues. These growths can be benign or malignant, malignant is cancerous while benign is non-cancerous [7].

The pretrained CNN Models are initially trained on large dataset like ImageNet, learning general features, Fine tuning removes last few layers and adding new ones trained on the target dataset with the required class. ViT's are gaining popularity in replacement of the CNN Models. These models break down images into fixed size patches and process them by capturing global relationships between the pixels [8].

The medical imaging dataset have very few numbers of data, training the model from scratch is a very data hungry approach. ViT Transfer Learning can be used to deal with this issue. Due to these limitations of low availability of datasets, transfer learning has become defacto-standard for training on medical imaging datasets [9].

2.1.1 Terminologies related to The Project

1. Brain Tumor: Irregular growth of the brain cells and adjacent tissues [7].

2. Classification Models: Classification Models are the machine learning models that are designed to classify the data into predefined class or group according to their characteristics.

3. ViT: Vision Transformer is a deeplearning model for the image classification that applies the Transformer Architecture similar to the NLPs [8].

4. Transfer Learning: Transfer Learning is a technique where a model is not trained from the scratch but instead some layers of the model is fine-tuned according to the required training dataset and reuse the pretrained attributes and weights for better results [9].

5. Learning Rate: Learning Rate is the step size used by the optimizers to update the weights.

2.2 Study of Existing System and Literature Review

Current advances in artificial intelligence (AI) have significantly impacted the field of medical imaging, including in the classification of brain tumors from MRI scans. Previously, diagnosis was conducted through the personal interpretation of radiologists, which not only requires a lot of time but also is prone to human error. Early AI systems employed conventional machine learning methods such as Support Vector Machines (SVM) and K-Nearest Neighbors (KNN), which were dependent heavily on handcrafted features such as shape and texture. These are labeled to be non-scalable and provided suboptimal outputs. Deep learning, specifically Convolutional Neural Networks (CNNs), has revolutionized this branch by making end-to-end learning from raw MRI scans themselves possible. Pereira et al. (2016) demonstrated the effectiveness of CNNs for brain tumor segmentation issues, recognizing the activity to be extremely precise through hierarchical feature extraction on MRI images [10].Also, Swati et al. adopted transfer learning through pre-trained weights of model such as VGG16 and modified the heads for multiple class brain tumor classification with progress in accuracy and performance [4]

A ViT model can be used efficiently in the medical images as Sukhendra et al. (2024) proposed a ViT model that is trained on chest x-rays for pneumonia detection. The proposed model achieves 97.61% accuracy, 95% sensitivity (recall), and 98% specificity in pneumonia detection, surpassing conventional CNN architectures on a benchmark public CXR dataset. ViT uses self-attention methods to capture complex global and local dependencies in chest X-ray images, which provides generally better spatial feature extraction which can improve performance [11].

Despite these advances, current systems are limited in that they are dependent on large, labeled datasets, have reduced generalizability across institutions due to scanner variability, and lack explainability of AI-driven decisions. Most of the current models are either best for binary classification or tumor segmentation only, which hinders their real-world applicability in end-to-end diagnostic pipelines. Our system attempts to bridge these loopholes by incorporating sophisticated deep learning techniques with the capacity of making 4 class predictions, improved generalization, and robust accuracy. It also checks whether the Uploaded Image is a MRI Scan of brain or not.

2.3. Data Collection Method

The Dataset for the purpose of training the model is collected from Kaggle. The original data is shared by Sartaj in the Kaggle [12]. The data consists of about 901 images for glioma tumor, 913 images for meningioma tumor, 844 images for pituitary tumor and 438 images for normal brain MRI scan. The dataset was split in 75% training set and 25% testing set. Some transformations made to the dataset before training are as follows:

1. Random rotation chance up to 15 degrees to account for slight variations in patient head positioning or scanner alignment.
2. Random horizontal and vertical flip with 25% chance for each to encourage the model to focus on tumor-related texture and intensity features rather than strict spatial orientation.
3. Color jitter with brightness and contrast 0.2 each to simulate natural intensity fluctuations caused by scanner hardware and tissue density.

Additionally, the following transformation take place for both training and testing sets:

1. All Images are resized to size 224*224-pixel size with center crop to align with the standard input size used for ImageNet-pretrained ViT models, ensuring compatibility with pretrained weights and efficient patch partitioning.
2. Normalization using the ImageNet mean/std (mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225]) to ensure that pixel intensity distributions are scaled consistently across all images.

We express our sincere gratitude to the original authors for their invaluable work in data collection.

Chapter 3: System Analysis

3.1 System Analysis

This section of the report defines the system's requirements and feasibility. The functional requirements and non-functional requirements have been described in this section. This chapter employs the object-oriented approach for defining the functional requirements for the projects. It uses tools like use case diagrams and ER model diagrams.

3.1.1 Requirement Analysis

i. Functional Requirements

Our system is designed to fulfill a set of functional requirements that define its core operations and user interaction.

- **Web-Based Interface**

  The system must offer a user-friendly graphical interface accessible via a web browser. This interface must be easy to use and navigate.

- **Image Upload**

  User must be able to upload the MRI scan to the system for classification.

- **Display of Results**

  After Classification, the system must provide the predicted category of the MRI Scan along with confidence score i.e. probability of the correctness of prediction.

- **Handling Low Confidence**

  In the case of low confidence, the system must classify the upload image as "Unknown/Uncertain". This is required to mitigate the risk of incorrect diagnosis.

- **Multi-Class Classification**

  The system must classify the MRI Scan into one of the following possible categories:

    - Glioma Tumor
    - Meningioma Tumor
    - Normal Brain
    - Pituitary Tumor

- **Keep Patient Records**

The system must keep record of the patients whose MRI scan have been analyzed along with the results so that doctor can view and provide further review to the patients.

- **Downloadable Report**

  The final result report can be printed and sent to the respective patients.

ii. Non-Functional Requirements

Non-functional requirements provide the quality standards that the system must meet like efficiency, security, usability, reliability.

- **Prediction Latency**

  The system must provide the result for any MRI scan as fast as 5 secs on a device with a standard CPU-based configuration.

- **Security**

  The system must provide secure password handling using JWT and It requires secure OTP verification for signing up and forget password.

- **Ease of Use**

  The web interface for the system must be easily understandable and user-friendly to interact with and won't require previous technical training and with minimal guidance.

- **Clarity of Output**

  The analysis result must be presented in a clear and structured format so that user can see proper results of the AI model.

- **System Stability**

  The system should be stable and should not fail often during standard use cases.

- **Code Readability**

  The codebase must be clean and well-documented with comments and function's docstring improving maintainability and readability.

- **Hardware Independence**

  The system must run on a basic device with standard CPU Configuration without need of any GPU Acceleration.

3.1.2 Feasibility Study

i. Technical

The system is technically feasible as it depends on libraries like PyTorch and Torchvision which are considered as the industry standard for the Deep Learning and Computer Vision task. The deployed model can easily run on a standard CPU configuration and hence removes the requirement for any GPU. The frontend is done in React and backend is done in FastAPI. Both are popular and consists of a well-made documentation to follow.

ii. Operational

The web-based interface provides ease of use for the project where user can access the project from any internet accessible device. It has a very user-friendly UI that would require very little guidance to use. It can be easily integrated with the current workflow to reduce the diagnosis delay.

iii. Economic

The project is economically viable as it requires minimal financial cost. All the software and programming languages used are free and would not cost anything. Only cost required here is the time of the developers, since there is no need for expensive hardware and software purchases. Also, the dataset needed for training and evaluation is already available from the Kaggle, avoiding costs related to data acquisition or preprocessing.

iv. Schedule

The project is viable according to schedule due to well documented frameworks and lack of any foreseeable obstacles. The expected project duration for this project is about 1 and a half months.

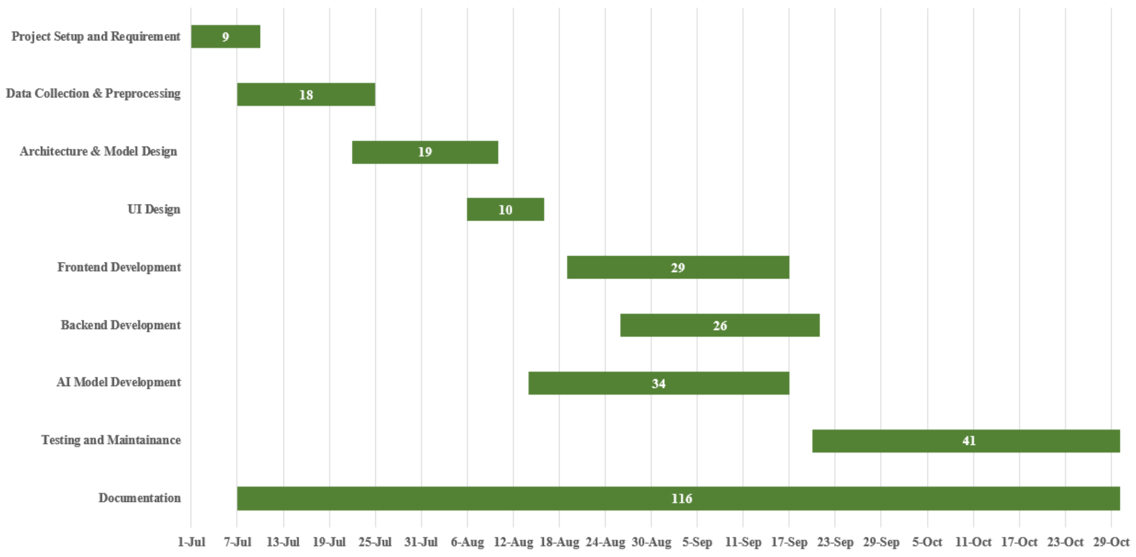


Figure 3. 1: Gantt Chart

3.1.3 Analysis

The system operates through a user-friendly web interface where a user uploads a brain MRI scan. The image is processed to match the input requirements of a Vision Transformer (ViT-B/16) model that has been fine-tuned for this specific task. Finally, the model's custom classifier head calculates the probabilities for each class: Glioma, Meningioma, Pituitary, or Normal and presents the diagnosis result and confidence score to the user, providing an efficient and accessible tool for automated medical image analysis.

i) Data modeling using E-R Diagram

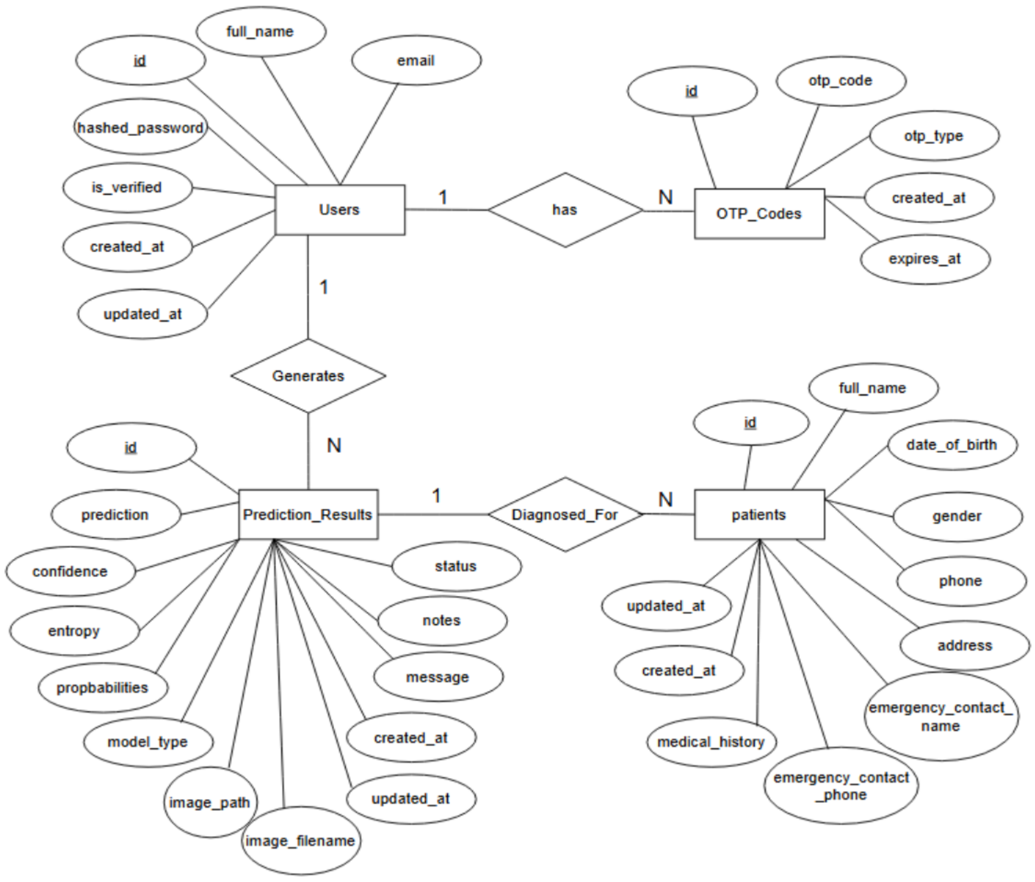


Figure 3. 2: ER Diagram

ii) DFD – Level 0 Diagram

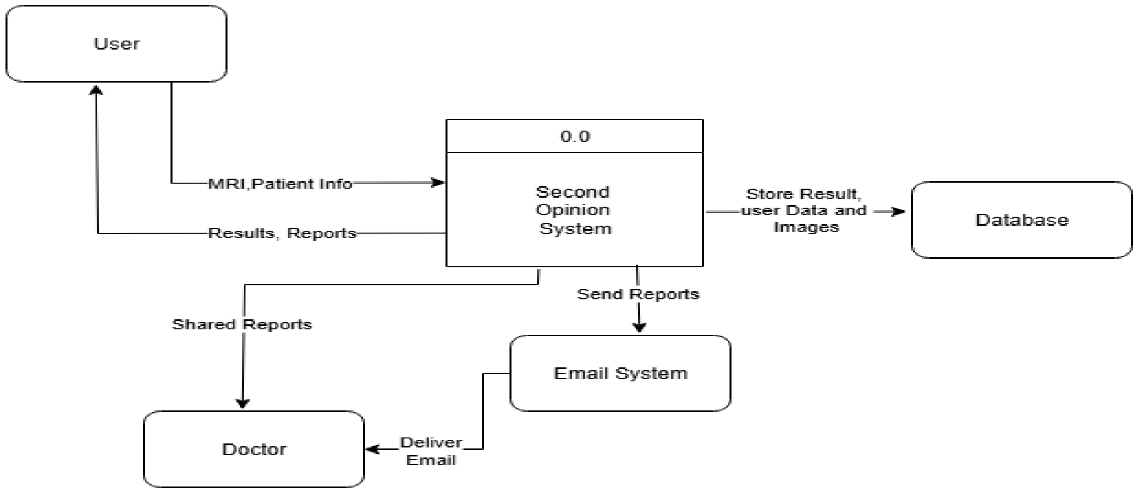


Figure 3.3: DFD- Level 0

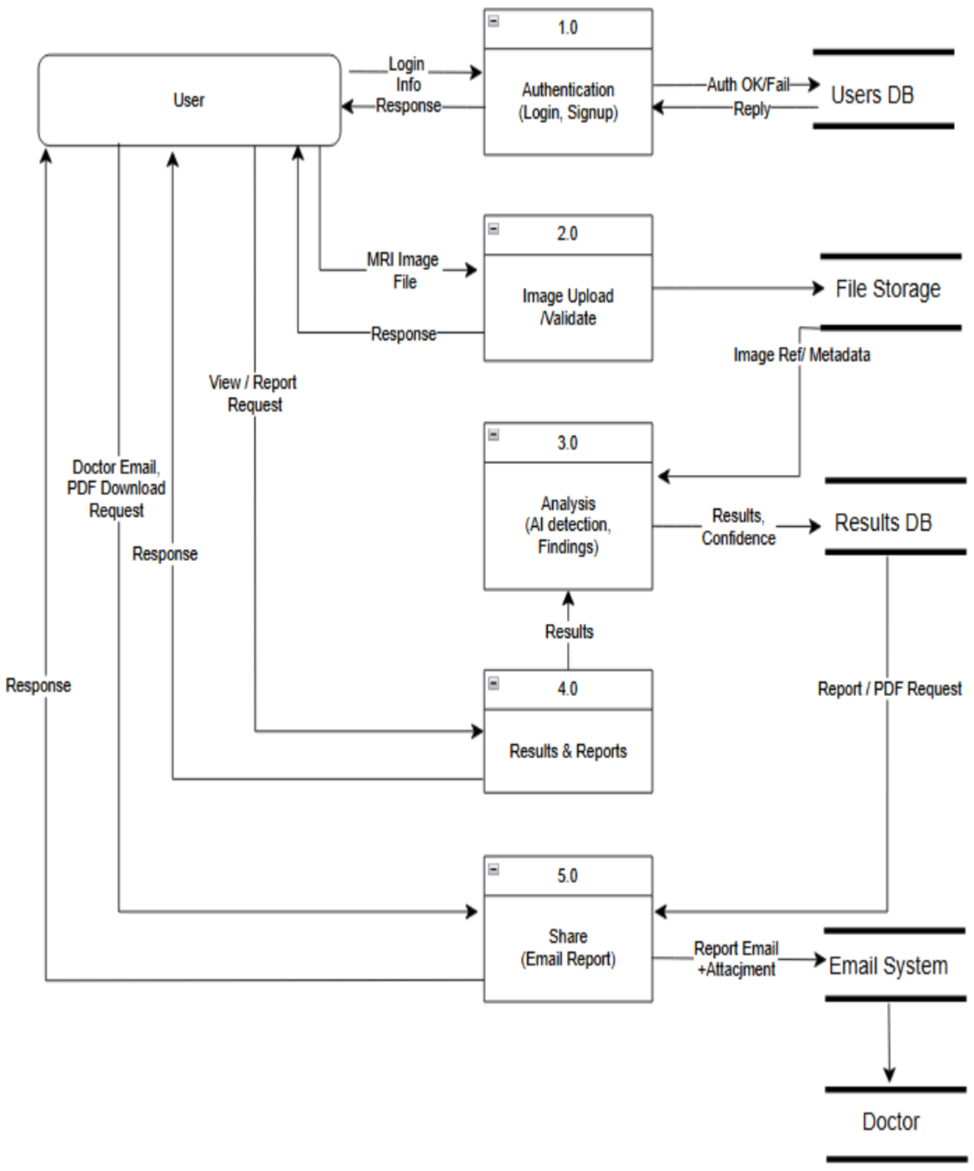


Figure 3. 4: DFD Level- 1

Chapter 4: System Design

4.1 Design

4.1.1 Database Design:

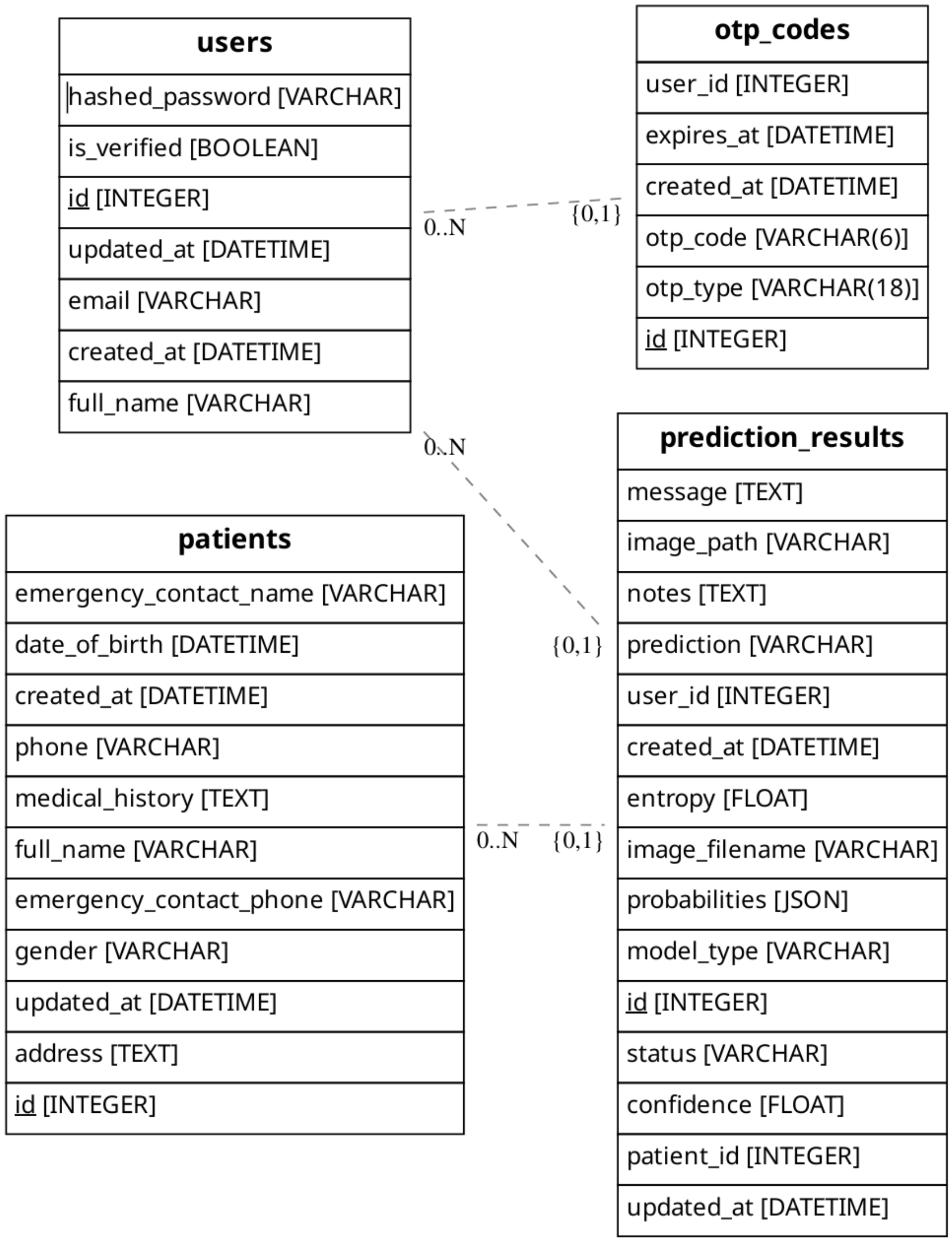
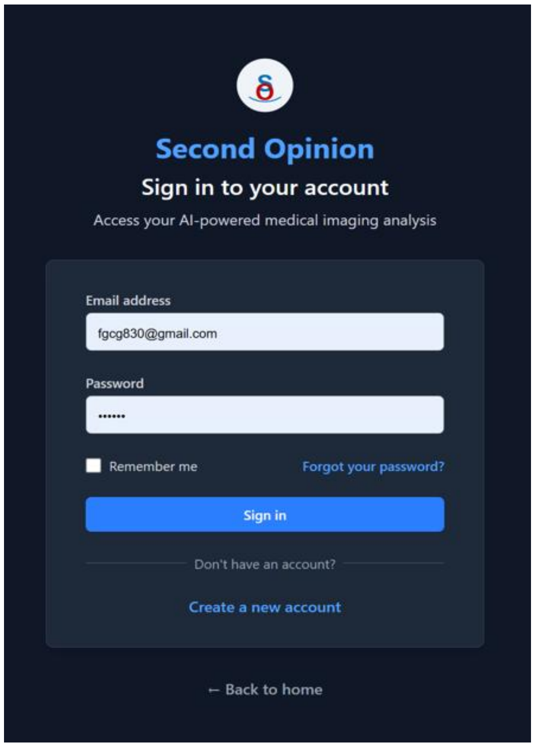


Figure 4. 1: Schema Design

4.1.2. Forms Design



Second Opinion

Sign in to your account

Access your AI-powered medical imaging analysis

Email address

fgcg830@gmail.com

Password

••••••

Remember me

Forgot your password?

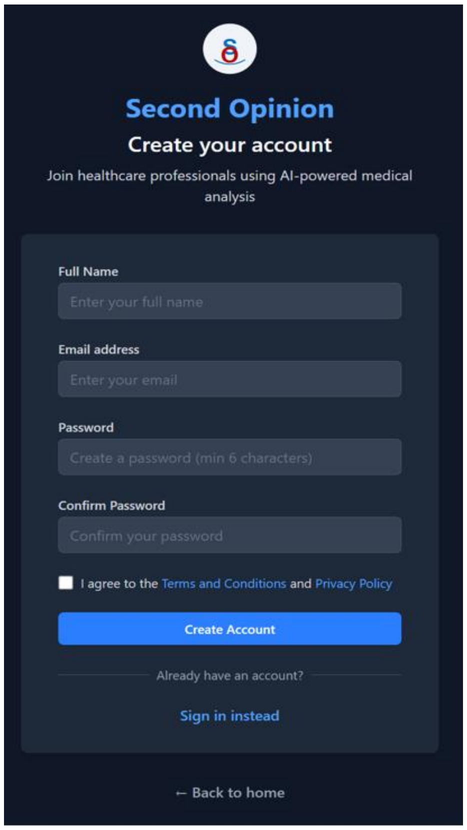
Sign in

Don't have an account?

Create a new account

← Back to home

Figure 4. 2: Login Form



Second Opinion

Create your account

Join healthcare professionals using AI-powered medical analysis

Full Name

Enter your full name

Email address

Enter your email

Password

Create a password (min 6 characters)

Confirm Password

Confirm your password

I agree to the Terms and Conditions and Privacy Policy

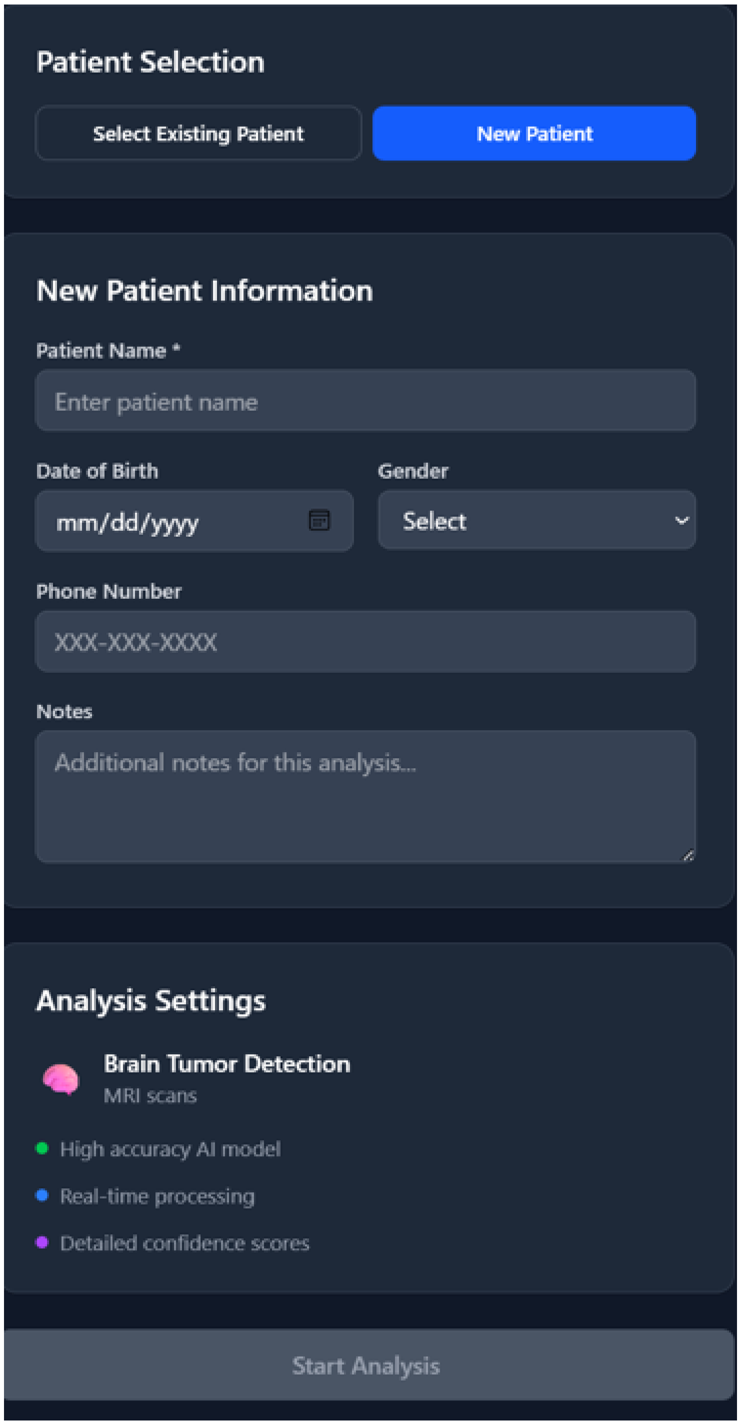
Create Account

Already have an account?

Sign in instead

← Back to home

Figure 4. 3: Signup Form


Patient Selection

New Patient Information

Patient Name *

Enter patient name

Date of Birth: mm/dd/yyyy

Gender: Select

Phone Number: XXX-XXX-XXXX

Notes

Additional notes for this analysis...

Analysis Settings

Brain Tumor Detection

MRI scans

- High accuracy AI model
- Real-time processing
- Detailed confidence scores

Figure 4. 4: Patient Information Fillup Form

4.2 Algorithm Details

1. Transfer Learning:

The transfer learning approach is used in this system while training the model. During training the model, the pretrained weights of the EfficientNetB2(used for MRI and non-MRI) and ViT_B_16(used for classifying the MRI brain scan for tumor) are freezed preserving the knowledge gained from ImageNet Dataset. The model uses these pretrained weights to extract the features of the image from the image. Then the model's head is replaced by domain specific classification header [13]

2. Adam Optimization:

The model uses adam optimization for optimization during model training. It combines the best properties of two other optimization technique:

- Momentum: remembers past gradients for smooth updates
- RMSProp: adapts learning rate for each parameter based on history of gradients

For each parameter θ at time t:

$m_t=\beta_1 m_{t-1}+(1-\beta_1)g_t$ # Update biased first moment

$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ # Update biased second moment

$\hat{m}_t = mt / (1 - \beta_1^t)$ # Bias-corrected first moment

$\hat{v}_t = vt / (1 - \beta_2^t)$ # Bias-corrected second moment

$\theta_t = \theta_{t-1} - \alpha * \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon)$ # Update parameters

Where,

α= learning rate

β_1=momentum decay(default)

β_2=variance decay (default:0.99)

ε= small number to avoid division by zero (default: 1e-8)

optimizer = torch.optim.Adam(params=effnetb2.parameters(), lr=1e-3) [14]

Our model using learning rate 0.001 and all other default values for Adam Optimization.

3. Cross Entropy Loss Function:

It is one of the most common loss functions used for classification models. It is used to measure the difference between true labels and the predicted label.

The smaller the entropy better the model predictions.

$$Loss = -\sum(y_i * \log(p_i))$$

Where,

y_i=1 if true class is i, else 0

p_i= predicted probability for class i

This runs for total of the no. of class present in our case 4, since there are 4 classes for the classification [15]

Chapter 5: Implementation And Testing

5.1 Implementation

5.1.1 Tools Used

The coding phase is considered as the phase for the development of code that will implement the design mentioned in the previous chapters. To achieve this, we need to use various tools for actual progress. Some of the tools used in the phases of Development of This Program are:

5.1.1.a Frontend Tools

Frontend provide the interface of the system i.e. way of interacting to the system. Different frontend Frameworks used in This system are:

- **React:** React is a JavaScript Framework that is a major part of this project. React is used to build the single page web applications. It was developed by Facebook now known as meta and now is used across the globe for creating the frontend.
- **Tailwind CSS**: Tailwind CSS is a CSS Framework that makes it easy to write the CSS. It integrates well with React and Vite. It allows developers to write the CSS faster by directly writing the shorter css in the *className* of the react components.

5.1.1.b Backend Tools

Backend is the backbone of this project. It handles the logic, data and communication between the database, server and frontend. The backend tools used in this system are:

- **FastAPI:** FastAPI is a backend framework used in this project. FastAPI is a python backend framework which also supports automatic documentation using the Swagger UI docs. This system mainly uses the FastAPI backend for all the tasks like authentication, image upload, prediction and history record management APIs.
- **SQLAlchemy:** SQLAlchemy is a python library that is used to work with database in easier and more structured way without having to know the raw SQL queries. It acts as an ORM to make the database feels closer to Python than SQL by developing tables as Python classes.
- **SQLite:** SQLite is the database used in this system. It is a lightweight, file based relational database, SQLite unlike PostgresSQL or MySQL doesn't run it's own server

but rather stores the database in the single file on disk. It doesn't require any extra hassle and configuration before-hand. SQLite is cross platform allowing it to work consistently on any of the system whether it is Windows, Linux or macOS.

5.1.1.c Training Tools:

- **PyTorch:** PyTorch is a python library used for deep learning developed by Facebook's AI Research. In this system, Pytorch is used to define the model, creating a custom liner layer in the model's head, device management for CPU and GPU based device, to call the optimization methods like Adam Optimizer.
- **Torchvision:** TorchVision is a python library used mainly for Computer Vison task. Torchvision is used for performing image preprocessing tasks and managing the weights of the model.

5.1.1.d Other Tools

- **Vs Code:** VS Code is the primary text-editor that is used to write all of the code of this system. The various extensions available in the VS Code for python and React are installed for the ease of coding. VS Code provides a good support for various languages including python and javascript so we went with VS Code as goto Text Editor for writing the code.
- **Git and Github:** Git is the major version control system that we used in the progress of this system. It enabled us to work collaboratively as a team while developing the project. It helped us to keep track of the project progress. Github is the major central repository where we pushed our code to. It integrates well with the Git Version Control and enables good collaboration and progress tracking.
- **MS-Word:** MS-Word is a tool used for the creating the report of the system. It is a Text Editor that makes ease for creating the documents.
- **Draw.io:** Draw.io is a free online drawing tool that we used to create the different diagrams that are needed to be drawn while developing the system like use-case diagram, flowchart, activity diagram.
- **uv:** uv is a python package manager and project manager which is written in Rust hence is faster and more reliable than other alternatives.

5.2 Testing

5.2.1 Unit Testing:

Unit testing is a type of testing where individual units are the subjects of the test. Its main purpose is to know if each individual unit works correctly independently of each other.

The following test cases were used for verifications:

Table 5.1: Test case for User Login:

Test Id	Test Case	Input	Expected Output	Observed Output
1	Unsuccessful User Login	Email: haha@gmail.com Password: Apple1	Incorrect email or password, or account not verified	Incorrect email or password, or account not verified
2	Successful User Login	Email: fgcg830@gmail.com Password: Passpass1	Login success and redirect to the dashboard	Login success and redirect to the dashboard

Table 5.2: Test Case for Forgot Password:

Test ID	Test Case	Input	Expected Output	Observed Output
1	Not Registered Email	Email : Okok123@gmail.com	Email address not found. Please check your email or sign up for an account.	Email address not found. Please check your email or sign up for an account.

2	Registered Email	Email: fgcg830@gmail.com	Email found Registered and sent OTP for reset password	Email found and the otp was sent

Table 5.3: Test Case for Upload Module

Test ID	Test Case	Input	Expected Output	Observed Output
1	Different Format Upload	Providing The Different format files other than image in upload section	Please upload a valid medical image (JPEG, PNG, or DICOM format)	Please upload a valid medical image (JPEG, PNG, or DICOM format)
2	Upload The image	Providing the image upload in the upload section	The image should be uploaded and then a preview of image should be shown	The image is uploaded and preview of image should be shown

Table 5.4: Test Case for Image Analysis

Test ID	Test Case	Input	Expected Output	Observed Output

1	MRI image identification Success	Upload The MRI Image and press on Start Analysis	The image should be analyzed by the model and redirect to the Result Details Page	The image is analyzed and application redirects to the Result Details Page
2	Non-MRI Image Identification	Upload a non - MRI image in the upload section	It should show Image Validation Failed: This appears to be a Non-MRI image	It successfully identifies the Non MRI Image and Show the error on the Upload Page.

5.2.2 System Testing

System Testing is the testing of complete and fully integrated system as a whole. System Testing is performed on the whole Existing System without seeing the small modules but how those modules incorporate with each other. It falls in black box testing, which means the code knowledge is not required in this testing.

Table 5.5 System Testing

Test ID	Test Case	Input	Expected Output	Observed Output
1	Succes Tumor Detection	Upload Image with Brain Tumor	Should Predict The type of tumor correctly	Predicts the type of tumor correctly.

2	Succes Normal brain Detection	Upload Image with Normal Brain MRI	Should predict as normal brain	Predicts the normal brain.

5.3. Result Analysis

Through the observation of the above test cases, we can discern that system is working as expected without any failures. The system correctly provides the authentication for the users and lets user upload the MRI Scan images for analyzing the MRI scan of the brain to find the classification of the tumor among 4 classes using the trained ViT model.

Main focus is to provide the doctors with a minimalistic web interface that is easy to use without any guidance on a standard CPU-configured Device.

Overall, from the above testing it is verified that the system is working as expected without any problems

5.3.1 Model Analysis

The Model when tested on the test set provided 96.89% accuracy. 750 out of 774 predictions were correct. The model finished training after 61 epochs with early stopping.

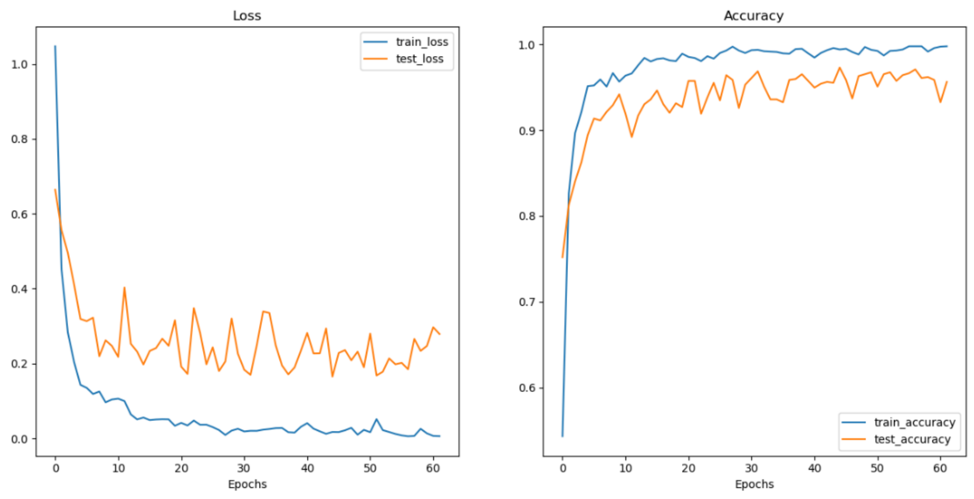


Figure 5.1 Loss and Accuracy Graph

From Above graph, we can see from the accuracy graph that the model achieves high accuracy on both train and test sets which stabilizes around (0.92-0.97).

The small gap between test and train loss shows very little overfitting but overall good generalization. The model provides a stable performance approximately after the first 10 epochs.

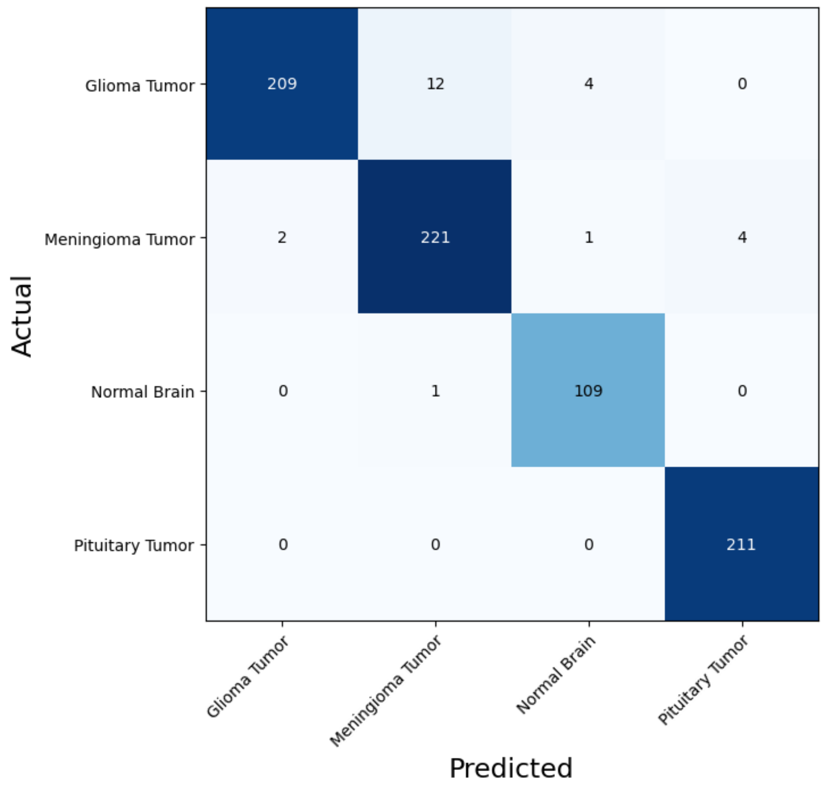


Figure 5.2: Confusion Matrix

The Confusion Matrix show that the model correctly predicts the Glioma Tumor 209 and gets confused with Meningioma 12 times. Also, Meningioma is predicted correct 221 times and Normal Brain Prediction is nearly perfect, and Pituitary is perfect as it predicts all 211 to same class without any errors. This concludes that model has a little confusion between the Glioma and Meningioma tumor class. The confusion for Glioma and Meningioma is quite common with radiologists too. Glioma and Meningioma can present similar shape, signal intensities and textures on MRI Scans [16].

Table 5.6 : Precision, Recall and F1 Score

Class	Precision	Recall	F1
Glioma Tumor	0.9905	0.9289	0.95879
Meningioma Tumor	0.9444	0.9693	0.9567
Normal Brain	0.9561	0.9909	0.9732
Pituitary Tumor	0.9814	1	0.9906

Also above shows precision, recall and f1 score. According to this figure, the model is pretty accurate among all classes however the weakest point is Glioma with a recall of 92.89 misclassified as Meningioma or normal brain a few times.

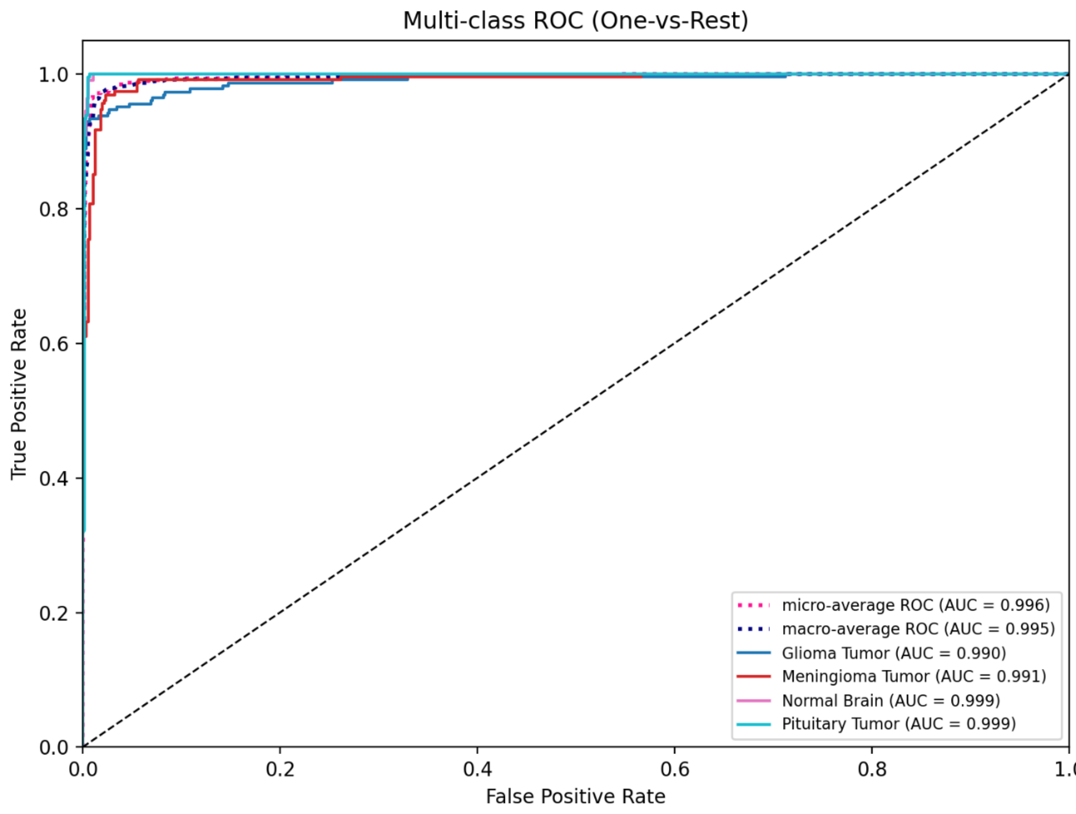


Figure 5.3: ROC Curves for Different Classes

The model achieves high sensitivity with very low FPR across classes. The near equality of the micro-average AUC and macro-average AUC (0.996 vs 0.995) shows that performance is consistently strong across classes. The model separates all four categories extremely well as even at small FPRs, TPRs are already very high.

Chapter 6: Conclusion and Future Recommendation

6.1 Conclusion:

The System is built using React, FastAPI and SQLite for web Application and Pytorch, TorchVision for training the ViT model's head with 4 classes (where 3 are types of tumors and one is normal brain) to effectively classify the MRI Scan to any one of the tumor or normal brain. The system is developed to provide ease for the doctors for the medical diagnosis of the MRI Scan and reduce the time required for diagnosis. This system intends to solve the delay of the MRI scan diagnosis.

6.2 Future Recommendations

This system is intended to make ease for the doctors to reduce the diagnosis of the MRI Scan of the brain for the doctors. However, there are some spaces that demands future improvements. Some of the future recommendations are:

1. Increasing the dataset to train the models, by doing so we can make the model more confident on the wider range of the data to correctly classify the types of tumors from an MRI Scan.
2. This is just a web app right now, which can be in future upgraded to a system that can directly integrate with hospital's EHR and PACS system.
3. Addition of a heat map would address the lack of transparency in the current black box system.
4. Add Other Models for the left-out space not only focusing on MRI Scan but other topics like Pneumonia, Breast Cancer, Skin Cancer.

References

[1] B. H. Richter, "Challenges in the Interpretation of MRI Examinations Without Radiographic Correlation: Pearls and Pitfalls to Avoid," [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8364739/.

[2] J. Chai, Z. Hao and A. Li, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666827021000670.

[3] "The number of brain tumor patients is increasing in Nepal, high risk in women, these are the symptoms," [Online]. Available: https://ekantipur.com/lifestyle/2025/06/11/en/the-number-of-brain-tumor-patients-is-increasing-in-nepal-high-risk-in-women-these-are-the-symptoms-19-08.html.

[4] Z. N. K. Swati, Q. Zhao, Z. Ali and M. Kabir, "Brain tumor classification for MR images using transfer learning and fine-tuning," 2018. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0895611118305937.

[5] S. akila, "AUTOMATED DETECTION OF BRAIN TUMOR USING DEEP LEARNING AND MAGNETIC RESONANCE IMAGING (MRI) FOR CLASSIFICATION," [Online]. Available: https://docs.neu.edu.tr/library/7031521744.pdf.

[6] P. Shrestha and B. Pant, "Management of brain tumor in Nepal," 2004. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0531513103014055.

[7] R. Vankdothu, M. A. Hameed and H. Fatima, "A Brain Tumor Identification and Classification Using Deep Learning based on CNN-LSTM Method," [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790622002361.

[8] S. Tummala, S. Kadry, S. A. C. Bukhari and H. T. Rauf, "Classification of Brain Tumor from Magnetic Resonance Imaging Using Vision Transformers Ensembling," [Online]. Available: https://www.mdpi.com/1718-7729/29/10/590.

[9] M. Usman, T. Tariq and Z. Ali, "Analyzing Transfer Learning of Vision Transformers for Interpreting Chest Radiography," [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9274969/.

[10] S. Pereira, A. Pinto and V. Alves, "Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images," 4 March 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7426413.

[11] S. Singh, M. Kumar, A. Kumar, B. K. Verma, K. Abhishek and S. Selvarajan, "Efficient pneumonia detection using Vision Transformers on chest X-rays," 2024. [Online]. Available: https://www.nature.com/articles/s41598-024-52703-2.

[12] N. Chakrabarty and S. Kanchan, "Brain Tumor Classification (MRI)," [Online]. Available: https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri.

[13] K. H. Ali, "ViT-BT: Improving MRI Brain Tumor Classification With Vision Transformer with Transfer Learning," [Online]. Available: https://www.ijsce.org/wp-content/uploads/papers/v14i4/D364414040924.pdf.

[14] S. Ruder, "An overview of gradient descent optimization algorithms," [Online]. Available: http://arxiv.org/abs/1609.04747.

[15] A. Mao, M. Mohri and Y. Zhong, "Cross-Entropy Loss Functions:Theoretical Analysis and Applications," [Online]. Available: https://proceedings.mlr.press/v202/mao23b/mao23b.pdf.

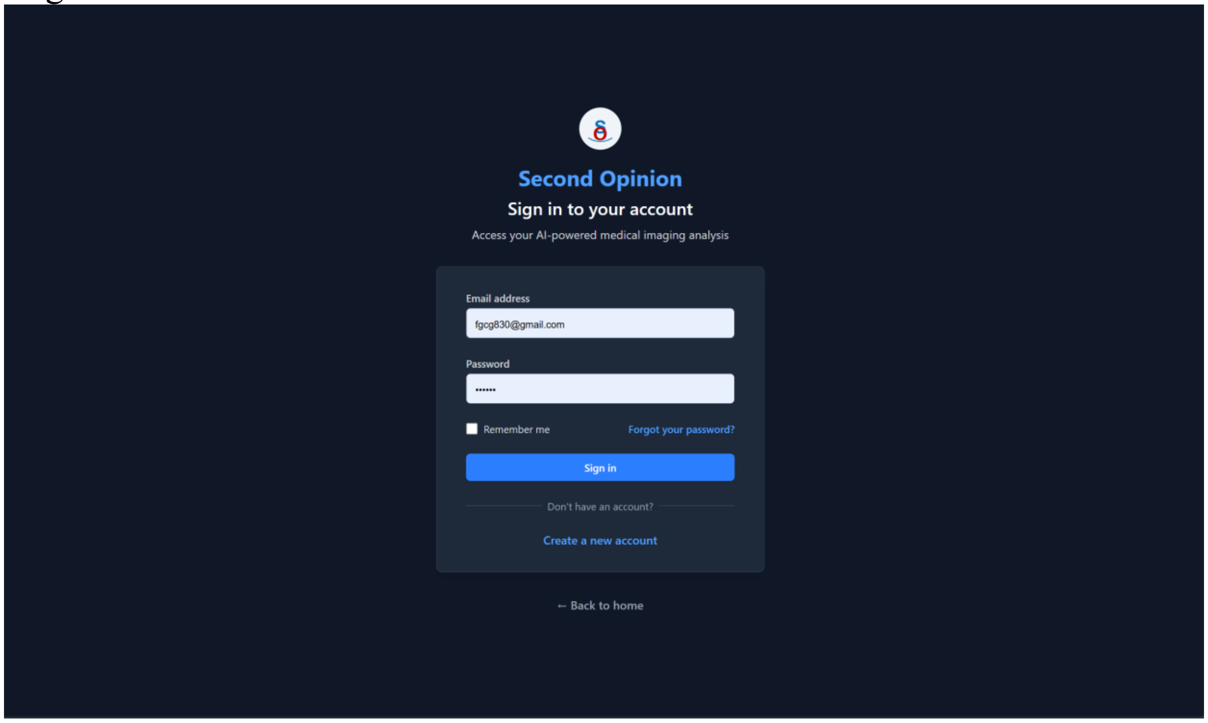
[16] R. J. Piper, S. Mikhael, J. M. Wardlaw, D. H. Laidlaw, I. R. Whittle and M. E. Bastin, "Imaging signatures of meningioma and low-grade glioma: a diffusion tensor, magnetization transfer and quantitative longitudinal relaxation time MRI study," [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4801649/.

[17] S. Sarker, "Transfer Learning and Explainable AI for Brain Tumor Classification: A Study Using MRI Data from Bangladesh," 8 June 2025. [Online]. Available: https://arxiv.org/abs/2506.07228.

APPENDICES

SCREENSHOTS

1. Login Form



Second Opinion

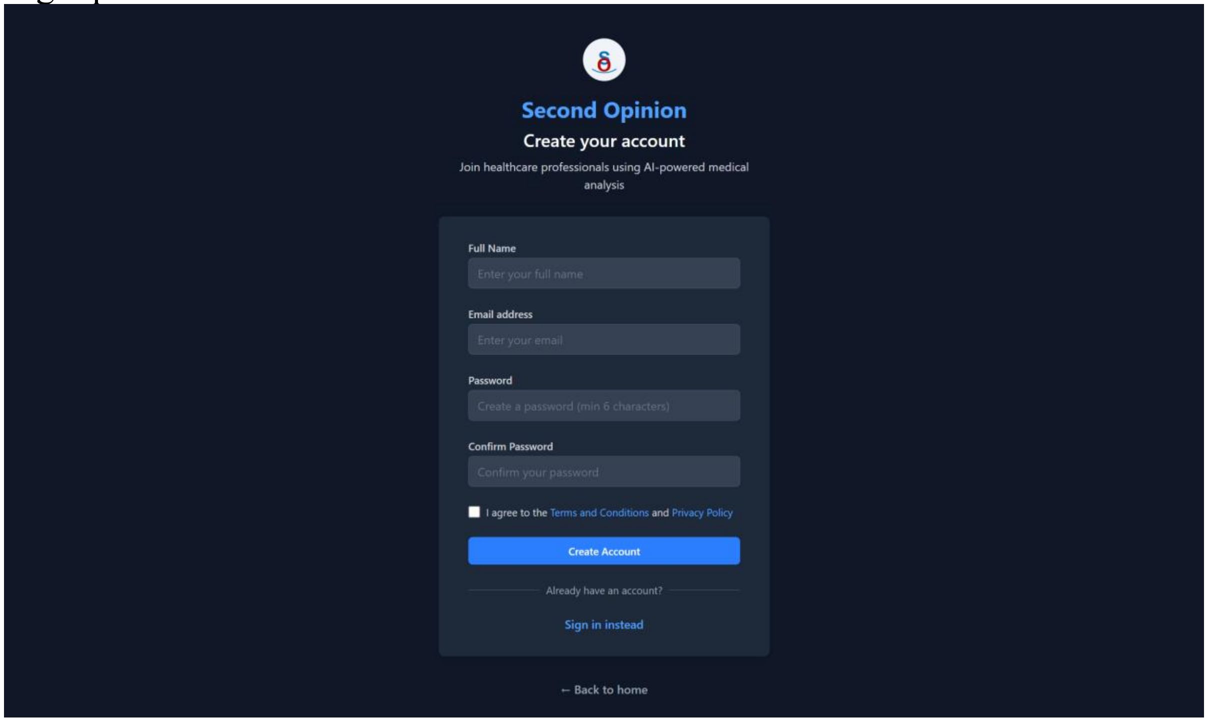
Sign in to your account

Access your AI-powered medical imaging analysis

Email address: fgcg830@gmail.com

Password: ••••••

2. Signup Form



Second Opinion

Create your account

Join healthcare professionals using AI-powered medical analysis

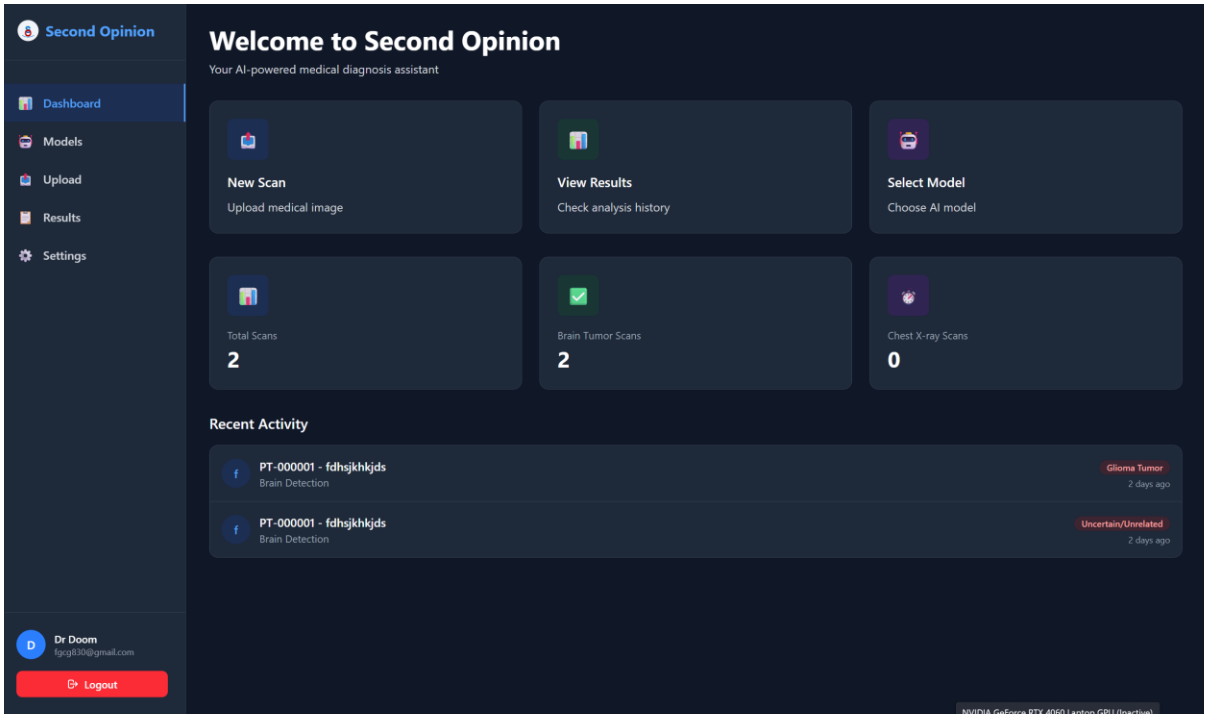
Full Name: Enter your full name

Email address: Enter your email

Password: Create a password (min 6 characters)

Confirm Password: Confirm your password

3. Dashboard



Second Opinion

Dashboard
Models
Upload
Results
Settings

Welcome to Second Opinion

Your AI-powered medical diagnosis assistant

New Scan
Upload medical image

View Results
Check analysis history

Select Model
Choose AI model

Total Scans
2

Brain Tumor Scans
2

Chest X-ray Scans
0

Recent Activity

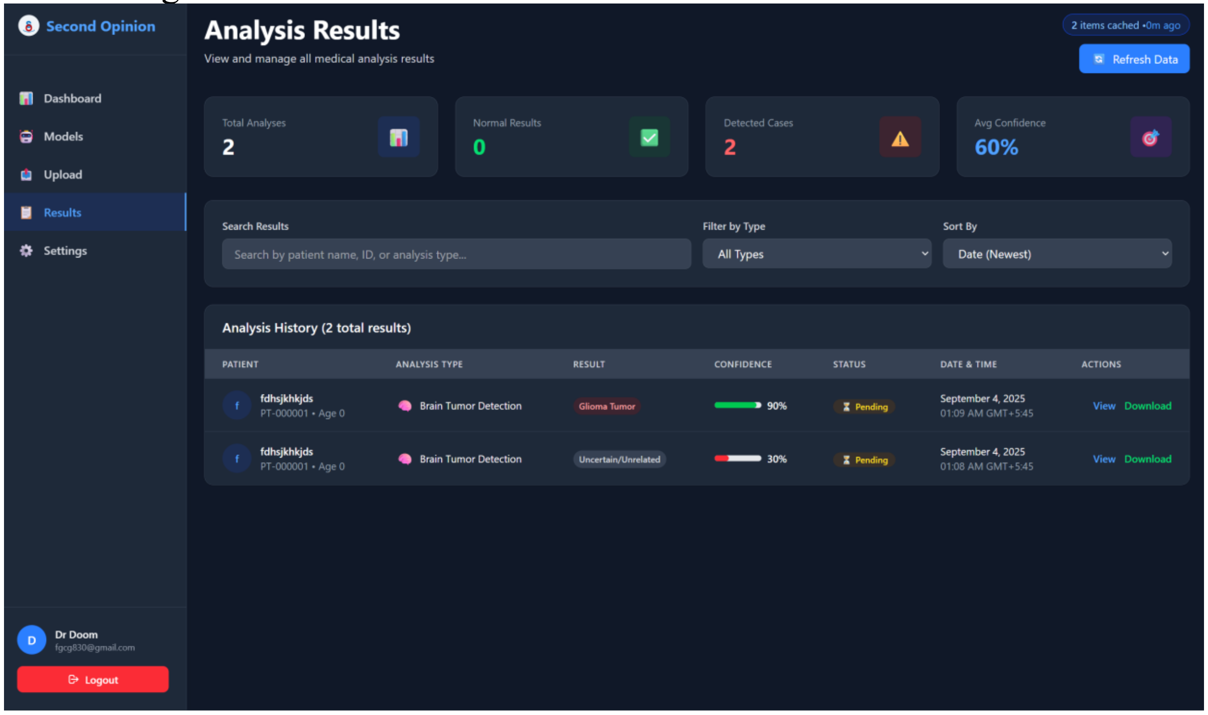
PT-000001 - fdhsjkhkjds
Brain Detection
Glioma Tumor
2 days ago

PT-000001 - fdhsjkhkjds
Brain Detection
Uncertain/Unrelated
2 days ago

Dr Doom
fgcg830@gmail.com
Logout

NVIDIA GeForce RTX 4060 Laptop GPU (Inactive)

4. Results Page



Second Opinion

Dashboard
Models
Upload
Results
Settings

Analysis Results

View and manage all medical analysis results

2 items cached •0m ago
Refresh Data

Total Analyses
2

Normal Results
0

Detected Cases
2

Avg Confidence
60%

Search Results
Search by patient name, ID, or analysis type...

Filter by Type
All Types

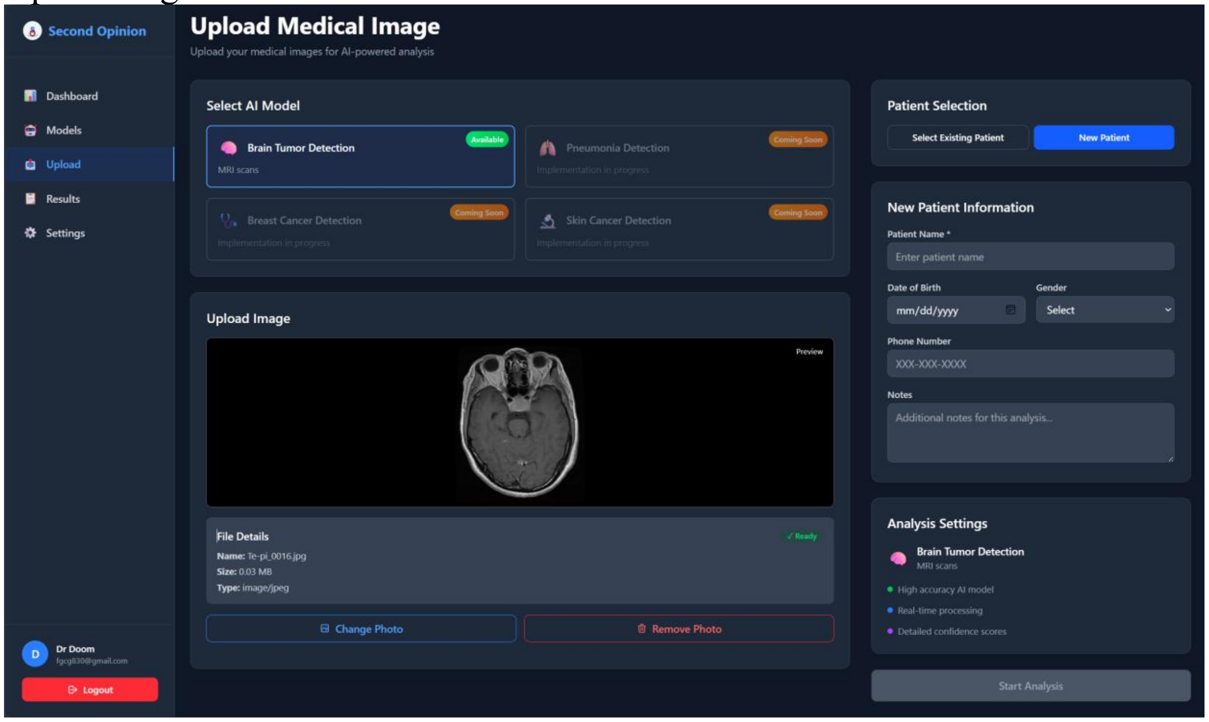
Sort By
Date (Newest)

Analysis History (2 total results)

PATIENT	ANALYSIS TYPE	RESULT	CONFIDENCE	STATUS	DATE & TIME	ACTIONS
fdhsjkhkjds PT-000001 • Age 0	Brain Tumor Detection	Glioma Tumor	90%	⏳ Pending	September 4, 2025 01:09 AM GMT+5:45	View Download
fdhsjkhkjds PT-000001 • Age 0	Brain Tumor Detection	Uncertain/Unrelated	30%	⏳ Pending	September 4, 2025 01:08 AM GMT+5:45	View Download

Dr Doom
fgcg830@gmail.com
Logout

5. Upload Page



Upload Medical Image

Upload your medical images for AI-powered analysis

Select AI Model

- Brain Tumor Detection — Available — MRI scans
- Pneumonia Detection — Coming Soon — Implementation in progress
- Breast Cancer Detection — Coming Soon — Implementation in progress
- Skin Cancer Detection — Coming Soon — Implementation in progress

Upload Image

Preview

File Details ✓ Ready

Name: Te-pi_0016.jpg
Size: 0.03 MB
Type: image/jpeg

Change Photo | Remove Photo

Patient Selection

Select Existing Patient | New Patient

New Patient Information

Patient Name *
Enter patient name

Date of Birth: mm/dd/yyyy
Gender: Select

Phone Number
XXX-XXX-XXXX

Notes
Additional notes for this analysis...

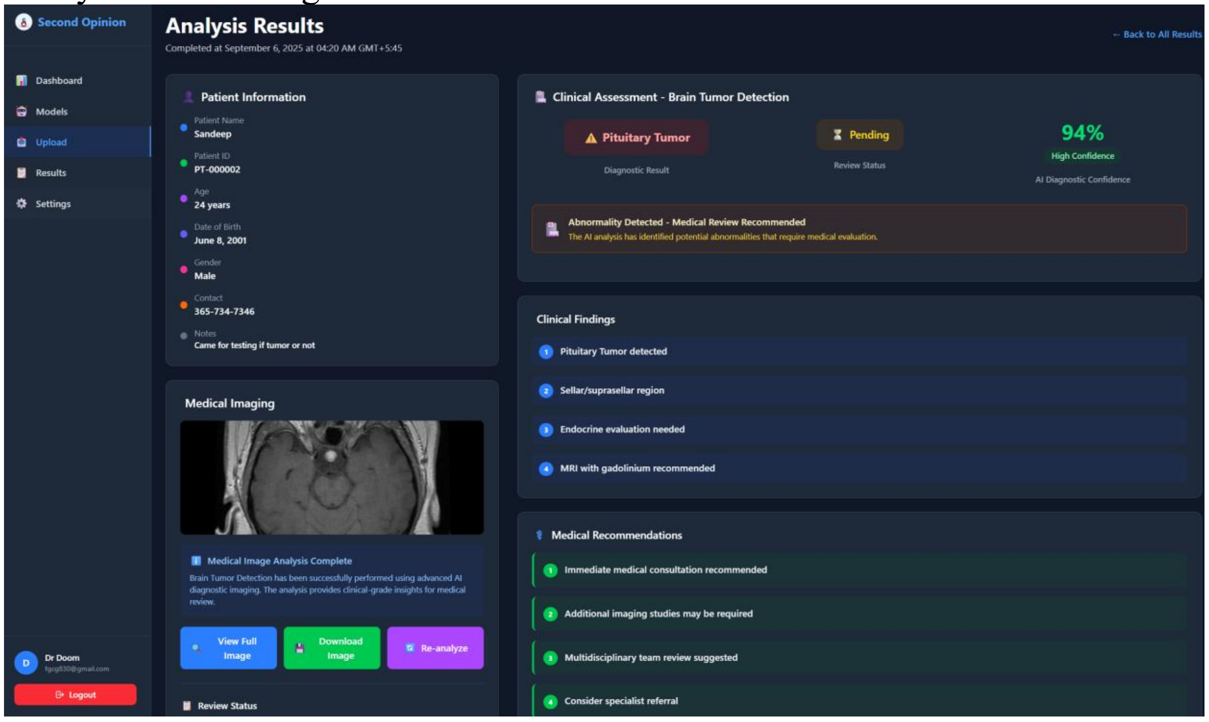
Analysis Settings

Brain Tumor Detection
MRI scans

- High accuracy AI model
- Real-time processing
- Detailed confidence scores

Start Analysis

6. Analysis Results Page



Analysis Results

Completed at September 6, 2025 at 04:20 AM GMT+5:45

Patient Information

Patient Name	Sandeep
Patient ID	PT-000002
Age	24 years
Date of Birth	June 8, 2001
Gender	Male
Contact	365-734-7346
Notes	Came for testing if tumor or not

Medical Imaging

Medical Image Analysis Complete

Brain Tumor Detection has been successfully performed using advanced AI diagnostic imaging. The analysis provides clinical-grade insights for medical review.

View Full Image | Download Image | Re-analyze

Review Status

Clinical Assessment - Brain Tumor Detection

Diagnostic Result	Review Status	AI Diagnostic Confidence
Pituitary Tumor	Pending	94% High Confidence

Abnormality Detected - Medical Review Recommended
The AI analysis has identified potential abnormalities that require medical evaluation.

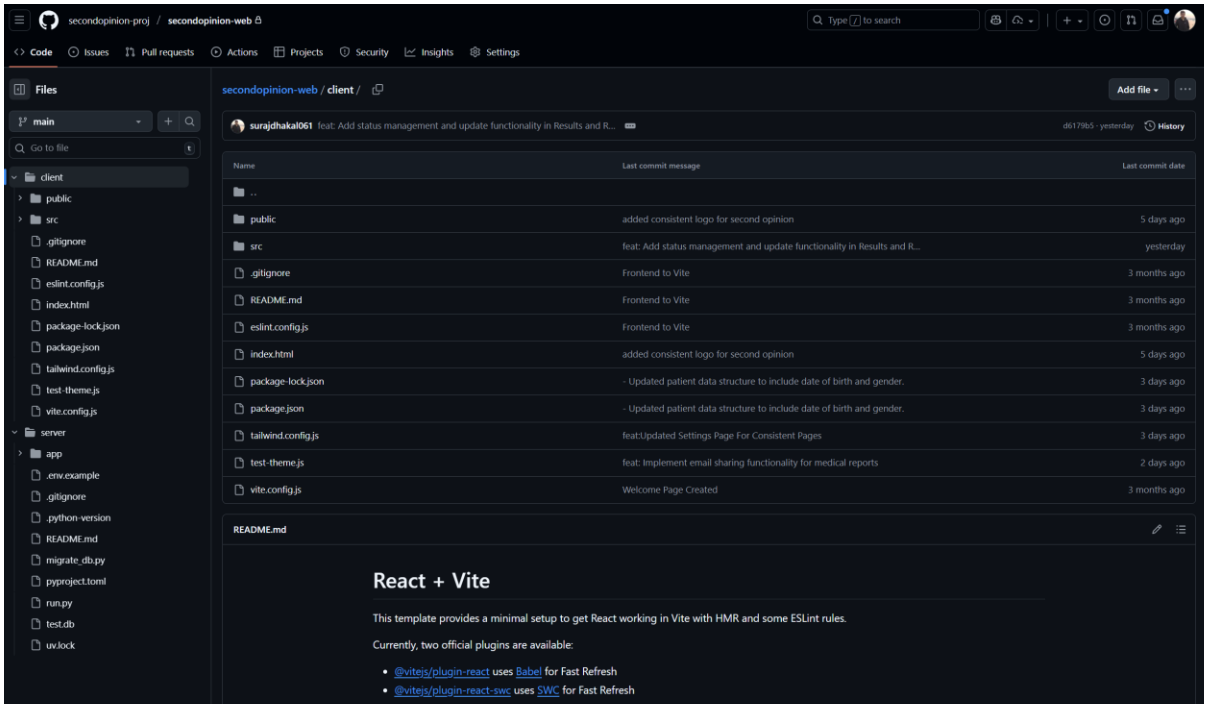
Clinical Findings

1. Pituitary Tumor detected
2. Sellar/suprasellar region
3. Endocrine evaluation needed
4. MRI with gadolinium recommended

Medical Recommendations

1. Immediate medical consultation recommended
2. Additional imaging studies may be required
3. Multidisciplinary team review suggested
4. Consider specialist referral

7. Git Repository Screenshot



secondopinion-proj / secondopinion-web

secondopinion-web / client /

surajdhakal061 feat: Add status management and update functionality in Results and R... d6179b5 · yesterday History

Name	Last commit message	Last commit date
..		
public	added consistent logo for second opinion	5 days ago
src	feat: Add status management and update functionality in Results and R...	yesterday
.gitignore	Frontend to Vite	3 months ago
README.md	Frontend to Vite	3 months ago
eslint.config.js	Frontend to Vite	3 months ago
index.html	added consistent logo for second opinion	5 days ago
package-lock.json	- Updated patient data structure to include date of birth and gender.	3 days ago
package.json	- Updated patient data structure to include date of birth and gender.	3 days ago
tailwind.config.js	feat:Updated Settings Page For Consistent Pages	3 days ago
test-theme.js	feat: Implement email sharing functionality for medical reports	2 days ago
vite.config.js	Welcome Page Created	3 months ago

```
client
  public
  src
  .gitignore
  README.md
  eslint.config.js
  index.html
  package-lock.json
  package.json
  tailwind.config.js
  test-theme.js
  vite.config.js
server
  app
  .env.example
  .gitignore
  .python-version
  README.md
  migrate_db.py
  pyproject.toml
  run.py
  test.db
  uv.lock
```

README.md

React + Vite

This template provides a minimal setup to get React working in Vite with HMR and some ESLint rules.

Currently, two official plugins are available:

- @vitejs/plugin-react uses Babel for Fast Refresh
- @vitejs/plugin-react-swc uses SWC for Fast Refresh

IMPORTANT CODE SNIPPETS

1. **Preparing ViT Model and Transforms for Edit**

```
def create_vit_model(num_classes: int, seed: int = 43):
    """Creates a ViT-B/16 feature extractor model and
transforms.

    Args:
        num_classes (int, optional): number of target
classes. Defaults to 3.
        seed (int, optional): random seed value for output
layer. Defaults to 42.

    Returns:
        model (torch.nn.Module): ViT-B/16 feature extractor
model.
        transforms (torchvision.transforms): ViT-B/16 image
transforms.
    """
    # Create ViT_B_16 pretrained weights, transforms and
model
    weights = torchvision.models.ViT_B_16_Weights.DEFAULT
    vit_transforms = weights.transforms()
    vit_transforms_augmented = transforms.Compose([
        transforms.RandomRotation(degrees=15),
        transforms.RandomHorizontalFlip(p=0.25),
        transforms.RandomVerticalFlip(p=0.25),
        transforms.ColorJitter(brightness=0.2,
contrast=0.2),
        vit_transforms
    ])
    model = torchvision.models.vit_b_16(weights=weights)

    # Freeze all layers in model
    for param in model.parameters():
        param.requires_grad = False

    # Unfreeze the last block of the transformer encoder
    for param in model.encoder.layers[-1].parameters():
        param.requires_grad = True

    # Unfreeze the final layer normalization layer (often
helps)
    for param in model.encoder.ln.parameters():
        param.requires_grad = True

    # Change classifier head to suit our needs (this will be
trainable)
    torch.manual_seed(seed)
    model.heads = nn.Sequential(
        nn.Linear(
            in_features=768,  # keep this the same as original
model
            out_features=num_classes,
        )
    )  # update to reflect target number of classes

    return model, vit_transforms, vit_transforms_augmented
```

2. **Training Loop for Model:**

```
def train(
    model: torch.nn.Module,
    train_dataloader,
    test_dataloader,
    optimizer: torch.optim.Optimizer,
    loss_fn: torch.nn.Module,
    epochs: int,
    device: torch.device,
    patience: int = 17,
    min_delta: float = 0,
) -> Dict[str, List]:
    results   =   {"train_loss":   [],   "train_acc":   [],
"test_loss": [], "test_acc": []}

    # Early stopping initializations
    epochs_no_improve = 0
    best_test_loss = float("inf")
    _BEST_MODEL_PATH = "Models/best_model.pth"  # [NEW] where
to save the best weights

    # Loop through training and testing steps for a number of
epochs
    for epoch in tqdm(range(epochs)):
        train_loss, train_acc = train_step(
            model=model,
            dataloader=train_dataloader,
            loss_fn=loss_fn,
            optimizer=optimizer,
            device=device,
        )
        test_loss, test_acc = test_step(
            model=model,         dataloader=test_dataloader,
loss_fn=loss_fn, device=device
        )

        # Print out what's happening
        print(
            f"Epoch: {epoch+1} | "
            f"train_loss: {train_loss:.4f} | "
            f"train_acc: {train_acc:.4f} | "
            f"test_loss: {test_loss:.4f} | "
            f"test_acc: {test_acc:.4f}"
        )
        # Update results dictionary
        results["train_loss"].append(train_loss)
        results["train_acc"].append(train_acc)
        results["test_loss"].append(test_loss)
        results["test_acc"].append(test_acc)

        # Early stopping check
        if test_loss < best_test_loss - min_delta:
            best_test_loss = test_loss
            epochs_no_improve = 0
            torch.save(model.state_dict(), _BEST_MODEL_PATH)
# [NEW] save best so far
        else:
            epochs_no_improve += 1

        if epochs_no_improve >= patience:
            print(f"\nEarly    stopping    triggered    after
{epoch+1} epochs.")
            break

    # Return the filled results at the end of the epochs
    # save the model    return results``` def train(
    model: torch.nn.Module,
    train_dataloader,
    test_dataloader,
    optimizer: torch.optim.Optimizer,
    loss_fn: torch.nn.Module,
    epochs: int,
    device: torch.device,
    patience: int = 17,
    min_delta: float = 0,
) -> Dict[str, List]:
    results   =   {"train_loss":   [],   "train_acc":   [],
"test_loss": [], "test_acc": []}

    # Early stopping initializations
    epochs_no_improve = 0
    best_test_loss = float("inf")
    _BEST_MODEL_PATH = "Models/best_model.pth"  # [NEW] where
to save the best weights
```

```
    # Loop through training and testing steps for a number of
epochs
    for epoch in tqdm(range(epochs)):
        train_loss, train_acc = train_step(
            model=model,
            dataloader=train_dataloader,
            loss_fn=loss_fn,
            optimizer=optimizer,
            device=device,
        )
        test_loss, test_acc = test_step(
            model=model,         dataloader=test_dataloader,
loss_fn=loss_fn, device=device
        )

        # Print out what's happening
        print(
            f"Epoch: {epoch+1} | "
            f"train_loss: {train_loss:.4f} | "
            f"train_acc: {train_acc:.4f} | "
            f"test_loss: {test_loss:.4f} | "
            f"test_acc: {test_acc:.4f}"
        )

        # Update results dictionary
        results["train_loss"].append(train_loss)
        results["train_acc"].append(train_acc)
        results["test_loss"].append(test_loss)
        results["test_acc"].append(test_acc)

        # Early stopping check
        if test_loss < best_test_loss - min_delta:
            best_test_loss = test_loss
            epochs_no_improve = 0
            torch.save(model.state_dict(), _BEST_MODEL_PATH)
# [NEW] save best so far
        else:
            epochs_no_improve += 1

        if epochs_no_improve >= patience:
            print(f"\nEarly    stopping    triggered    after
{epoch+1} epochs.")
            break
    # Return the filled results at the end of the epochs
    # save the model
    return results
```

3. **Prediction making endpoint:**

```
@app.post("/tumor")
async def post_image_tumor(file: UploadFile = File(...)):
    # First, use the Separator model to check if the image is
MRI
    separator_class_names = load_labels(
        model_basename="Seperator",
        default_labels=[
            "MRI",
            "Non-MRI"
        ],
    )

    # Load the separator model
    separator_model,         separator_transforms          =
load_separator_model()

    # Process the uploaded image
    image_data = await file.read()
    img = Image.open(io.BytesIO(image_data))
    if img.mode != "RGB":
        img = img.convert("RGB")

    # First check with separator model
    if separator_transforms is not None:
        img_tensor = separator_transforms(img).unsqueeze(0)

        with torch.inference_mode():
            separator_raw_probs                               =
torch.softmax(separator_model(img_tensor), dim=1)
```

```
        # Optionally reorder probs if an index map exists
        separator_index_map   =   load_index_map("Seperator",
num_classes=len(separator_class_names))
        separator_pred_probs                                 =
reorder_probs(separator_raw_probs,  separator_index_map)  if
separator_index_map else separator_raw_probs

        # Get the predicted class
        separator_pred_labels_and_probs                      =
{separator_class_names[i]: float(separator_pred_probs[0][i])
for i in range(len(separator_class_names))}
        predicted_image_type                                 =
max(separator_pred_labels_and_probs,
key=separator_pred_labels_and_probs.get)
        separator_confidence                                 =
max(separator_pred_labels_and_probs.values())
        mri_probability                                      =
separator_pred_labels_and_probs.get("MRI", 0.0)

        #  If  the  image  is  not  classified  as  MRI  with
reasonable confidence, return early
        if predicted_image_type != "MRI" or mri_probability
< 0.6:
            return {
                "error":  "Invalid  image  type  for  tumor
analysis",
                "message":   f"This   appears   to   be   a
{predicted_image_type}     image     (MRI     probability:
{mri_probability:.2f}). Tumor analysis requires MRI images.",
                "separator_prediction":
predicted_image_type,
                "separator_confidence":
float(separator_confidence),
                "mri_probability": float(mri_probability),
                "separator_probabilities":
separator_pred_labels_and_probs
            }
```

```
    # If we reach here, the image is classified as MRI,
proceed with tumor analysis
    # Load labels from file if present; otherwise use expected
default ordering
    class_names = load_labels(
        model_basename="Tumor",
        default_labels=[
            "Glioma Tumor",
            "Meningioma Tumor",
            "Normal Brain",
            "Pituitary Tumor",
        ],
    )

    # Load the cached tumor model
    vit, vit_transforms = load_tumor_model()

    # Transform and predict with tumor model
    if vit_transforms is not None:
        img_tensor = vit_transforms(img).unsqueeze(0)

        with torch.inference_mode():
            # Pass the transformed image through the model
and turn the prediction logits into prediction probabilities
            raw_probs    =    torch.softmax(vit(img_tensor),
dim=1)

        # Optionally reorder probs if an index map exists (to
match desired label order)
        index_map          =          load_index_map("Tumor",
num_classes=len(class_names))
        pred_probs = reorder_probs(raw_probs, index_map) if
index_map else raw_probs
```

```
        tumor_result = validate_image_confidence(pred_probs,
class_names, image_type="tumor")

        # Add separator information to the result
        tumor_result["separator_prediction"]                =
predicted_image_type
        tumor_result["separator_confidence"]                =
float(separator_confidence)
        tumor_result["mri_probability"]                     =
float(mri_probability)
        tumor_result["separator_probabilities"]             =
separator_pred_labels_and_probs

        return tumor_result
    else:
        return {"error": "Tumor model transforms not loaded"}
```