B.M.S. COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

OOMD Mini Project Report

BRAIN TUMOUR DETECTION

Submitted in partial fulfilment for the award of degree of

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

ADITYA RAM S H(1BM22CS019)
AGNEYA D A (1BM22CS024)
ANAGHA BHARADWAJ (1BM22CS038)
ANJAN C (1BM22CS044)

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
2024-2025

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, Aditya Ram S H (1BM22CS019), Agneya D A (1BM22CS024), Anagha Bharadwaj (1BM22CS038), Anjan C (1BM22CS044) students of 5th Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this OOMD Mini Project entitled "BRAIN TUMOUR DETECTION" has been carried out in Department of CSE, B.M.S. College of Engineering, Bangalore during the academic semester Sep - Jan 2025. I also declare that to the best of our knowledge and belief, the OOMD mini Project report is not from part of any other report by any other students.

Signature of the Candidate

Aditya Ram S H (1BM22CS019)

Agneya D A (1BM22CS024)

Anagha Bharadwaj (1BM22CS038)

Anjan C (1BM22CS044)

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the OOMD Mini Project titled "**BRAIN TUMOUR DETECTION"** has been carried out by Aditya Ram S H (1BM22CS019), Agneya D A (1BM22CS024), Anagha Bharadwaj (1BM22CS038), Anjan C (1BM22CS044) during the academic year 2024-2025.

Signature of the Faculty in Charge

Table of Contents

CHAPTER 1: PROBLEM STATEMENT

Brain tumours are one of the most serious and potentially life-threatening conditions affecting individuals worldwide. Early detection plays a critical role in improving the chances of successful treatment and better patient outcomes. However, the traditional methods for detecting brain tumours, such as manual analysis of medical images (MRI scans), are prone to human error and require significant time and expertise. As a result, there is a growing need for automated systems that can assist medical professionals in detecting brain tumours accurately and quickly.

Manual detection of brain tumours from medical images can be time-consuming, error-prone, and dependent on the experience of the healthcare professional. Moreover, small or early-stage tumours may go unnoticed, leading to delayed diagnosis and treatment. Current methods lack the ability to offer consistent results across different cases and may struggle with analysing large datasets efficiently. The objective of this project is to develop an automated brain tumour detection application that leverages advanced image processing and machine learning techniques to accurately detect and classify brain tumours from medical images (such as MRI scans).

CHAPTER 2: SOFTWARE REQUIREMENT SPECIFICATION

Brain Tumour Detection

1. Introduction

1.1 Purpose of this Document

The purpose of this brain tumour detection application is to address the critical need for efficient and accurate diagnosis of brain tumours. Early detection of tumours significantly improves treatment outcomes and patient survival rates. Traditional methods rely heavily on manual inspection by medical experts, which can be time-consuming and prone to errors. By automating the detection process using advanced image processing and machine learning techniques, the application aims to provide consistent and reliable results. It serves as a decision-support tool for healthcare professionals, enhancing their diagnostic capabilities. The application also reduces the diagnostic workload, enabling radiologists to focus on more complex cases.

1.2 Scope of the Document

The scope of the brain tumour detection application includes processing medical images such as MRI or CT scans to identify and classify tumours. It is designed for use by healthcare professionals, including radiologists and neurologists, in clinical settings. The system supports both binary classification (tumour presence or absence) and further categorization into benign or malignant types. It includes a user-friendly interface for uploading images and viewing results with visual indications of the tumour's location. While primarily a diagnostic aid, it can also be integrated into larger hospital management systems for improved patient care. Future extensions may include advanced features like 3D tumour mapping and integration with telemedicine platforms.

1.3 Overview

This brain tumour detection application utilizes machine learning to enhance the accuracy and speed of tumour diagnosis. The system processes medical images and detects anomalies, highlighting areas of concern for medical professionals. It incorporates pre-trained models, such as convolutional neural networks (CNNs), to analyze image patterns indicative of tumours. The application outputs result in an easy-to-understand format, aiding doctors in making informed decisions. The project combines concepts of object-oriented modeling for practical implementation. It emphasizes reliability, efficiency, and usability, ensuring it aligns with the needs of the medical community. The application represents a step toward leveraging technology for better healthcare outcomes.

2. General Description

2.1 Product Perspective

The brain tumour detection application is a standalone software designed to assist medical professionals in diagnosing brain tumours accurately and efficiently. It uses advanced image processing and machine learning techniques to analyse MRI scans for abnormalities. The application operates independently but can integrate with existing hospital management systems if needed. It acts as a decision-support tool, complementing the expertise of radiologists and neurologists. By automating the detection process, it reduces the workload on medical professionals and minimizes the chances of oversight. The system is designed to be scalable, accommodating enhancements like multi-modality image analysis or real-time data processing in future versions.

2.2 Product Features

- Automated Detection: The system processes medical images to detect and highlight tumours with high accuracy.
- Classification: Tumours are categorized as benign or malignant based on image analysis.
- Visual Indicators: Detected tumour regions are marked on the image for easy identification.
- User-Friendly Interface: Simple and intuitive interface for uploading images and viewing results.
- Accuracy and Speed: Advanced algorithms ensure fast processing and reliable detection results.
- Data Security: Ensures patient data confidentiality through encryption and secure storage.

2.3 User Characteristics

- Primary Users: Radiologists, neurologists, and medical professionals involved in diagnosing brain tumours.
- Technical Proficiency: Users may have limited technical expertise; hence, the system is designed to be user-friendly and intuitive.
- Accessibility: Users may access the system in clinical settings or remotely through secured platforms.
- Medical Knowledge: Users have a background in medical imaging and diagnosis, enabling them to interpret system outputs effectively.
- Adaptability: Users should find the system adaptable to their workflow and integrate it seamlessly into existing diagnostic processes.
- Requirement for Accuracy: Users expect highly accurate and reliable results to assist in critical medical decisions.

2.4 System Features

- Image Upload and Preprocessing: Accepts MRI scans, pre-processes them for analysis, and ensures compatibility with multiple formats.
- Machine Learning Integration: Employs trained machine learning models to detect and classify tumours.
- Real-Time Results: Provides results promptly, allowing immediate access to diagnostic insights.
- Interactive Visualization: Offers zooming and other interactive tools to examine detected tumour regions closely.
- Scalability: Designed to accommodate additional features like multi-organ analysis or support for other imaging modalities.
- Error Logging and Feedback: Tracks errors and collects user feedback to improve future versions.

3. Functional Requirements

3.1 Image Upload and Input Validation

- The system must allow users to upload medical images (e.g., MRI scans) in standard formats such as TIF.
- The system must validate the uploaded image for compatibility and notify the user of any errors or unsupported formats.

3.2 Image Preprocessing

- The application must preprocess the input image to enhance quality, including noise reduction, normalization, and resizing for analysis.

3.3 Tumour Detection

- The system must analyse the uploaded medical image using machine learning models to detect abnormalities or tumour-like regions.

3.4 Tumour Classification

- The system must classify the detected tumour as either benign or malignant based on its features and characteristics.

3.5 Visualization and Highlighting

- The system must visually mark the detected tumour region(s) on the uploaded image, enabling easy identification by the user.
- The system must allow users to interact with the marked image (e.g., zoom, pan) for closer inspection.

3.6 Result Reporting

- The system must generate a detailed report containing diagnostic results, including tumour presence and affected regions.
- The report must be exportable in formats such as PDF for documentation or sharing with other professionals.

3.7 Data Management

- The system must store patient data and diagnostic results securely for future reference.
- It must provide options to retrieve, update, or delete stored records as needed.

3.8 Real-Time Processing

- The system must process the uploaded image and provide results within a reasonable time (e.g., less than 4 seconds per image).

3.9 Feedback and Error Reporting

- The system must include a feature for users to provide feedback on the results.
- Errors encountered during analysis must be logged and displayed with meaningful messages for troubleshooting.

3.10 Integration Support

- The system must support integration with hospital management systems to fetch and store patient data seamlessly.

3.11 Help and Support

- The application must include a help section with instructions and troubleshooting guidance.
- A contact support option must be available for users facing technical issues.

4. Interface Requirements

4.1 User Interface (UI)

- Provide a user-friendly and intuitive graphical interface with clear navigation, an upload section for medical images, and a results display area.
- Include interactive tools such as zoom and pan for detailed inspection of medical images.

4.2 Result Output Interface:

- Display diagnostic results with the tumour presence, classification, confidence score, and visually marked tumour regions.
- Allow users to download results and reports in standard formats like PDF.

4.3 Data Input and Validation Interface:

- Support secure uploading of medical images via file browser.
- Validate input files for format and size compatibility, providing real-time feedback on errors.

5. Performance Requirements

5.1 Processing Speed

- The system must process and analyse uploaded medical images within 30 seconds per image to ensure quick diagnostic results.

5.2 Accuracy and Reliability

- The application must achieve a minimum detection accuracy of 95% for identifying tumour presence and classification (benign or malignant).

5.3 Scalability

- The system must handle up to 100 concurrent users without a noticeable decline in performance or response times.

5.4 System Availability

- The application must maintain an uptime of at least 99.5%, ensuring consistent availability for users.

5.5 Data Handling

- The system must support processing of large image files without delays or crashes.

5.6 Resource Utilization

- The application must optimize CPU and memory usage, ensuring efficient performance on both high-end and mid-range devices or servers.

6. Design Constraints

- The system must be optimized to run on devices with limited hardware resources (e.g., 8GB RAM, mid-range processors).
- Development must adhere to budget and time constraints, balancing core functionality with performance.

7. Non-Functional Requirements

- **Usability:** The application must have an intuitive interface, allowing users to upload images and interpret results with minimal training.
- **Scalability:** The system must support simultaneous processing for up to 100 users without performance degradation.
- **Reliability:** The application must maintain an uptime of at least 99.5%, ensuring consistent availability.
- **Performance:** Image processing and tumour detection must be completed within 30 seconds per upload.

8. Preliminary Schedule and Budget

8.1 Schedule

Phase	Duration
Requirement Analysis	1 week
System Design	2 weeks
Development & Implementation	6 weeks
Testing & Deployment	2 weeks

8.2 Budget

Category	Estimated Cost
Personnel	$50,000
Development Tools & Infrastructure	$15,000
Testing & Miscellaneous	$5,000
Total Estimated Budget	$70,000

9. Conclusion

The brain tumour detection application aims to revolutionize the diagnostic process by leveraging advanced machine learning and image processing techniques to accurately detect and classify brain tumours. With its user-friendly interface and high accuracy, the system enhances the diagnostic capabilities of healthcare professionals, aiding in faster and more reliable decision-making. By automating tumour detection, the application reduces human error and workload, leading to improved patient care. The development of this tool represents a significant step toward integrating AI into healthcare, making medical diagnostics more efficient and accessible. Future enhancements can further improve its capabilities, broadening its impact on the medical field.

CHAPTER 3: CLASS MODELING

UML Advanced Class Diagram

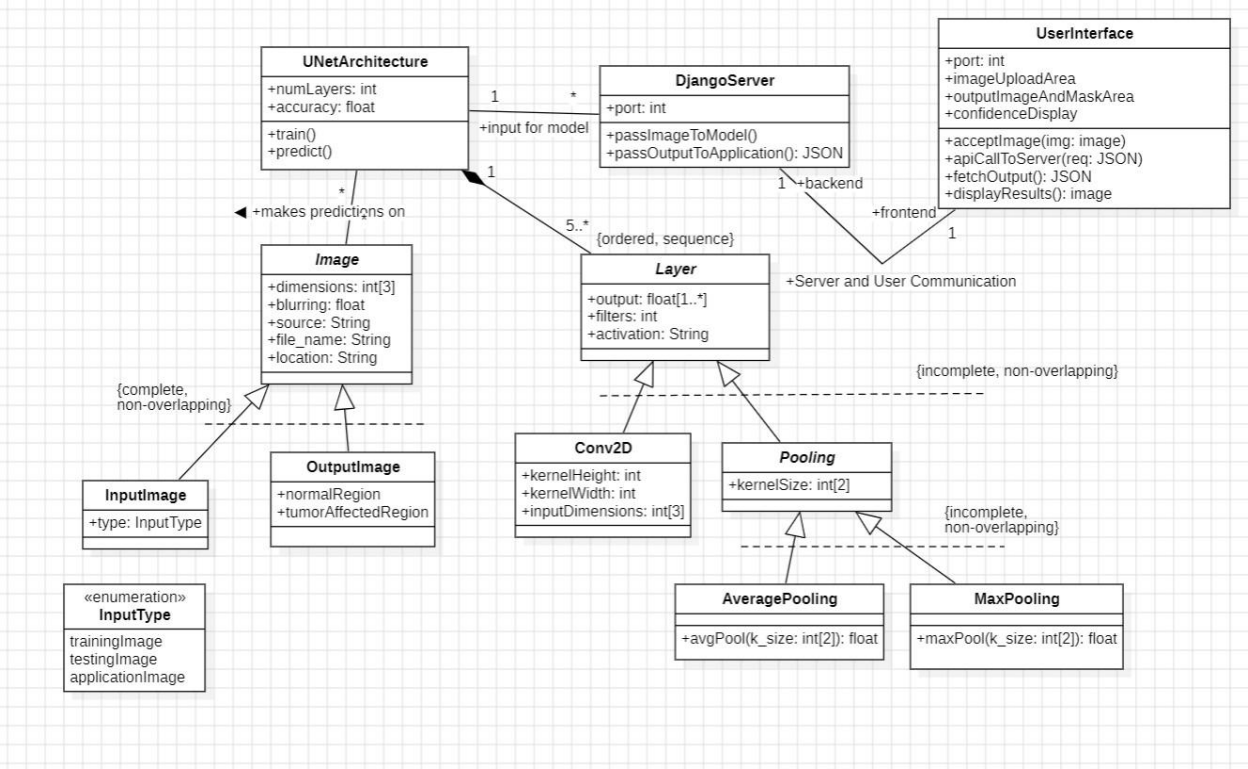

Figure 3.1: Class Diagram

This UML (Unified Modeling Language) class diagram showcases the architecture of a system involving image processing, using a deep learning model and server-client interaction.

1. UNetArchitecture

- Represents the core deep learning model.
- **Attributes:**
    - numLayers: Number of layers in the U-Net architecture.
    - accuracy: Model accuracy as a float.
- **Methods:**
    - train(): Method to train the model.

- predict(): Method to make predictions.

- It interacts with other components for predictions.

2. Image

- A generic class representing images used in the system.
- **Attributes:**
  - dimensions: Dimensions of the image as an array of three integers.
  - blurring: Level of blurring applied to the image.
  - source: Source of the image.
  - file_name: Name of the image file.
  - location: File path or location of the image.
- Specialized into two subclasses: InputImage and OutputImage.

3. InputImage

- A subclass of Image.
- **Attribute:**
  - type: Indicates the type of input (enumeration: trainingImage, testingImage, applicationImage).

4. OutputImage

- A subclass of Image.
- **Attributes:**
  - normalRegion: Segmented normal region of the image.
  - tumorAffectedRegion: Segmented region affected by a tumor.

5. DjangoServer

- Represents the server managing backend operations.
- **Attributes:**

- port: Server port number.

- **Methods:**
  - passImageToModel(): Sends input images to the model for predictions.
  - passOutputToApplication(): Returns output data in JSON format to the client.

6. UserInterface

- Represents the frontend interface of the system.
- **Attributes:**
  - port: Port for UI communication.
  - imageUploadArea: Area for users to upload images.
  - outputImageAndMaskArea: Display area for segmented outputs and masks.
  - confidenceDisplay: Displays prediction confidence.
- **Methods:**
  - acceptImage(img): Accepts an uploaded image.
  - apiCallToServer(req): Makes API calls to the server.
  - fetchOutput(): Fetches processed results from the server.
  - displayResults(): Displays the results (images and masks) to the user.

7. Layer

- Represents a neural network layer.
- **Attributes:**
  - output: An array of floats representing the layer's output.
  - filters: Number of filters in the layer.
  - activation: Activation function used (e.g., ReLU, sigmoid).
- Organized as a sequence, used by the U-Net model.

8. Conv2D

- A specific type of Layer for convolution operations.
- **Attributes:**
    - kernelHeight: Height of the convolutional kernel.
    - kernelWidth: Width of the convolutional kernel.
    - inputDimensions: Input image dimensions as an array.

9. Pooling

- Represents a generic pooling layer.
- **Attribute:**
    - kernelSize: Size of the pooling kernel.

Subclasses:

- **AveragePooling**: Computes average values in the pooling window.
    - Attribute: avgPool(k_size): Computes average pooling given kernel size.
- **MaxPooling**: Computes maximum values in the pooling window.
    - Attribute: maxPool(k_size): Computes max pooling given kernel size.

CHAPTER 4: STATE MODELING

UML Advanced State Diagram

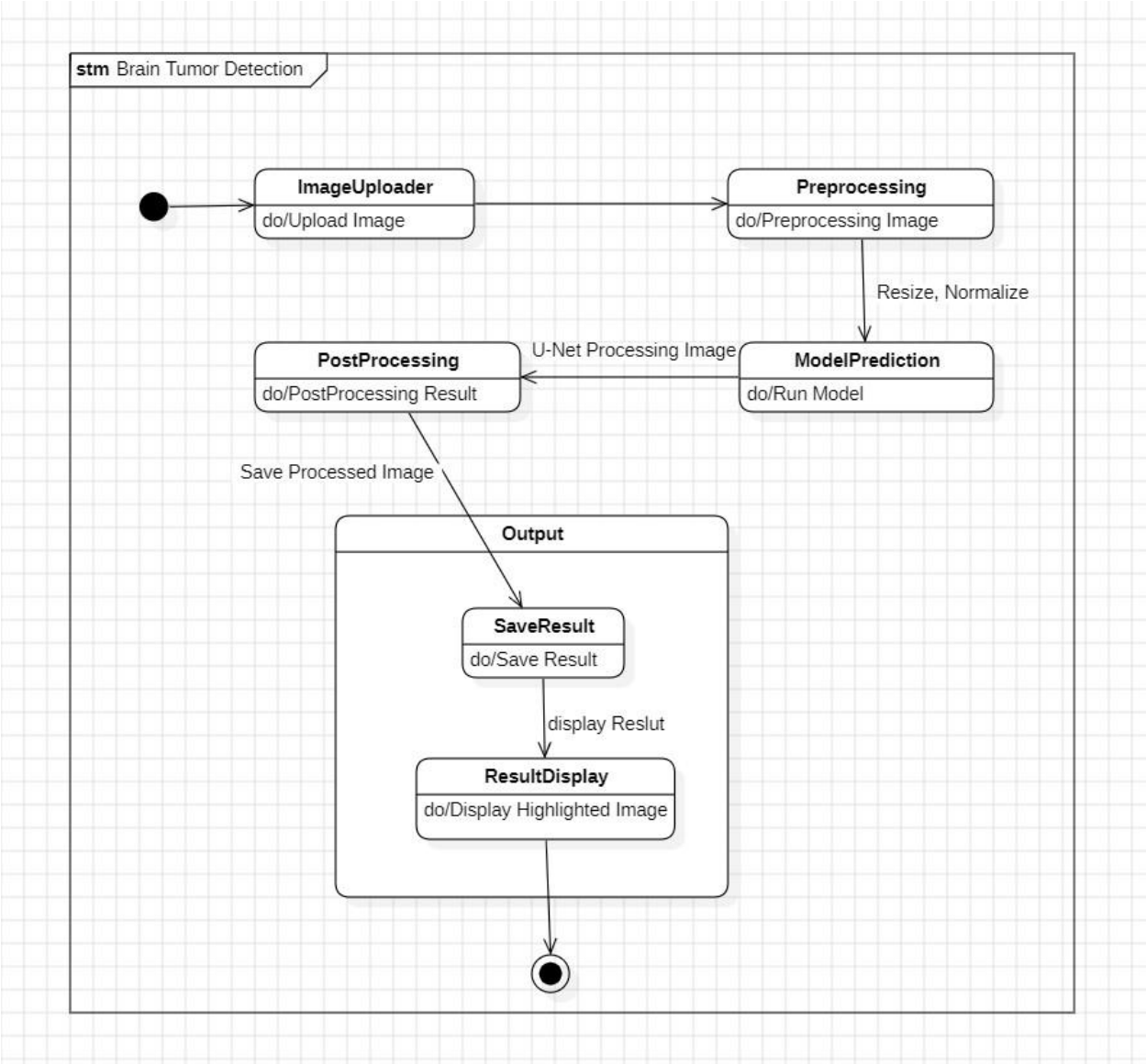

Figure 4.1: State Diagram

This state machine diagram illustrates the workflow of a Brain Tumor Detection System. It describes the sequence of states and events involved in processing a medical image from uploading the image to generating and visualizing the results. The relevance of each state and associated events is highlighted to show how they contribute to the overall functionality of the system.

Relevance of Each State

1. **ImageUploader**
    - This is the starting state of the system, allowing users to upload medical images for analysis. It ensures that the system receives the necessary input to begin the tumor detection process. Without this step, the workflow cannot proceed.
2. **Preprocessing**
    - Prepares the uploaded image for processing by the U-Net model. It performs essential adjustments like resizing and normalization, ensuring that the image adheres to the model's input requirements for optimal predictions.
3. **ModelPrediction**
    - The central state where the U-Net model processes the preprocessed image to detect brain tumors. This state is critical as it generates the primary outputs, such as segmented tumor regions and prediction confidence.
4. **PostProcessing**
    - Enhances the raw outputs from the model by refining segmentation masks or smoothing boundaries. This state improves the clarity and usability of results, ensuring better visual representation and accuracy.
5. **Output**
    - Serves as the final stage where results are saved and displayed. This state provides functionality to store processed results and present visual feedback to users, making it actionable for medical interpretation.
6. **End State**
    - Marks the successful completion of the process. It ensures that all required steps have been executed and results are accessible to the user.

Relevance of Events:

1. **do/Upload Image**
    - Initiates the workflow by allowing users to upload an image. This event is crucial for providing the system with the necessary input to start the analysis.
2. **do/Preprocessing Image**
    - Triggers the preprocessing tasks like resizing and normalization, which are fundamental for ensuring compatibility between the image and the model.
3. **do/Run Model**
    - Executes the U-Net model to perform segmentation and generate predictions. This event is the backbone of the system as it performs the primary tumor detection task.
4. **do/PostProcessing Result**
    - Launches the postprocessing steps to refine and optimize the model's raw output, enhancing its visual and practical utility.
5. **do/Save Result**
    - Saves the processed and refined output. This event is vital for archiving results for later review or further analysis.
6. **do/Display Highlighted Image**
    - Displays the final output with highlighted tumor regions. This event provides visual feedback to the user, making the results actionable and easy to interpret.

Each state in the system has a distinct and essential role in transforming raw medical images into actionable insights. The system ensures a seamless workflow, from image upload and preparation to advanced segmentation and visualization, aiding in the detection and analysis of brain tumors effectively.

CHAPTER 5: INTERACTION MODELING

UML Advanced Use Case Diagram

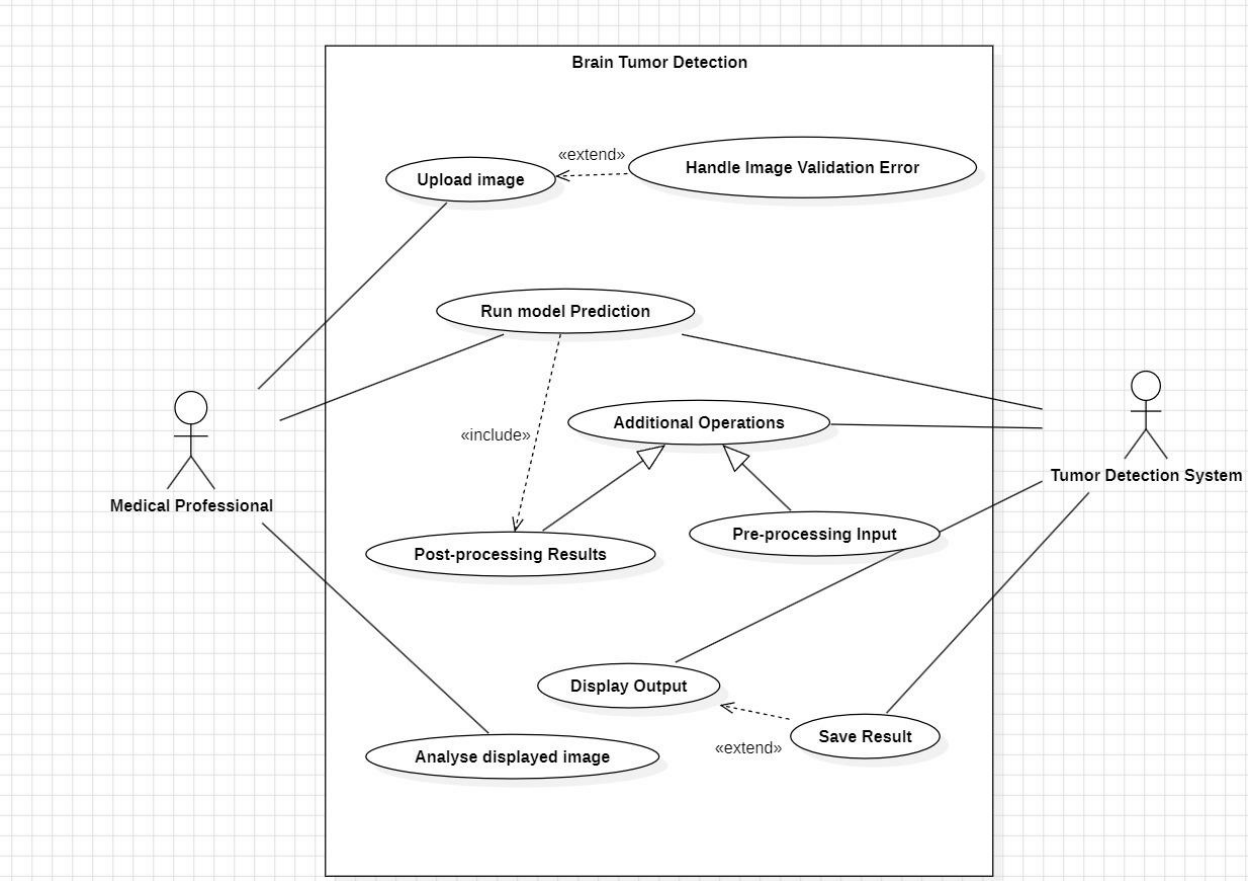

Figure 5.1: Use Case Diagram

The use case diagram represents the functionality of a Brain Tumor Detection System designed to assist medical professionals in analyzing medical images for tumor detection. It highlights the interactions between the medical professional and the tumor detection system through various use cases. These use cases cover essential tasks, including uploading images, running model predictions, and displaying results with highlighted tumor regions.

Relevance of Actors

1.Medical Professional

- Represents the primary user who interacts with the system. They upload medical images, interpret the results, and take actionable decisions based on the system's outputs. Their role ensures that the system outputs are utilized in a clinical or diagnostic context.

2.Tumor Detection System

- Acts as the automated backend entity performing image validation, tumor detection using machine learning models, and result visualization. It supports the medical professional by handling complex computational tasks with minimal manual intervention.

Relevance of Use Cases

Upload Image

- **Relevance**:
  This is the entry point for the entire process. The medical professional provides an image (e.g., MRI or CT scan) for analysis.
- **Relationship**:
  Extends to "Handle Image Validation Error" in cases where the image fails initial validation (e.g., corrupted or unsupported format).

Handle Image Validation Error

- **Relevance**:
  Ensures that only valid and processable images proceed in the pipeline. This is critical for avoiding errors later in the workflow.
- **Relationship**:
  Extends from "Upload Image" when validation fails.

 Run Model Prediction

- **Relevance**:

  This is the core functionality of the system, where the AI model analyzes the uploaded and preprocessed image to detect brain tumors.

- **Relationship**:

  Includes "Pre-processing Input" and "Additional Operations" to ensure that input data is properly prepared before predictions are made.

 Pre-processing Input

- **Relevance**:

  Refers to preparing the uploaded image for the AI model by resizing, normalizing, or enhancing its quality.

- **Relationship**:

  Included by "Run Model Prediction" because preprocessing is an essential step before running predictions.

 Additional Operations

- **Relevance**:

  Represents any extra steps to improve the prediction process, such as augmenting data or enhancing prediction confidence for borderline cases.

- **Relationship**:

  Included by "Run Model Prediction."

 Post-processing Results

- **Relevance**:

  Converts the raw outputs from the AI model (e.g., detection probabilities) into a user-friendly format, such as visual highlights or statistical summaries.

- **Relationship**:

  Follows "Run Model Prediction."

- **Display Output**
  - **Relevance**:
    Provides the medical professional with the results of the analysis in a comprehensible format. This could include visual annotations on the image or a detailed report.
  - **Relationship**:
    Extends to "Save Result," allowing the user to store results for later reference.
- **Save Result**
  - **Relevance**:
    Allows the analyzed results to be stored in a database or exported for documentation or further review.
  - **Relationship**:
    Extended from "Display Output" when results need to be preserved.
- **Analyze Displayed Image**
  - **Relevance**:
    Represents the medical professional's role in reviewing and interpreting the output from the system. This step ensures clinical validation and decision-making.
  - **Relationship**:
    Interacts with "Display Output" as the displayed results guide further analysis or diagnosis.

This use case diagram outlines the interactions between a medical professional and the tumor detection system, emphasizing the system's ability to support decision-making in medical imaging. By streamlining processes like prediction, post processing, and result visualization, the system enhances efficiency and accuracy in brain tumor detection workflows.

UML Advanced Sequence Diagram

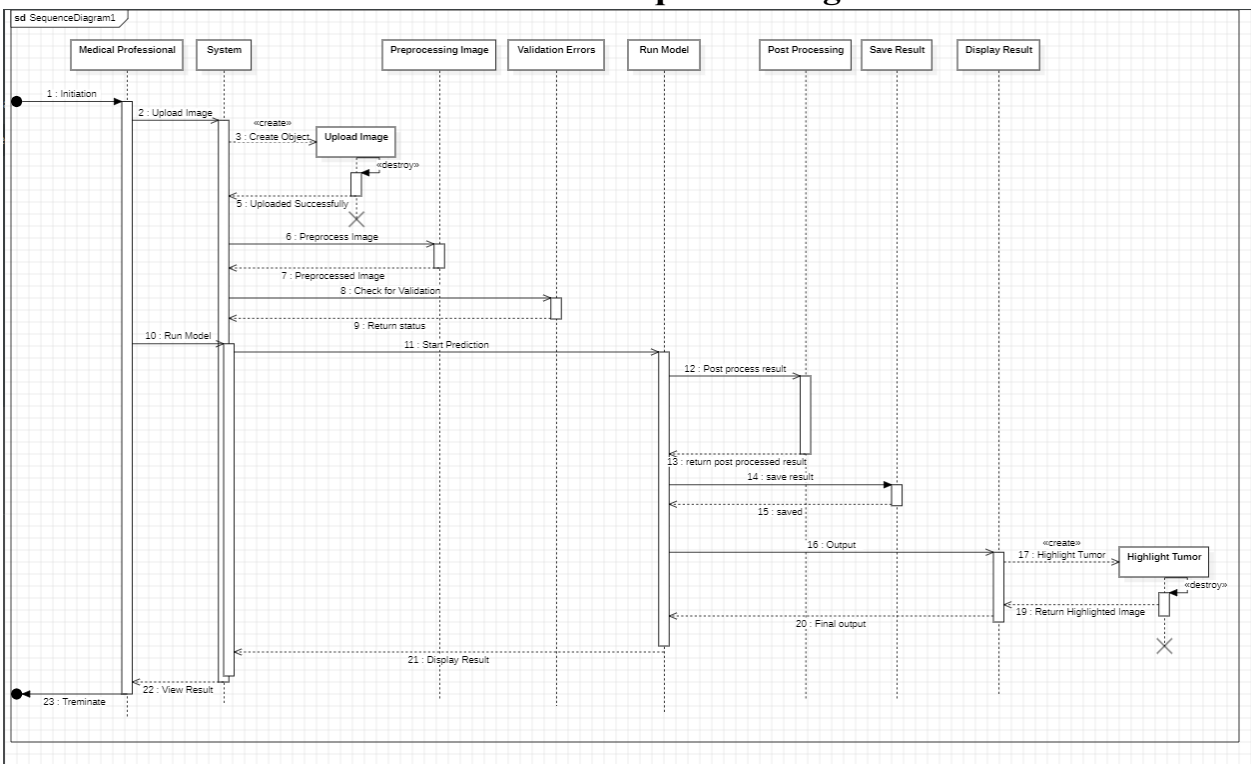

Figure 5.2: Sequence Diagram

The sequence diagram illustrates the workflow of a medical imaging system, focusing on image preprocessing, model execution, and result analysis. It highlights the interactions between various actors and processes, showcasing how medical professionals and systems collaborate for accurate diagnosis and predictions.

Objects (Lifelines)

1. **Medical Professional**
    - Represents the user (doctor/technician) interacting with the system.
    - Initiates the process by uploading an image and views the final result.
2. **System**
    - Refers to the AI-based Brain Tumor Detection System that processes and analyzes the uploaded image.
3. **Preprocessing Image**

- Represents the stage where the uploaded image is prepared for analysis by adjusting dimensions, removing noise, or enhancing quality.

4. **Validation Errors**
    - A checkpoint that ensures the uploaded image meets necessary criteria (e.g., correct format, sufficient resolution) before proceeding.
5. **Run Model**
    - Refers to the stage where the core AI model processes the preprocessed image to detect potential brain tumors.
6. **Post Processing**
    - Processes the raw output of the model, such as refining results, adding overlays to images, or generating user-friendly data.
7. **Save Result**
    - Stores the results of the analysis, ensuring that they can be accessed later for clinical documentation or further analysis.
8. **Display Result**
    - Outputs the final results to the medical professional in a comprehensible format (e.g., visual representation or report).

Flow of Activities

1. **Step 1-2**:
    - The **Medical Professional** starts the process and uploads an image to the **System**.
2. **Step 3-5**:
    - The system creates an object for the uploaded image and confirms a successful upload.
3. **Step 6-9**:
    - The uploaded image goes to the **Preprocessing Image** stage. The system validates the preprocessed image and checks for any validation errors.
    - If validation is successful, a status is returned to proceed further.
4. **Step 10-11**:
    - The system begins the **Run Model** phase, using the AI model to make predictions based on the image.

5. **Step 12-13**:
    - The results are sent to the **Post Processing** stage for refinement and formatting.
6. **Step 14-15**:
    - The processed results are saved using the **Save Result** component.
7. **Step 16-20**:
    - The processed and saved results are sent to the **Highlight Tumor** stage, where a visual representation of the detected tumor is created.
    - The highlighted tumor image is returned to the system and forms part of the final output.
8. **Step 21-23**:
    - The system displays the final results to the **Medical Professional**, who reviews the output. The process terminates after the result is viewed.

UML Advanced Activity Diagram

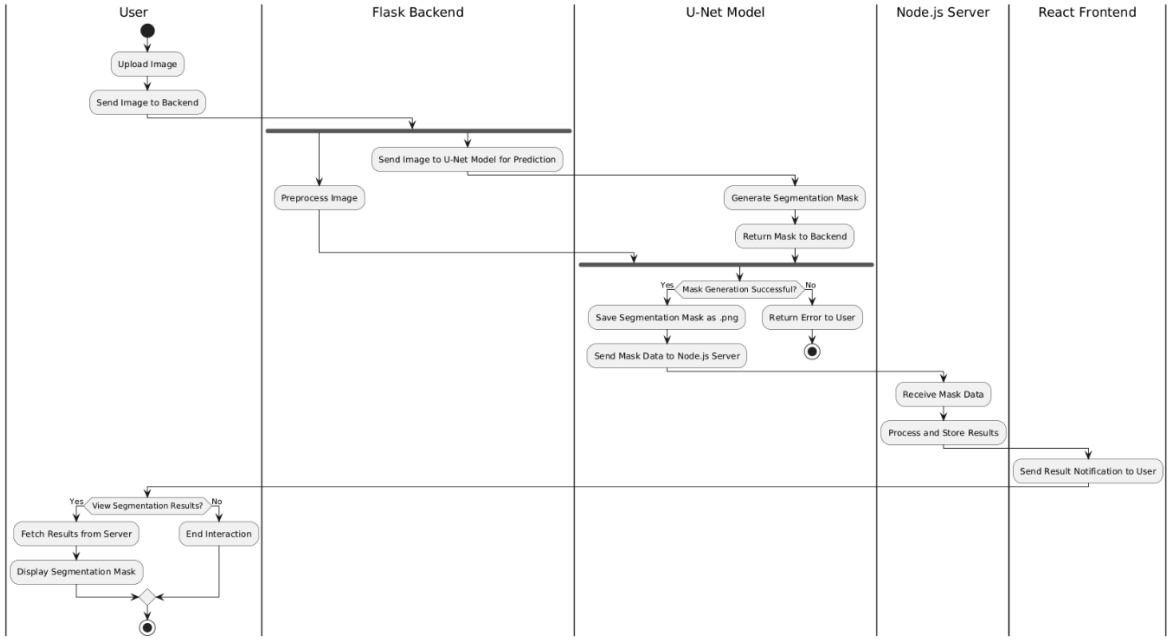

Figure 5.3: Activity Diagram

Components (Swimlanes):

1. **User:** The person uploading and interacting with the application.
2. **Flask Backend:** A Python-based server responsible for preprocessing and routing data.
3. **U-Net Model:** A deep learning model used for generating image segmentation masks.
4. **Node.js Server:** A server responsible for processing and storing results.
5. **React Frontend:** The client-side interface where the user views the results.

Workflow:

1. User Interaction

- **Upload Image:** The user uploads an image.
- **Send Image to Backend:** The image is sent to the Flask backend for further processing.

2. Flask Backend

- **Preprocess Image:** The backend processes the image (e.g., resizing or normalization) to prepare it for the U-Net model.

- **Send Image to U-Net Model for Prediction:** The preprocessed image is sent to the U-Net model for segmentation.

3. U-Net Model

- **Generate Segmentation Mask:** The model generates a segmentation mask for the uploaded image.
- **Return Mask to Backend:** The generated mask is returned to the Flask backend.
- **Mask Generation Check:**
    - If **successful:** The mask is saved as a .png file, and data is forwarded to the Node.js server.
    - If **unsuccessful:** An error message is sent back to the user.

4. Node.js Server

- **Receive Mask Data:** The server receives the segmentation mask data from the Flask backend.
- **Process and Store Results:** It processes and stores the segmentation results.
- **Send Result Notification to User:** A notification is sent to the user (possibly via the React frontend) indicating the results are ready.

5. React Frontend

- **View Segmentation Results?**
    - If **Yes:** The React frontend fetches the results from the server and displays the segmentation mask.
    - If **No:** The interaction ends

CHAPTER 6: UI Design with Screenshots

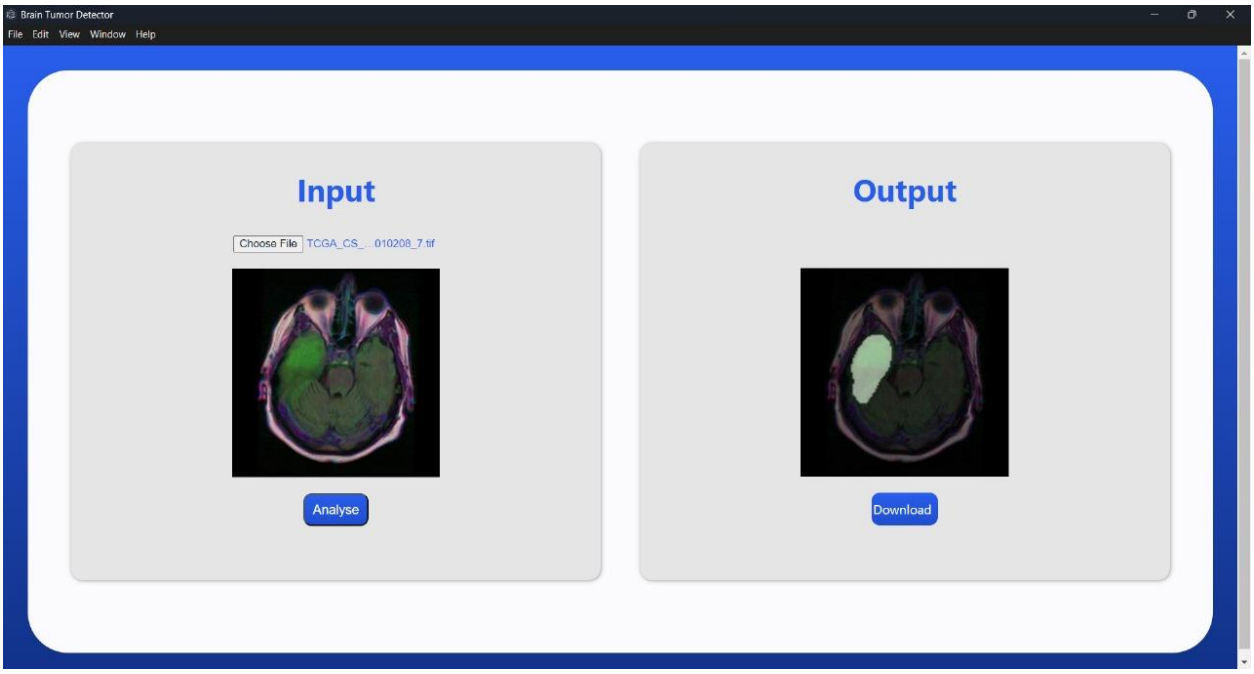

Figure 6.1: UI Design Screenshot

This UI design represents a simple and user-friendly **Brain Tumor Detection application**. Here's an explanation of its components:

Overall Layout

The interface is split into two sections: **Input** and **Output**, arranged side by side for a clear workflow.

Input Section (Left Panel)

1. **Title:** Labelled "Input" to indicate where the user interacts with the application to upload an image.
2. **File Upload:** A "Choose File" button allows users to select an image file (likely a medical scan such as an MRI or CT scan) from their computer.
3. **Preview Image:** Below the file selector, the chosen image is displayed, giving a visual confirmation of the uploaded file.
4. **Analyse Button:** A blue button labelled "Analyse" initiates the tumor detection process. It likely runs the algorithm to detect and analyse the uploaded scan.

Output Section (Right Panel)

1. **Title:** Labelled "Output" to indicate the processed results.
2. **Result Image:** Displays the processed image with the detected tumor region highlighted. In this example, the tumor region is shaded in **green**, which helps in visualizing the abnormal area.
3. **Download Button:** A blue button labelled "Download" allows users to save the processed result image to their device.

Styling

- The UI has a clean and professional look, using a combination of white and blue, which is common for medical applications.
- Buttons are well-highlighted in blue, drawing attention to actionable items.
- The preview and result images are prominently displayed for clarity.

Intended Workflow

1. User selects a medical scan via the "Choose File" button.
2. The image is displayed under the "Input" section.
3. The user clicks "Analyse" to process the image.
4. The output, highlighting the detected tumor, is shown in the "Output" section.
5. If needed, the user can download the processed result for further use.

This design is intuitive and makes it easy for users, including non-technical individuals, to interact with the application efficiently.