Brain Tumor Diagnosis System Using Vision Transformers

Capstone Project Presentation: Final Requirement for a Master of Science in Data Science

Andy Achouche, April, 2025.

Objective

- Use a Vision Transformer model to diagnose and classify brain tumors
- Design a web-based application to host the model
- Deploy the application on AWS (Amazon Web Services)
- Secure data in transit between the user and the server

Introduction

- Today’s clinical workflows rely heavily on expert radiologists for MRI interpretation, which can be time-consuming and subjective.
- This project introduces an automated system using Vision Transformers to consistently detect and classify brain tumors from MRI scans.
- The solution integrates a state-of-the-art deep-learning model with a user-friendly web interface for seamless image upload and real-time results.
- Fully deployed on AWS with HTTPS/TLS encryption, ensuring secure, scalable access anywhere.

CNN vs. Vision Transformers

- **CNNs:** use convolutional filters for hierarchical feature extraction (local focus)
    - Strengths: efficient parameter sharing, proven performance on **smaller datasets**
    - Limitations: limited global context, potential overfitting on large images
- **Vision Transformers:** split images into patches and apply self-attention across patches (global context)
    - Strengths: capture long-range dependencies, scalable with **large datasets**
    - Limitations: data-hungry, higher computational cost

Data & Preprocessing

- **Dataset Source:** Brain Tumor MRI dataset by Masoud Nickparvar on Kaggle (2021)
- **Composition:** MRI images labeled as glioma, meningioma, pituitary tumor, or no tumor
- **Preprocessing Steps:** skull stripping and bias field correction to remove artifacts and normalize intensities
- **ViT Preparation:** resized images to 224×224 pixels, normalized pixel values, and split into 16×16 patches for transformer input

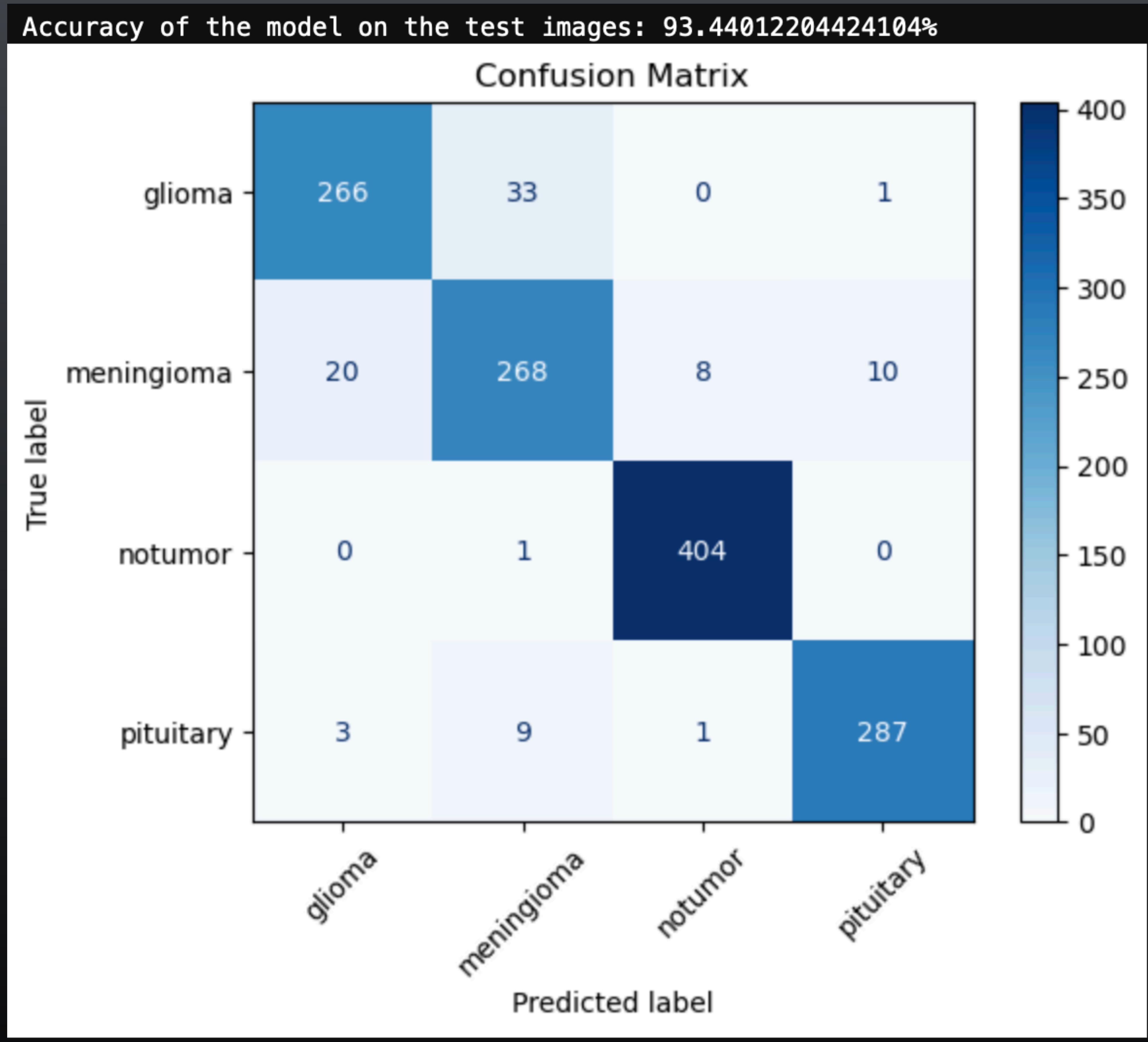
Model Training & Key Parameters

- **Model Capacity:** dim=512, depth=6, heads=8, mlp_dim=1024
- **Patch & Input:** patch_size=16, image_size=224
- **Regularization:** dropout=0.1, emb_dropout=0.1, weight_decay=1e-5
- **Optimization:** Adam optimizer with learning rate = 1e-4
- **Training Regimen:** batch_size=32, epochs=20, early stopping on validation loss

[6]:
```
# Training loop
epochs = 20
loss_values = []  # List to store loss values

for epoch in range(epochs):
    model.train()
    running_loss = 0.0
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()

    epoch_loss = running_loss / len(train_loader)
    loss_values.append(epoch_loss)  # Store the loss for each epoch

    print(f'Epoch {epoch+1}/{epochs}, Loss: {epoch_loss}')

# Save the trained model
torch.save(model.state_dict(), 'vit_mri_model.pth')
```

Model Performance

- **Accuracy:** 93.4% on the independent test set
- **Confusion Matrix:**
    - Glioma: 266 correct, 33 misclassified (11% error)
    - Meningioma: 268 correct, 38 misclassified (12% error)
    - No Tumor: 404 correct, 1 misclassified (0.2% error)
    - Pituitary: 287 correct, 13 misclassified (4% error)

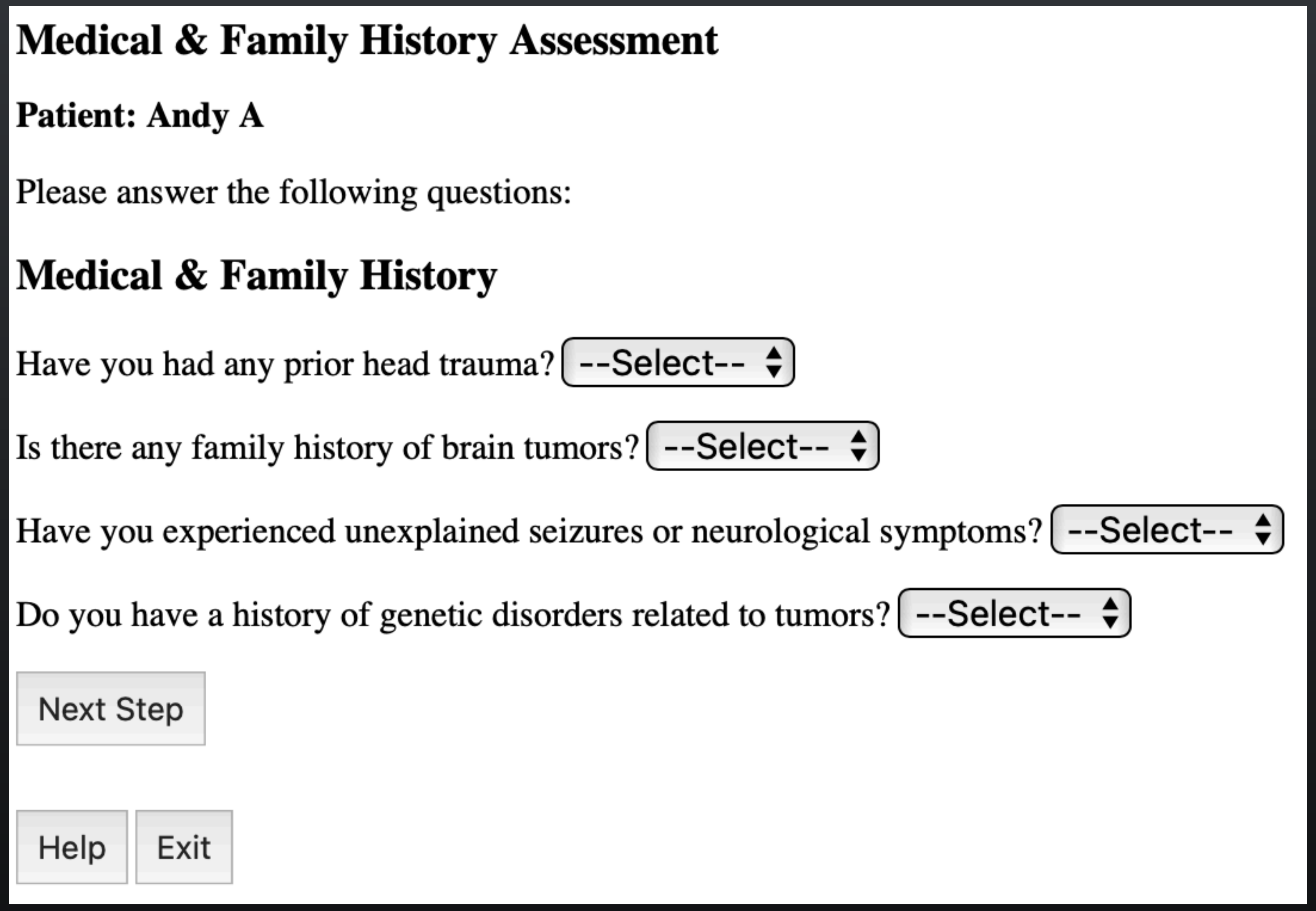


Web Application Design

- **Front-end:** React single-page interface for image uploads, patient navigation, and results display
- **Back-end:** Flask API routes handling authentication, patient records, test data collection, image inference, and report generation
- **UI Components:** doctor login, patient profile (search/add), medical history, combined tests, imaging instructions, upload, and results pages.

Flask App Architecture

- Built with Flask and SQLite for lightweight, session-based web services
- Modular route design: `/login`, `/patient_profile`, `/medical_history`, `/combined_tests`, `/imaging_studies`, `/upload_mri`, `/predict`, `/generate_report`

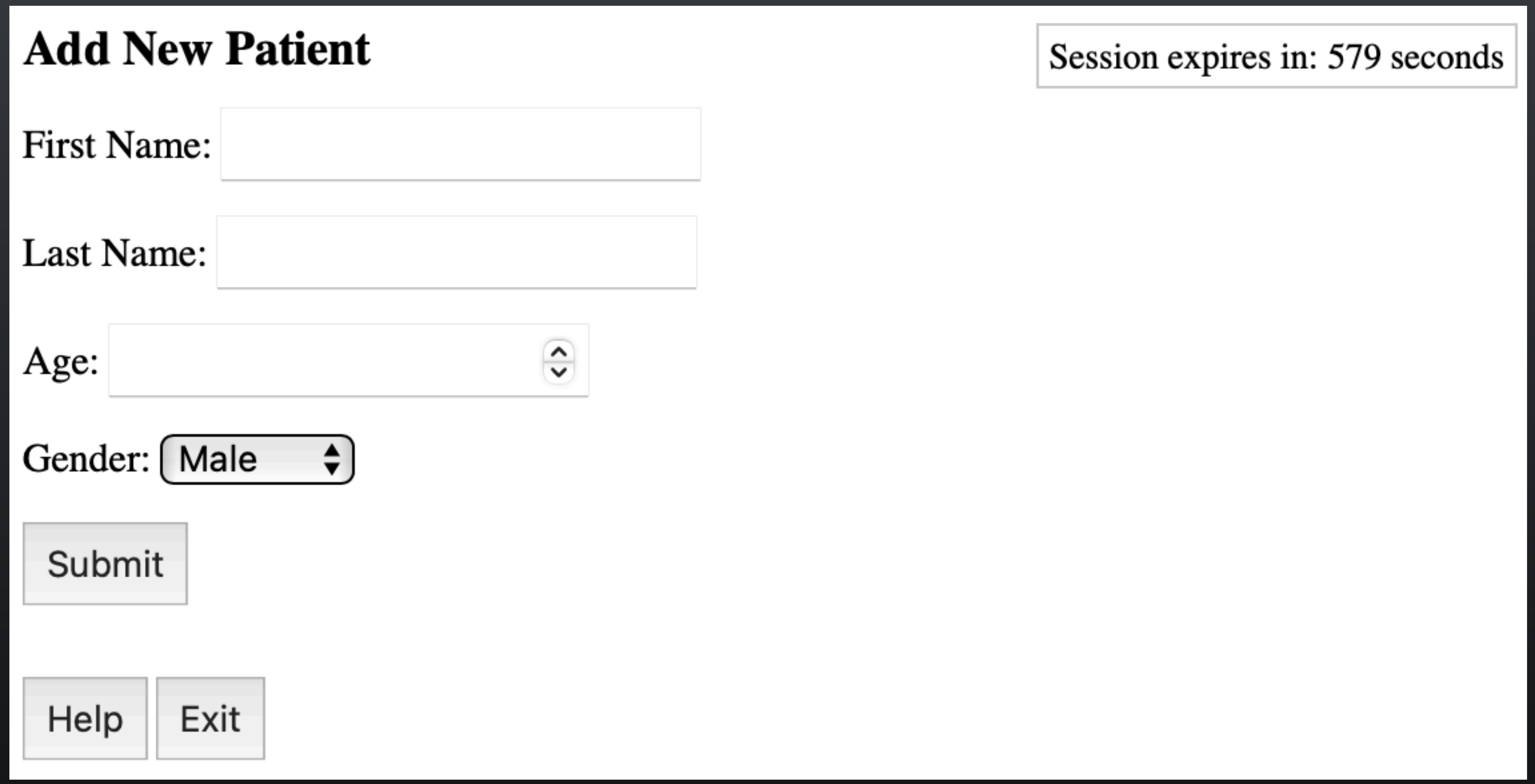


Authentication & Session Management

- `@app.before_request` guard redirects unauthenticated users to `/login`
- Doctor login via security code; sessions secured with `secret_key` and 10-minute inactivity timeout
- Inactivity timer resets on user events; automatic logout after timeout

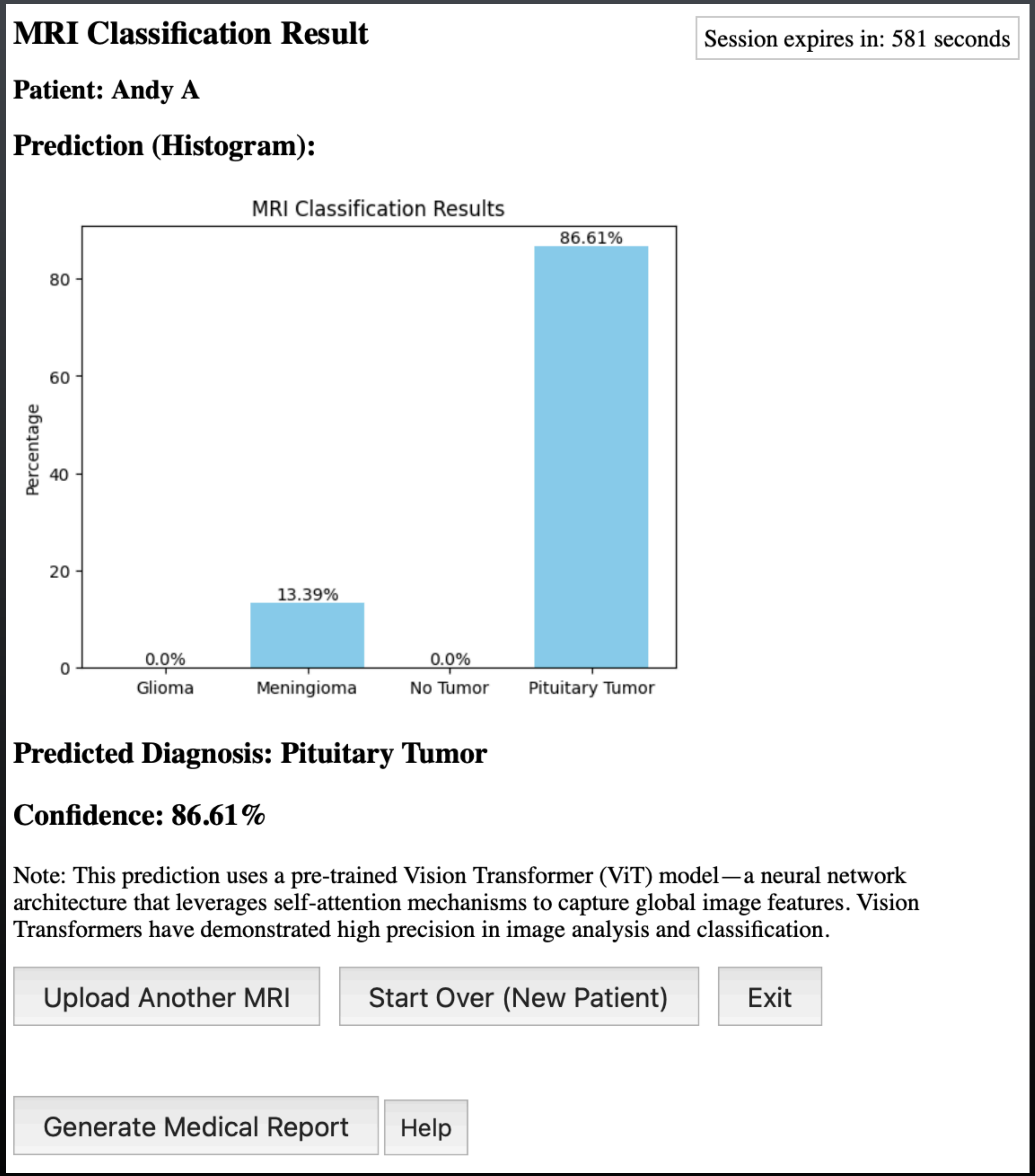
Patient Management Interface

- SQLite `patients.db` stores patient demographics; accessed via `get_db_connection()`
- Add new patient form captures name, age, gender; Search existing patient dropdown populates from the database
- Session stores selected patient for downstream questionnaire and imaging workflows



MRI Upload & Prediction

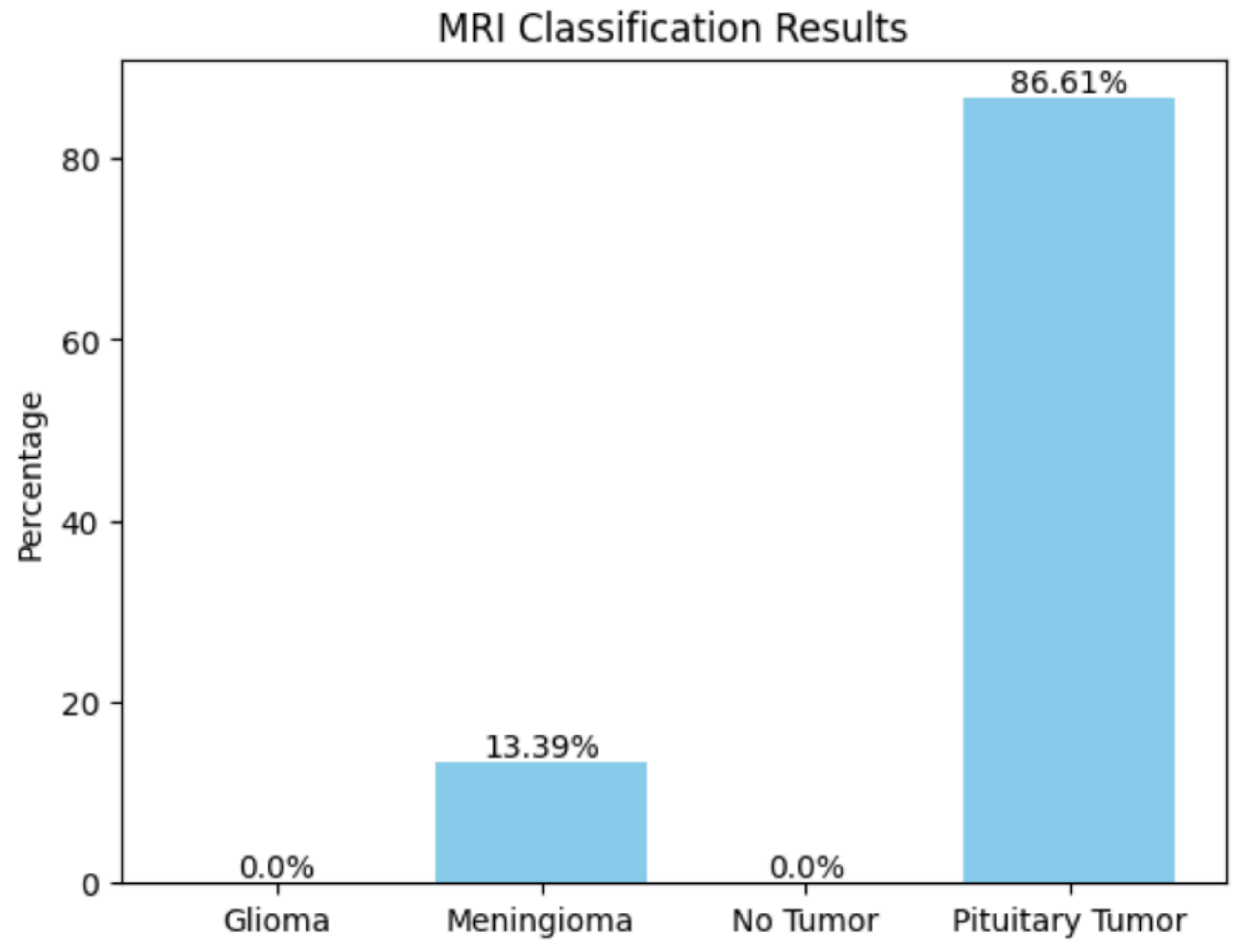
- `/upload_mri` route presents upload form; /predict handles file POST, runs ViT inference
- Preprocessing: resize, normalize; inference with model loaded in-memory
- Outputs: HTML page with uploaded image, histogram of class probabilities, predicted label, and confidence %



Report Generation Module

- `/generate_report` compiles patient info, history, test results, MRI findings, and tumor explanation
- Generates a histogram via Matplotlib and embeds it in an HTML report
- Response served with Content-Disposition header for user download as `medical_report.html`

MRI Classification Results



Predicted Diagnosis: Pituitary Tumor
Confidence: 86.61%

Tumor Explanation for Pituitary Tumor

Pituitary tumors occur in the pituitary gland and can affect hormone levels; further evaluation is recommended.

Note: This prediction uses a pre-trained Vision Transformer (ViT) model—a neural network architecture that leverages self-attention mechanisms to capture global image features. Vision Transformers have demonstrated high precision in image analysis and classification.

AWS Deployment & Security

- **Infrastructure:** Dockerized Flask app on EC2 instances behind an Application Load Balancer
- **Scaling:** Auto Scaling group configured for traffic spikes
- **Security:** HTTPS via AWS Certificate Manager; IAM roles least-privilege; security groups restrict access
- **Monitoring:** CloudWatch for logs, metrics, and alerts

Domain & HTTPS Configuration

- **Domain Purchase:** Registered **`brain-vit.com`** through GoDaddy
- **DNS Management:** Created a hosted zone in AWS Route 53, pointing to Load Balancer
- **SSL/TLS Certificate:** Provisioned via AWS Certificate Manager for brain-vit.com
- **Load Balancer Integration:** Attached certificate to ALB to enforce HTTPS
- **Secure Access:** Application now available at **`https://brain-vit.com`**, ensuring encrypted data in transit

Future Developments

- **Dataset Expansion:** Incorporate additional imaging modalities and larger, multi-center datasets
- **Edge Deployment:** Optimize model for on-device inference in clinical or mobile settings
- **Continuous Learning:** Implement feedback loops to retrain the model with new real-world cases
- **Data Security Enhancements:** Implement SQLCipher to encrypt the SQLite database at rest for robust application-layer protection
- **Clinical Validation:** Partner with hospitals for pilot studies and regulatory approval

Live Demonstration

Thank you!