# DTSC 5565/DTSC 4565: Software Engineering for Data Scientists

# TITLE - AI PRECISION FOR BRAIN TUMOR PREDICTION

# Team Details

**Team Leader:** Jahnavi Chintakindi

**Team Members:**

* Anoohya Alivelu Bhaskarla
* Jahnavi Chintakindi
* Pranav Moses
* Saipradeep Bomma
* Sai Vaishnavi Govindula

**Team Responsibilities:**

| **Team Member** | **Responsibility** |
| --- | --- |
| **Anoohya Alivelu Bhaskarla** | Programming, Design, Presentation |
| **Jahnavi Chintakindi** | Programming, Design, Presentation |
| **Pranav Moses** | Programming, Design, Presentation |
| **Saipradeep Bomma** | Documentation, Management, Presentation |
| **Sai Vaishnavi Govindula** | Documentation, Management, Presentation |

# Table of Contents

1. Introduction
2. Customer Statement of Requirements
3. Glossary of Terms
4. Functional Requirements Specification
5. Nonfunctional Requirements
6. Domain Analysis
7. Interaction Diagrams
8. Class Diagram and Interface Specification
9. System Architecture and Design
10. Algorithms and Data Structures
11. User Interface Design and Implementation Details
12. summary of changes
13. Unique Capability
14. Conclusions and Future Work
15. References

## INTRODUCTION

## Revolutionizing brain tumor detection, the Brain Tumor Detection Website employs advanced Convolutional Neural Networks (CNNs) to rapidly and accurately identify tumors from MRI scans. This innovative tool is designed to empower researchers, physicians, and patients, making sophisticated AI technology accessible to the broader healthcare community.

## 1.1 Core Objectives

## a) Early Tumor Detection: By accurately classifying tumor, the platform enables timely diagnosis and intervention, improving patient outcomes.

## b) Broad User Accessibility: The user-friendly design caters to diverse users, from patients seeking information to researchers driving advancements.

## c) Protecting Patient Privacy: The system prioritizes data security by implementing robust measures to safeguard sensitive medical information.

## d) High-Performance Capability: The scalable architecture ensures efficient processing of large datasets, meeting the demands of busy healthcare environments.

## 1.2 Key Features

1. AI-Powered Analysis: Using pretrained CNN models, the platform processes MRI images to identify tumor types, providing results with confidence scores.
2. Comprehensive Reports: Generated reports include tumor size, location, type, and recommendations for further testing or treatment, aiding in holistic diagnosis.
3. Multi-Role Support: Custom dashboards for patients, medical professionals, and researchers provide personalised experiences.
4. Secure Data Handling: All data is encrypted and securely kept, with access controlled via roles to protect sensitive information.

## 1.3 Impact on Healthcare

The Brain Tumour identification Website solves major healthcare concerns by shortening diagnostic time, decreasing mistakes, and providing an affordable option for tumour identification. The platform connects powerful AI technologies to real-world medical demands, making critical diagnostics simpler to obtain, accurate, and efficient.   
This initiative improves healthcare results while also establishing a precedent for subsequent AI-based diagnostic systems by using cutting-edge algorithms and complying to strong data protection regulations. It shows AI's potential to improve patient care, speed advancements in medicine, and streamline healthcare procedures worldwide.

## Customer Statement of Requirements

The Brain cancer detection Website allows users (scholars, patients, and clinical professionals) to submit MRI pictures for evaluation in order to facilitate brain cancer detection. The technology use algorithms that apply machine learning to provide a comprehensive classification (dangerous or harmless) and predict the presence of a brain tumour. The website will be created with precision, speed, and medical confidentiality of information in consideration, and it will be easy to use by a wide variety of users.

## 2.1 Target Audience:

**Medical Professionals**: Upload MRI scans to the system to aid in diagnosis and spot tumours quickly.

**Patients**: those who are looking for a second opinion or an early diagnosis on the presence of a brain tumour.

**Researchers**: scholars who want to assess the correctness of models, examine datasets, and suggest enhancements.

## Set of Requirements:

## 2.2.1 Medical Professionals:

## Functional Requirements:

**Secure Login**: To use the system and submit MRI images, medical practitioners must securely log in utilising authentication measures (such as multifactor authentication).

**MRI Scan Upload**: Compatibility with various hospital systems and imaging modalities is ensured by the ability to upload MRI images in several formats (such as PNG, JPEG).

**Tumor Detection and Classification**: The uploaded MRI images should be analysed by the system using machine learning models, which will then produce findings that include a likelihood score for the tumours' categorisation as benign or malignant.

**Report Generation**: Following analysis, a thorough tumour analysis report in PDF format ought to be available for download by medical practitioners. Details such tumour size, kind, location, and suggestions for more testing or action will all be included in this report.

## Non-Functional Requirements:

**High Accuracy**: To be recognised as a viable diagnostic tool, the instrument must routinely have a tumour detection accuracy rate that is more than 90%.

**Fast Processing**: Results should be accessible within five minutes of upload, and the system should offer realtime or almost realtime feedback. In clinical situations where prompt diagnosis is crucial, this speed is crucial.

**Data Security**: The system must adhere to applicable data privacy rules since it handles sensitive medical data. To avoid unwanted access, all uploaded MRI reports and images need to be safely kept and encrypted.

**Scalability**: The platform needs to be scalable in order to manage massive amounts of requests and uploads, particularly in big hospitals or during periods of heavy usage.

* + 1. Patients:
* Functional Requirements:

User Friendly Interface: Even patients with little technological expertise should be able to submit their MRI images with ease because to the system's straightforward, userfriendly interface.

Preliminary Tumor Detection: A tumour identification report that offers concise, understandable feedback should be given to patients. The report will specify the kind of tumour (malignant or benign) and if a tumour is likely to be present.

Access to Historical Data: For their own records, patients ought to be able to view their prior reports. They should be able to access or download previous analyses, and the system should safely preserve their MRI images and findings.

* NonFunctional Requirements:

Accessible Design: In accordance with online accessibility principles such as WCAG, the user interface is intended to be inclusive and accessible to people with disabilities. It eliminates medical jargon and utilises simple English to make it easy for everyone to use.

**Quick Turnaround Time:** For the system to be helpful for initial analysis, patients need obtain data in a short period of time (less than five minutes), much like medical professionals.

**Security and Privacy:** Only authorised usersthat is, the patient and system administratorsshould be able to access the encrypted patient data, according to the platform. The system should also abide by laws pertaining to data security and patient privacy.

**2.2.3 Researchers**:

* Functional Requirements:

Dataset Upload: Large MRI datasets have to be accessible for study by researchers. For the system to analyse numerous scans at once, batch processing must be supported.

Comparison of Machine Learning Models: Researchers should be able to assess the efficacy and accuracy of various machine learning models in identifying brain tumours by running them on the uploaded datasets.

Access to Data Analytics: The platform need to include sophisticated analytics capabilities that enable researchers to spot patterns and trends in tumour identification across various datasets.

* **Non-Functional Requirements:**

Customization Options: By modifying model parameters or experimenting with various methods, researchers need to be able to personalise the anal

mnytytic procedure. Because of its adaptability, researchers may experiment and improve the detection models.

Robust System:Large datasets must be handled by the platform, and precise, thorough analysis reports must be produced. It should be able to process large amounts of data without experiencing performance issues or system failures.

2.2.4 System Administrators:

* Functional Requirements:

User Management: Adding new users, changing user roles, and making sure that only authorised users have access to particular system functions are all tasks that administrators should be able to perform.

System Monitoring: The system's security, utilisation, and performance should all be observed by administrators. This include keeping an eye on data storage, controlling server load, and making sure the system is operating effectively.

Model Updates: To ensure ongoing improvements in tumour detection accuracy, administrators should be able to change the machine learning models on a frequent basis when new datasets or improved methods become available.

* NonFunctional Requirements:

High Uptime: Because medical professionals depend on the system for vital diagnostic reasons, it must be accessible and functional 99.9% of the time.

Security and Auditing: For auditing purposes, the platform should keep thorough logs of every user action. This guarantees responsibility and openness, especially when handling private medical information.

System Backup: Maintaining regular backups is necessary to guard against data loss in the event of a system breakdown.

2.3 Use case scenarios for each actor

2.3.1 Scenario 1: Medical Professional Uploading MRI Scan

Unique Name: Upload an MRI scan to find tumours.

Participating Actors: Health Care Provider (Physician or Radiologist), System

Entry Conditions:

The physician is signed in to the system.   
b. The MRI scan file can be uploaded.

Exit Conditions:

a. The MRI scan has been uploaded and examined satisfactorily.   
b. The report on tumour detection is created and stored.

Flow of Events:

1. The physician logs into the system safely.   
2. The option to upload an MRI scan is chosen by the physician.  
3. From the local computer, the doctor browses and chooses the MRI scan file.   
4. The system uses a machine learning model to process the MRI scan that was submitted.   
5. A thorough tumour analysis report with suggestions and a classification (malignant or benign) is produced by the system.   
6. The report is downloaded or stored for further use by the physician.

Special Requirements:

a. The system must guarantee that MRI files are in a recognised format, such as DICOM, JPEG, or PNG.   
b. For realtime diagnostic reasons, the analysis needs to be finished in five minutes.  
c. To adhere to medical data privacy standards (such as ), highlevel encryption must be used for both the submitted data and the generated report.

2.3.2 Scenario 2: Patient Seeking Preliminary Diagnosis

Unique Name: MRI Upload of the Patient for Initial Tumour Identification

Participating Actors: System, Patient

Entry Conditions:

a. The patient is now logged into the system after creating an account.   
b. The MRI scan can now be uploaded.

Exit Conditions:

a. The MRI scan has been uploaded and examined satisfactorily.   
b. A preliminary tumour detection report is given to the patient.

Flow of Events:

1. First, the patient accesses their account.   
2. For a preliminary diagnostic, the patient chooses to upload an MRI scan.  
3. The MRI scan file is browsed and chosen by the patient.   
4. The machine learning model is used by the system to process the uploaded scan.   
5. The system produces a tumour detection report that indicates the likelihood of a tumour (malignant or benign).   
6. After reading the report, the patient determines whether to seek more medical guidance.

Special Requirements:

a. Nontechnical users should be able to easily use the user interface.   
b. Reports should concentrate on whether a tumour is found and its possible categorisation, using plain language and little technical jargon.  
c. To safeguard patient privacy, secure data management must be used.

2.3.3 Scenario 3: Researcher Testing ML Models

Unique Name: Uploading MRI Datasets for Model Testing

Participating Actors: Researcher, System

Entry Conditions:

a. The researcher has access to the platform's research tools and login credentials.   
c. A fresh MRI dataset is ready for testing and upload.

Exit Conditions:

a. The machine learning models are successfully used to analyse the uploaded MRI dataset.   
b. Comparative results between several models are provided to the researcher.

Flow of Events:

1. The investigator enters the system.   
2. To test the model, the researcher chooses to upload a fresh MRI dataset.  
3. Several machine learning models are used by the system to process the dataset.   
4. The researcher examines the findings and contrasts the model's tumour detection accuracy.  
5. The researcher uploads fresh datasets for additional testing and modifies the model's parameters as needed.   
6. The final analysis is stored for use in further studies or model enhancements.

Special Requirements:

a. Batch processing should be supported by the system, which can manage big datasets.   
b. The researcher has to be able to alter testing factors including data preparation and model settings.  
c. To visualise trends, accuracy rates, and false positives/negatives in tumour identification, comprehensive analytical tools must be made available.

2.3.4 Scenario 4: System Administrator Managing Users

Unique Name: Overseeing User Accounts and System Upkeep

Participating Actors: System Administrator, System

Entry Conditions:

a. The administrator can access the admin panel and has login credentials.

Exit Conditions:

a. The system continues to function and user accounts are maintained.   
b. If required, the machine learning models are upgraded.

Flow of Events:

1. Using their admin credentials, the system administrator signs in.   
2. To create, delete, or change user roles, the administrator goes to the user management panel.  
3. The administrator looks for any security or performance issues in the system logs.   
4. The administrator adds fresh datasets or algorithm enhancements to the machine learning model as needed.   
5. The administrator makes sure the system is operating well by keeping an eye on performance indicators.   
6. After making sure everything is functioning properly, the administrator signs out.

Special Requirements:

For security and transparency reasons, the system must keep an audit record of all administrative actions.   
b. The system needs to comply with data protection regulations (such as GDPR and ) and enable safe data processing.  
c. Frequent updates and maintenance plans must be put into place without compromising user access to the system.

## Glossary of Terms

* **MRI (Magnetic Resonance Imaging):** MRI, or magnetic resonance imaging, is a secure medical technology that creates detailed images of internal body parts, particularly soft tissues including the brain.
* **CNN (Convolutional Neural Network):** A machine learning method for classification and recognition of images.. It identifies patterns and characteristics in input pictures, making it appropriate for health care imaging jobs like detecting brain tumours.
* **Flask:** This project uses Flask, a lightweight Python web framework, to ease backend development. It allows for the seamless integration of predictive algorithms with user interfaces, making it easier to create efficient and responsive online applications.
* **React.js:** A JavaScript library to generate responsive, dynamic user interfaces. React.js powers the front end of the Brain Tumour Detection Website, ensuring an automatic and engaging user experience.
* **MongoDB:** NoSQL database system designed to hold semi-structured or unorganised data. It offers the safe storing of user data, magnetic resonance imaging information about them, along with evaluation reports in scalable format.
* **AUC (Area Under the Curve):** A performance metric designed to assess the accuracy of models of classification. It measures a model's ability to differentiate across classes, where higher scores representing better performance.
* **NumPy:** A Python tool for performing maths on big collections of integers, particularly helpful for interacting with MRI pictures and data.
* **Pandas:** A Python library used for data manipulation and analysis. It is employed to handle data frames, enabling researchers to analyze model performance and generate insights.
* **TensorFlow/Keras:** Open-source neural network libraries are used for training and deploying deep learning models, like as CNNs, for applications like tumour classification.
* **Matplotlib/Seaborn:** Python data visualisation components help to create diagrams, charts, and heat maps, facilitating users to gain a better understanding of the efficacy of models.
* **Role-Based Access Control (RBAC):** A system that restricts access by applying role-based criteria, allowing only authorised users to access specific information or services.

1. Functional Requirements Specification

**4.1 Stakeholders**

The people or organisations that have a stake in the Brain Tumour Detection Website's functioning, success, and results are known as stakeholders. Key stakeholders are broken down as follows:

1. Medical Professionals (Doctors, Radiologists):

Interest: Their decision-making process is accelerated by the system's ability to analyse MRI data and help diagnose brain tumours.

Role: System users who submit scans, get tumour detection reports, and use the information to inform their diagnosis.

1. Patients:

Interest: Patients submit their MRI scans and receive an automated analysis report in order to obtain second views or preliminary diagnosis. This aids in their decision to seek more medical assistance.

Role: Results of tumour detection are sent to those who engage with the system.

1. Researchers:

Interest: The platform is used by researchers to evaluate big MRI datasets and test machine learning models in an effort to increase the precision and effectiveness of brain tumour identification.

Role: Users who compare models, do analyses, and submit datasets for scholarly study or model enhancement.

1. System Administrators:

Interest: Make that the system runs effectively, safely, and seamlessly. They are in charge of updating machine learning models, managing users, and maintaining the system.

Role: backend administration to guarantee the system's security and functionality.

1. Sponsors/Investors:

Interest: people or groups providing cash for the system's creation and upkeep. Their interest is in the platform's success as it may result in lucrative applications in the research and healthcare sectors.

Role: Give the initiative both strategic and financial backing.

1. IT Support and Developers:

Interest: In charge of the system's creation, testing, and ongoing enhancement. They concentrate on making sure the system satisfies user needs and technical criteria.

Role: Create and manage the system, making sure it functions properly and fulfils all requirements.

4.2 Actors and Goals

The entities (individuals or systems) that communicate directly with the Brain Tumour Detection Website are known as actors. They fall into two categories: initiating actors, who set off systemic events, and participating players, who react to those events. The actors and their objectives are listed below:

Actors:

Medical Professional (Doctor or Radiologist)

Type: Initiating actor

Goals:

i. Upload patient MRI images.   
ii. Get tumour detection results that are classified as either benign or malignant.  
iii. Use the system's reports to help with patient diagnosis and care.   
iv. Download patient records or reports for future use.

* Patient

Type: Initiating actor

Goals:

i. Provide their MRI scans for a second opinion or a preliminary diagnosis.   
ii. Get a report on tumour detection that shows the probability of a tumour.  
iii. Consider the report while determining whether to get additional medical help.

* Researcher

Type: Initiating actor

Goals:  
i. Provide MRI datasets for machine learning model testing.   
ii. Examine model performance and find patterns or trends in tumour identification.  
iii. To improve detection algorithms, compare several models.   
iv. Apply the information and understanding to scholarly works or more study.

* System Administrator

Type: Participating actor

Goals:

i. Control patient, researcher, and medical professional accounts and permissions.   
ii. As new datasets become available, keep machine learning models up to date.  
iii. Keep an eye on system performance, security, and dependability.   
iv. Verify that the system conforms with strong security requirements and data privacy laws.

4.3 Use Cases

Casual Description

The Brain Tumour Detection Website's primary use cases, which involve interactions between the system and its main players (patients, researchers, medical practitioners, and system administrators), are as follows:

Medical Professional Uploading MRI Scan: After uploading an MRI scan, a physician gets a thorough analysis along with a report on the tumor's categorisation as benign or malignant.

Patient: Seeking Preliminary DiagnosisTo determine whether additional medical consultation is required, a patient uploads their MRI image to receive an initial tumour detection result.

Researcher Testing ML Models: To test and evaluate several machine learning models for tumour identification, a researcher uploads MRI datasets.

System Administrator : Managing UsersThe administrator refreshes machine learning models, keeps an eye on system performance, and oversees user accounts.

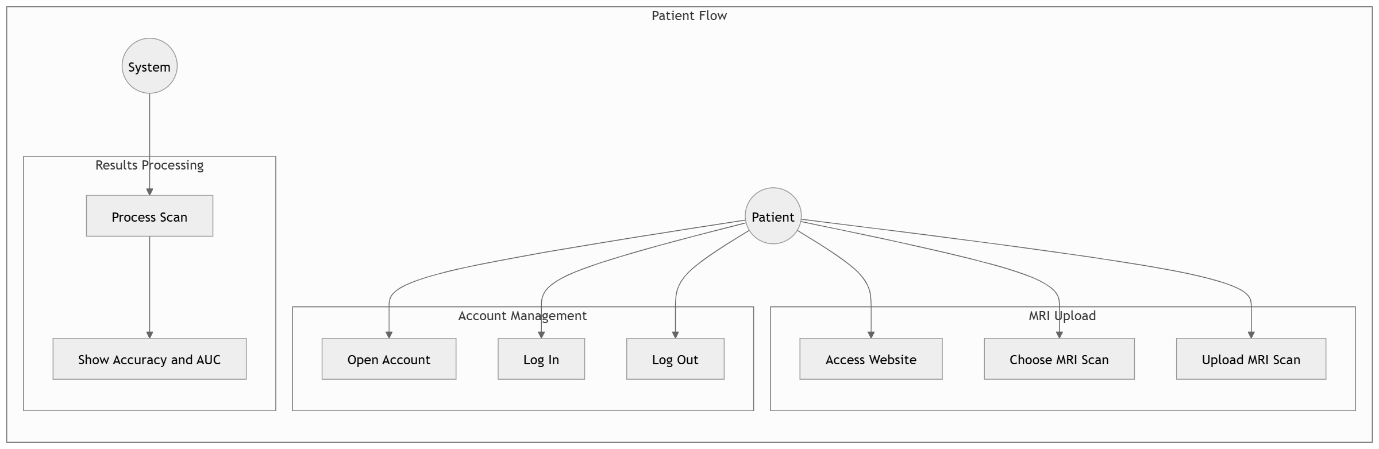
Fully Described Use Cases

* Use Case 1: Patient Uploading MRI Scan for Preliminary Analysis

1.**Unique Name**: Submit MRI Picture for Initial Evaluation   
2. **Participating Actors**: System, Patient  
3. **Entry Conditions**:   
The patient has successfully registered and logged in.   
The file containing the MRI scan can be uploaded.   
4. **Exit Conditions**:   
The database has successfully received and saved the MRI scan.   
After processing the scan, the system shows the accuracy and AUC values.   
After seeing the findings, the patient logs off.   
5. **Flow of Events**:   
The patient opens their account and logs in.   
2. The patient accesses the website for uploading the MRI.   
3. An MRI scan is chosen and uploaded by the patient.   
4. The pretrained CNN model is used by the system to process the scan.   
5. On the patient's dashboard, the system shows accuracy and AUC measurements.   
6. The patient logs out after seeing the results.

6. **Special Requirements:**

The interface needs to be easy to use and straightforward, with unambiguous upload instructions.  
The maximum processing time is five minutes.   
For privacy and security, all uploaded data must be encrypted.

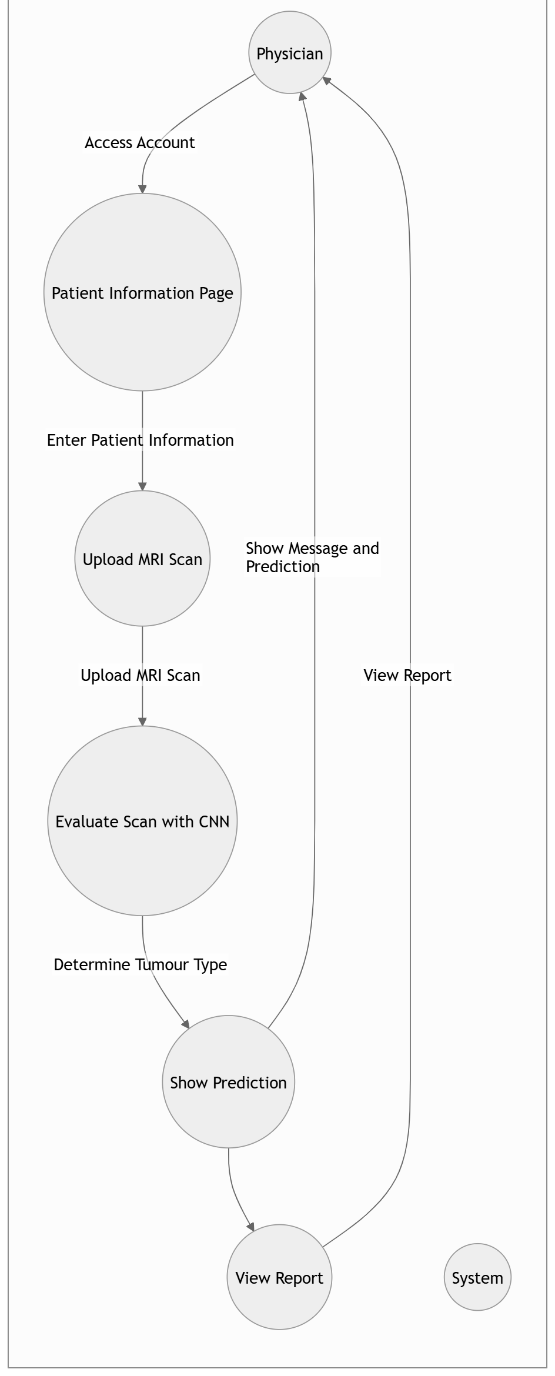


* **Use Case 2: Doctor Uploading and Viewing Patient's MRI Analysis**

1 Unique Name: MRI Scan Upload for Patient Identification   
2. Participating Actors: Physician, System  
3. Entry Conditions:   
The physician has successfully made an account, logged in, and obtained patient information.   
The MRI scan file for the patient is prepared for upload.   
4. Exit Conditions:   
The MRI scan is processed, uploaded, and examined.   
A static notice regarding the tumour categorisation is shown next to the prediction results.   
After examining and perhaps storing the analysis, the doctor signs off.   
5. Flow of Events:   
1. The physician accesses their account.   
2. The physician accesses the page to enter patient information.   
3. The patient's MRI scan is uploaded by the doctor.   
4. The system uses CNN to evaluate the scan and determine the type of tumour.   
5. A static message and prediction on the MRI analysis are shown.   
6. Doctor views report

**6. Special Requirements**:

The patient's profile must be connected to the MRI scan information and analysis.



* **Use Case 3: MRI Algorithm Evaluation and Comparison**

**1. Unique Name**: MRI Algorithm Evaluation and Comparison

**2. Participating Actors**:

Primary Actor: Researcher

Secondary Actor: System

**3. Entry Conditions**:

The researcher has login credentials to access the system.

An MRI image or dataset is ready for upload.

The system has preloaded models (e.g., CNN, SVM, Logistic Regression).

**4. Exit Conditions:**

The system successfully processes the dataset/image and displays a comparison of metrics for selected algorithms.

The researcher logs out after reviewing or downloading the results.

**5. Flow of Events:**

The researcher logs into the system.

The researcher navigates to the ML Researcher Page.

The researcher uploads an MRI image or dataset for analysis.

The researcher selects one or multiple algorithms for evaluation (e.g., CNN, SVM, Logistic Regression).

The system processes the input using the selected algorithms.

The system generates and displays the following results for each algorithm:

1. Classification report (accuracy, precision, recall, F1-score).

2. Confusion matrix (numerical data and heatmap visualization).

3. ROC curve with AUC score.

4. Predictions visualization (sample MRI images with predicted vs actual labels).

The researcher navigates to the Algorithms Dashboard to view a comparative analysis of all selected algorithms.

The researcher downloads the results (PDF/CSV format) if needed.

The researcher logs out.

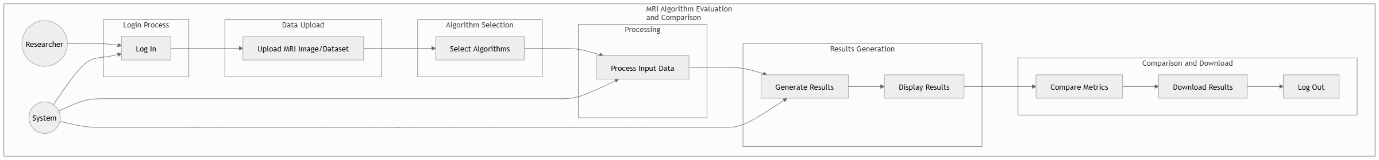
**6. Special Requirements:**

Support for batch uploads and large datasets.

Option to select multiple algorithms for evaluation simultaneously.

Clear and intuitive visualizations, including confusion matrix heatmaps, ROC curves, and prediction grids.

Downloadable reports and visualizations in standard formats (PNG, PDF, CSV).



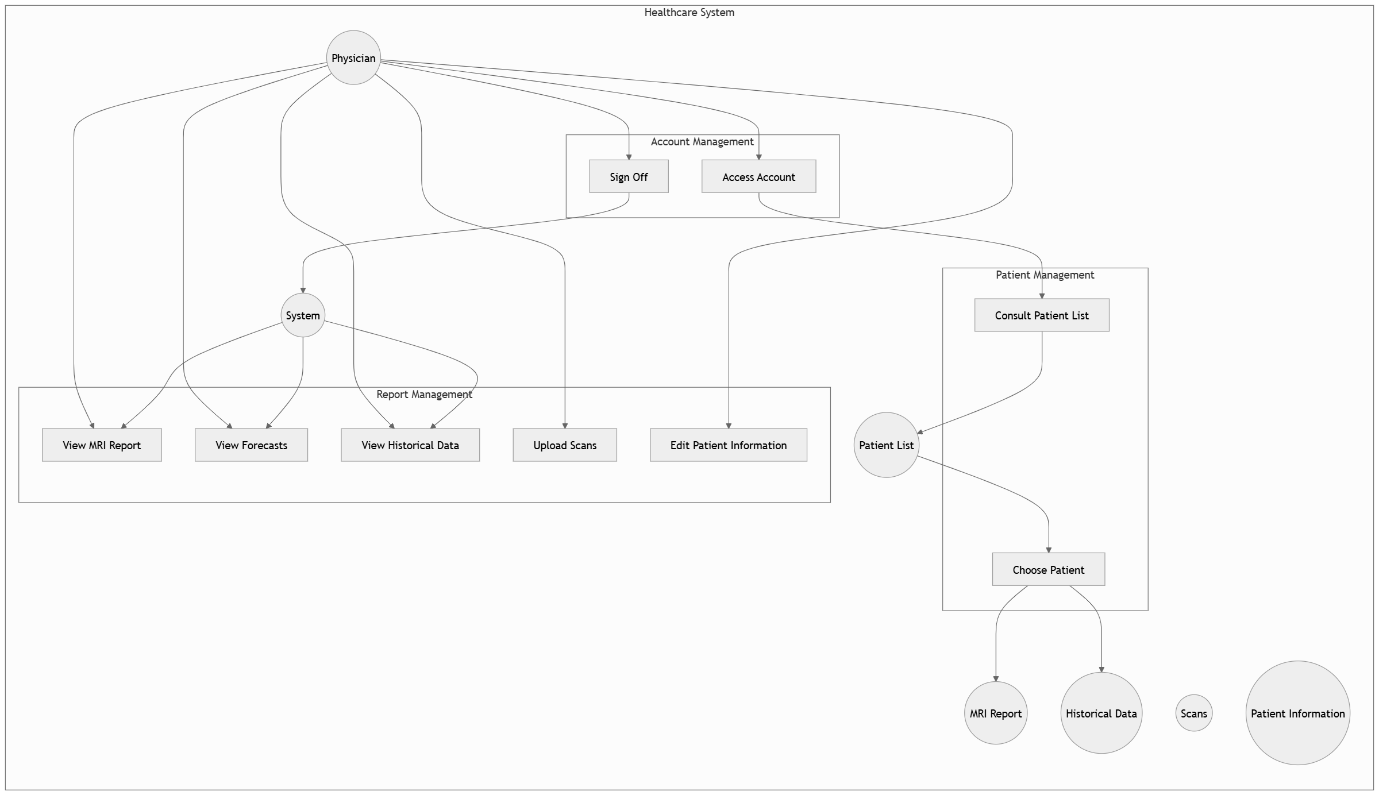
* Use Case 4: Doctor Managing Multiple Patients' MRI Reports

1. Unique Name: View and Manage Reports for Several Patients   
2. Participating Actors: Physician, System  
3. Entry Conditions:   
The patient's data is accessible to the doctor, who is logged in.   
4. Exit Conditions:   
The physician is able to obtain and manage the MRI reports of several patients.   
The doctor signs off after going over the required reports.   
5. Flow of Events:   
1. The physician accesses their account.   
2. The doctor consults a patient list.   
3. To examine the MRI analysis, the doctor chooses a certain patient.   
4. The MRI report, forecasts, and, if relevant, historical data are shown by the system.

5. A doctor can upload fresh scans or change patient information.   
6. After handling the reports, the doctor signs off.

6. Special Requirements:

Rolebased access control to guarantee that patient data is accessed by only authorised physicians.  
For fast report retrieval, the system must include a searchable interface.



**4.4 System Requirements Use Case Traceability Matrix**

The functional requirements and use cases are connected via the Use Case Traceability Matrix. A matrix like this may be made:

|  |  |  |
| --- | --- | --- |
| **Use Case** | **System Requirement** | **Description** |
| Upload an MRI scan to find tumours. | Secure login | MRI scans may only be uploaded by verified individuals. |
| Patient Looking for a Preliminary Assessment | Simple interface | Uploading scans is simple even for nontechnical people. |
| ML Model Testing by Researchers | Batch processing | MRI datasets may be uploaded to the system in batches. |
| Managing Users as a System Administrator | User management | Admins have the ability to modify, add, and delete user roles. |

1. **System Sequence Diagram Creation:**

A System Sequence Diagram (SSD) illustrates the interaction between an external actor (such as a user) and the system. It focuses on the sequence of messages exchanged to fulfill a particular use case. Below is a detailed explanation of how to create the System Sequence Diagram (SSD) for key use cases in the Brain Tumor Detection Website.

**Flow of Events**

a. The user logs into the MRI Management System.

b. Based on their role, they select an action:

* **Patients** upload an MRI scan for analysis.
* **Doctors** upload a patient's MRI scan, view reports, or manage multiple patient records.
* **Researchers** upload a dataset for algorithm comparison.

c. The uploaded data is processed by the system:

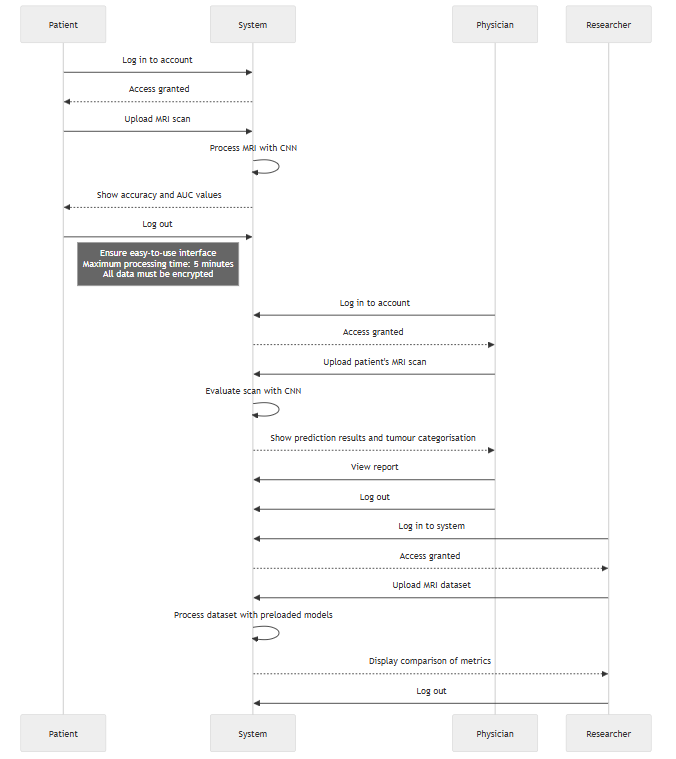
* Pretrained CNN model is used for MRI analysis.
* Multiple algorithms are applied for dataset comparison.
* Predictions, such as tumor type, and metrics (accuracy, AUC, confusion matrix) are calculated.

d. Results are displayed:

* **Patients** view accuracy and AUC on their dashboard.
* **Doctors** see detailed MRI reports, access patient lists, and review historical or updated data.
* **Researchers** receive comparative analysis reports of various algorithms.

e. Doctors can update patient information or add new scans if needed.

f. The user logs out after completing their tasks.



1. **Nonfunctional Requirements**

A System Sequence Diagram (SSD) depicts the communication between an external actor (often a user) and the system. It focusses on the chain of information sent to complete a specific use case. Below is a detailed explanation of how to create the System Sequence Diagram (SSD) for key use cases in the Brain Tumor Detection Website

* 1. Functionality

Security:

a. All MRI uploads, reports, and user data must be encrypted by the system to safeguard sensitive medical data.   
b. Only authorised users (such as physicians or patients) will be able to access particular system operations thanks to rolebased access control, or RBAC.  
b. The program will secure medical data in accordance with requirements.

Error Handling:

a. The system will have extensive error detection and recovery capabilities for malformed uploads or inappropriate MRI file types.   
b. Errors will be recorded, and users will see the relevant feedback (such as "Invalid File Format").

* 1. Usability

User Interface:

a. The website will include an easytouse interface that makes it simple to submit MRI scans and obtain results.   
b. Special attention will be paid to accessibility, guaranteeing that the website is easy for people with disabilities to use (WCAG compliance).  
**Ease of Use**:

Patients and other nontechnical users will engage with a streamlined interface to read reports and submit scans.   
b. Reports shall be written in clear English with little technical jargon.

* 1. **Reliability**

**Availability**:

a. The system must be accessible 99.9% of the time to provide continuous access to essential features, particularly for healthcare providers who depend on it for realtime diagnosis.

b. Emergency notifications will be put in place in the event of system problems, and planned maintenance downtime will be reduced.

**Data Backup and Recovery**:

To guarantee that all user information, MRI scans, and reports are safe and recoverable in the event of a system failure or data loss, automated backups will be planned.

* 1. Performance

Response Time:

Within five minutes, the system must evaluate MRI uploads and provide tumour detection findings. In healthcare situations, quick analysis is essential.   
b. When researchers upload several datasets, the system will facilitate batch processing.  
**Scalability**:

a. The platform needs to be scalable in order to accommodate a large number of concurrent users and MRI uploads, particularly during busy periods in medical facilities or research laboratories.

* 1. Supportability

Maintainability:

a. Because the system will be constructed using a modular design, it will be simple to upgrade certain pieces, such the database or machine learning models, without impacting other system components.

Documentation:

a. Detailed documentation, such as installation instructions, user manuals, and API documentation for developers or administrators, will be supplied.

Monitoring:

Tools for ongoing system monitoring will be put in place to keep tabs on performance and identify any problems before they have an impact on users

1. **Domain Analysis**

**7.1 Domain Model:**

The fundamental ideas and connections in the Brain Tumour Detection Website system are represented by the domain model. The domain model is described here, followed by the required definitions:

**Concept Definitions Table**

|  |  |
| --- | --- |
| **Concept** | **Definition** |
| **User** | General entity representing any individual interacting with the system (patient, doctor, researcher). |
| **Patient** | A user who uploads MRI scans to check for brain tumors and accesses previous reports. |
| **Doctor** | A medical professional who uploads MRI scans and generates diagnostic reports. |
| **Researcher** | A user who uploads datasets to evaluate and improve ML models for tumor detection. |
| **System** | The primary application managing MRI uploads, processing, data storage, and report generation. |
| **MRI\_Scan** | Represents an MRI image file uploaded to the system for analysis. |
| **Report** | Document containing analysis results for an MRI scan, specifying tumor type and classification. |
| **Dataset** | A collection of multiple MRI scans uploaded by researchers for analysis and ML model evaluation. |
| **Machine Learning Model** | A machine learning algorithm used to classify MRI scans as malignant or benign. |
| **System Administrator** | A user managing platform configurations, monitoring security, and updating ML models. |

## 

**7.2 Association Definitions Table**

|  |  |
| --- | --- |
| **Association** | **Definition** |
| **User uploads MRI\_Scan** | Users (Patients, Doctors) upload MRI scans to the system for processing. |
| **System processes MRI\_Scan** | The system analyses uploaded MRI scans using a Machine Learning Model and generates a Report. |
| **Patient views Report** | Patients can view their MRI analysis report to understand diagnostic results. |
| **Doctor generates Report** | Doctors generate reports after MRI scan analysis and may recommend further action. |
| **Researcher evaluates Model** | Researchers upload datasets to evaluate the performance of Machine Learning models. |
| **System Administrator manages User** | System Administrators manage user accounts, roles, and ensure secure data handling. |
| **Dataset contains MRI\_Scan** | A Dataset is a collection that holds multiple MRI scans for batch processing. |
| **System generates Report** | The system creates a diagnostic report based on the analysis of an MRI scan using ML models. |

**7.3 Attribute Definitions Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Class/Entity** | **Attribute** | **Data Type** | **Definition** |
| **User** | userId | String | A unique identifier for each user in the system. |
|  | username | String | The login name for the user. |
|  | password | String | Password for user authentication. |
|  | role | Enum | Defines if the user is a Patient, Doctor, Researcher, or Administrator. |
| **Patient** | patientId | String | Unique identifier for a patient. |
|  | name | String | Full name of the patient. |
|  | age | Integer | Age of the patient. |
| **Doctor** | doctorId | String | Unique identifier for a doctor. |
|  | specialization | String | Area of medical expertise. |
| **Researcher** | researcherId | String | Unique identifier for a researcher. |
|  | affiliation | String | Organization or institution the researcher is associated with. |
| **System** | systemId | String | Unique identifier for the system instance. |
| **MRI\_Scan** | scanId | String | Unique identifier for an MRI scan. |
|  | filePath | String | Location of the MRI image file in the system. |
|  | uploadDate | DateTime | Timestamp of when the scan was uploaded. |
| **Report** | reportId | String | Unique identifier for a diagnostic report. |
|  | analysisDate | DateTime | Date and time when the analysis was conducted. |
|  | tumorType | Enum | Specifies if the tumor is benign or malignant. |
|  | size | String | Estimated size of the detected tumor. |
|  | location | String | Location of the detected tumor in the brain. |
|  | confidenceScore | Float | Probability score indicating the model’s confidence. |
| **Dataset** | datasetId | String | Unique identifier for a dataset. |
|  | datasetName | String | Name of the dataset uploaded by a researcher. |
|  | mriScans | List<MRI\_Scan> | A collection of MRI scans included in the dataset. |
| **Machine Learning Model** | modelId | String | Unique identifier for the machine learning model. |
|  | algorithm | String | The type of algorithm used (e.g., CNN, SVM). |
|  | accuracy | Float | Accuracy of the model in classifying tumors. |
|  | precision | Float | Precision metric of the model’s performance. |
|  | recall | Float | Recall metric of the model’s performance. |
| **System Administrator** | adminId | String | Unique identifier for a system administrator. |
|  | privileges | List<String> | Special access permissions or roles. |

## 7.4 System Operation Contracts

**Operation Contract 1: Upload MRI Scan for Tumor Detection**

**Operation:** uploadMRI\_Scan(user: MedicalProfessional, scan: MRI\_Scan)

**Cross reference:** Use Case Medical Professional Uploading MRI Scan

**Preconditions:**

The user is authenticated and logged into the system.

The MRI scan is in a valid format.

**Postconditions:**

The system stores the MRI scan.

The system invokes the machine learning model to analyze the scan.

A report is generated and linked to the MRI scan and the user.

**Operation Contract 2: Generate Tumor Detection Report**

**Operation:** generateReport(scan: MRI\_Scan)

**Crossreference:** Use Case Medical Professional Uploading MRI Scan

**Preconditions:**

The MRI scan has been successfully uploaded and processed by the machine learning model.

Postconditions:

The system creates a report with the tumor classification and medical recommendations.

The report is linked to the MRI scan and the medical professional who uploaded it.

**Operation Contract 3**: Upload Dataset for Research

**Operation:** uploadDataset(user: Researcher, dataset: MRI\_Scan[])

**Crossreference**: Use Case Researcher Testing ML Models

**Preconditions:**

The researcher is logged into the system with appropriate access rights.

The dataset is in an acceptable format and size.

**Postconditions:**

The system processes the dataset using various machine learning models.

Results are generated, including performance metrics for each model.

The researcher can review and compare model accuracy.

**Operation Contract 4: Manage User Accounts**

**Operation:** manageUserAccount(admin: SystemAdministrator, user: User)

**Crossreference:** Use Case System Administrator Managing Users

**Preconditions:**

The system administrator is authenticated and logged in.

The user to be managed (added, removed, updated) exists in the system.

**Postconditions:**

The user account is updated, deleted, or created as per the administrator’s request.

The changes are logged for auditing purposes.

1. **Interaction Diagrams**

**Interaction Diagrams for the Use Cases**

The interaction diagrams below provide a detailed flow of how the system interacts with the actors for each use case, incorporating the **Singleton** and **Observer Design Patterns**. These diagrams use **sequence diagrams** to model actor interactions with the system components.

**Patient Uploading MRI Scan for Preliminary Analysis**

**Diagram Description:**  
This interaction involves a patient uploading an MRI scan for analysis, receiving results (accuracy and AUC values), and logging out.

**Steps for Creating the Diagram:**

1. **Actors:** Include the **Patient** and **System**.
2. **Lifelines:** Represent patient, front-end interface, backend logic, CNN model, database (singleton), and observer for monitoring.
3. **Messages:**
   * Patient logs in.
   * System verifies login credentials.
   * Patient uploads an MRI scan.
   * System processes the scan using the CNN model.
   * System saves the scan and results in the database.
   * System returns results to the patient.
4. **Patterns:**
   * **Singleton Pattern:** Database connection ensures one instance handles all data operations.
   * **Observer Pattern:** Tracks and logs the patient’s upload and results access actions for security.
5. **Doctor Uploading and Viewing Patient's MRI Analysis**

**Diagram Description:**  
The doctor uploads a patient’s MRI scan, views analysis results, and downloads a detailed report.

**Steps for Creating the Diagram:**

1. **Actors:** Include the **Doctor** and **System**.
2. **Lifelines:** Represent the doctor, front-end interface, backend logic, CNN model, database (singleton), and observer.
3. **Messages:**
   * Doctor logs in.
   * System verifies credentials and retrieves the patient’s profile.
   * Doctor uploads the patient’s MRI scan.
   * System processes the scan using the CNN model.
   * System generates a report and stores it in the database.
   * Doctor downloads the report.
4. **Patterns:**
   * **Singleton Pattern:** Database ensures seamless data retrieval and storage.
   * **Observer Pattern:** Monitors actions like uploads and downloads for auditing.
5. **Researcher Uploading Dataset for Algorithm Comparison**

**Diagram Description:**  
The researcher uploads an MRI dataset to compare multiple algorithms, views analysis results, and logs off.

**Steps for Creating the Diagram:**

1. **Actors:** Include the **Researcher** and **System**.
2. **Lifelines:** Represent the researcher, front-end interface, backend logic, various ML models, database (singleton), and observer.
3. **Messages:**
   * Researcher logs in.
   * System verifies credentials.
   * Researcher uploads an MRI dataset.
   * System processes the dataset using multiple ML algorithms.
   * System generates and stores comparative results in the database.
   * Researcher views and downloads results.
4. **Patterns:**
   * **Singleton Pattern:** Database handles secure storage of datasets and results.
   * **Observer Pattern:** Logs researcher’s dataset uploads and analysis actions.
5. **Doctor Managing Multiple Patients' MRI Reports**

**Diagram Description:**  
The doctor accesses and manages multiple patients' MRI reports, views historical data, and makes updates as needed.

**Steps for Creating the Diagram:**

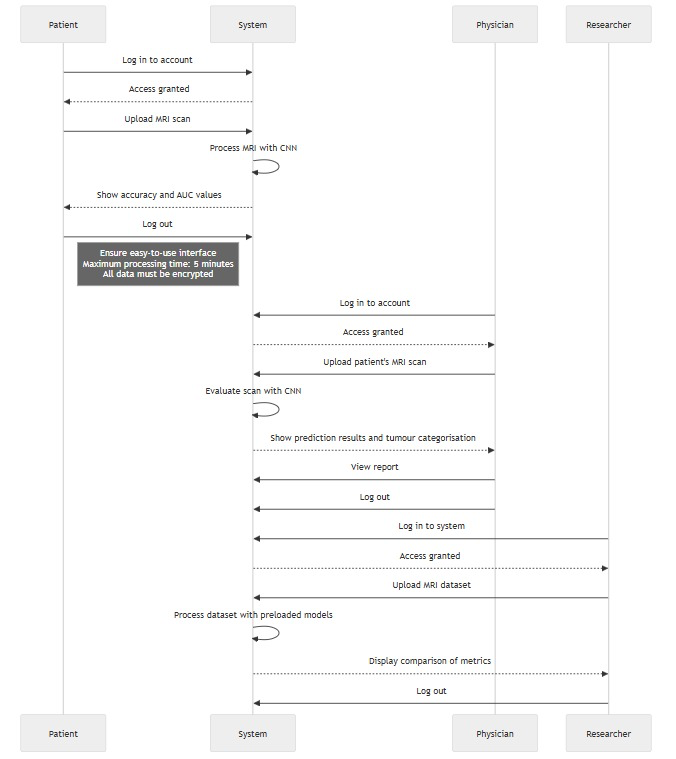
1. **Actors:** Include the **Doctor** and **System**.
2. **Lifelines:** Represent the doctor, front-end interface, backend logic, database (singleton), and observer.
3. **Messages:**
   * Doctor logs in.
   * System verifies credentials.
   * Doctor retrieves the patient list.
   * Doctor selects a patient and views MRI reports.
   * Doctor uploads new scans or updates patient details.
   * System saves the updated information and logs the actions.
4. **Patterns:**
   * **Singleton Pattern:** Database ensures quick retrieval and updates of patient data.
   * **Observer Pattern:** Logs and monitors all access and modifications for compliance.

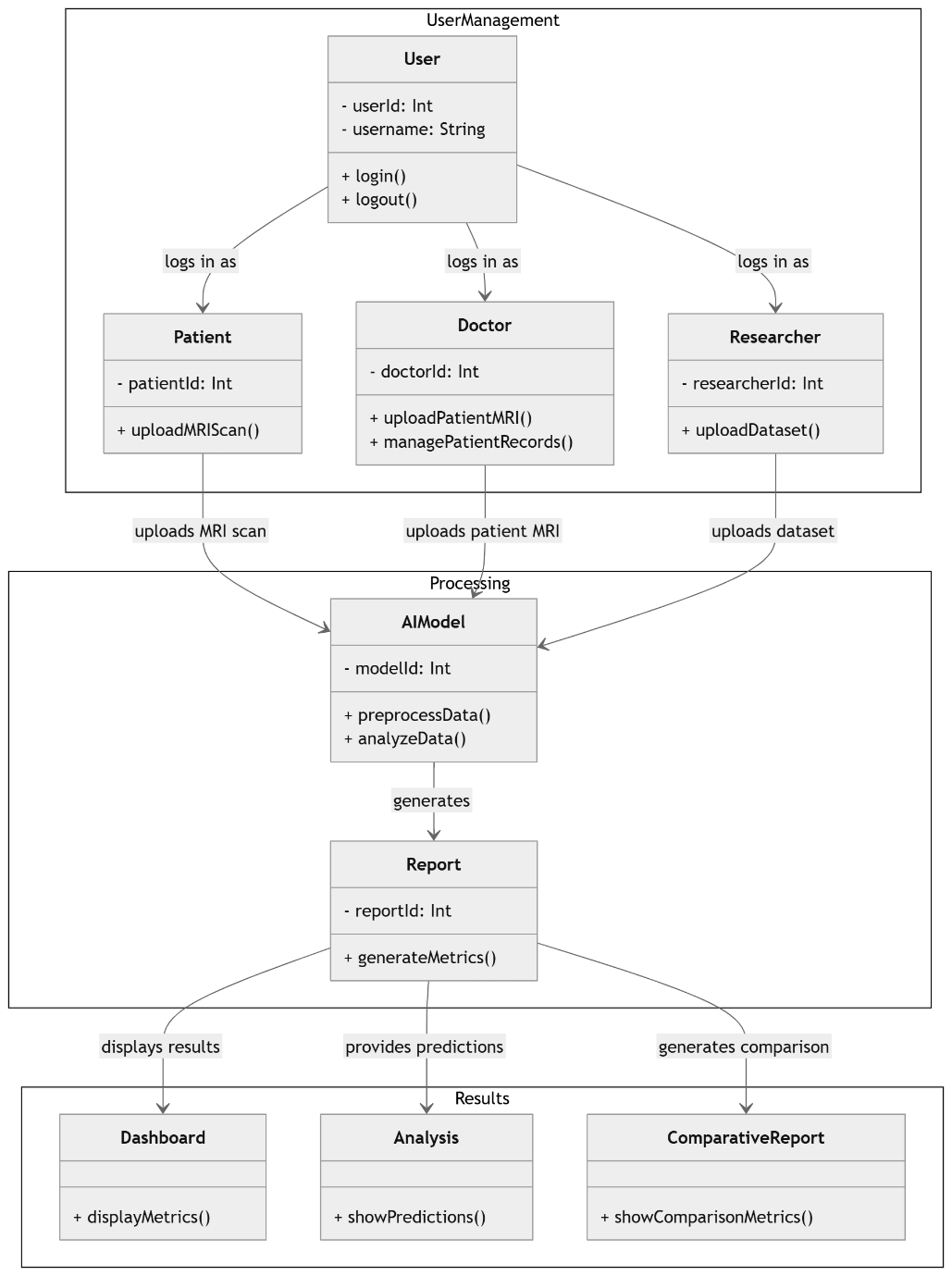
**Justification for Design Patterns:**

1. **Singleton Pattern:**
   * Simplifies management of database connections, ensuring that all data transactions occur through a single instance.
   * Reduces overhead and ensures thread safety, making the system scalable for concurrent users.
2. **Observer Pattern:**
   * Provides real-time monitoring of critical user actions such as logins, uploads, and report downloads.
   * Enhances security by logging these events for audits and compliance with regulations like .
3. **Class Diagram and Interface Specification**

**Flow of Events**

* **Login:**  
  The user (patient, doctor, or researcher) opens the system and logs into their account.
* **Account Type Identification:**  
  Based on the account type (patient, doctor, or researcher), the system displays different interfaces and functionalities.
* **Uploading MRI Scan or Dataset:**
* **For Patients:** The patient accesses the page to upload their MRI scan.
* **For Doctors:** The doctor accesses the patient information page, where they can upload an MRI scan for a specific patient.
* **For Researchers:** The researcher uploads a dataset of MRI scans for comparative analysis.
* **Preprocessing with AI Models (CNN and others):**
* The system uses a pretrained CNN model to process the MRI scan (or dataset) and analyze it.
* For patients: the system shows metrics such as accuracy and AUC (Area Under the Curve).
* For doctors: the system analyzes the scan to detect potential conditions (such as tumour type) and displays the result with prediction details.
* For researchers: the system compares multiple algorithms (including CNN) and generates a report with comparison metrics (precision, confusion matrix, etc.).
* **Display of Results:**
* **For Patients:** The dashboard shows the analysis results, including relevant metrics.
* **For Doctors:** A static message, prediction, and MRI analysis are displayed for the doctor’s review, along with historical patient data.
* **For Researchers:** A comparative report based on different models is shown, highlighting performance metrics.
* **Managing MRI Reports and Historical Data (Doctors):**
* Doctors can manage multiple patient records, view MRI reports, and track patient progress over time.
* Doctors can upload additional scans or modify patient information.
* **Logout:**  
  After viewing or analyzing the results, the user logs out of the system.





1. **System Architecture and System Design**

A three-tier architecture is used by the Brain Tumor Detection Website to guarantee maintainability, scalability, and modularity. The structure, the tools, and their interactions are described in detail below:

**10.1 System Architecture: 3-Tier Architecture**

The application itself is divided into three layers by the system's adherence to the 3-tier design pattern:

* Presentation Layer (UI)
* Application Layer (Business Logic)
* Data Layer (Database)

## 10.1.1 Presentation Layer (UI)

This is the user interface for the application. React.js, a JavaScript library for making responsive and changing user interfaces, is used in its construction.

* **Function**: Responds to user input and displays findings to clients (researchers, physicians, and patients).
* **Key Features:**

User Registration & Login: Patients, doc’s, and researchers can register and log in safe and securely.

MRI Uploading: Interface for uploading MRI scans (PNG, JPEG, and DICOM).

Results Display: Displays the tumour classification (malignant/benign), accuracy, and AUC values after processing.

Accessibility: The interface is intended to be user-friendly, allowing even non-technical users, such as patients, to easily navigate the platform.

## 10.1.2 Application Layer (Backend/Business Logic)

The backend is built with Python Flask, a lightweight web framework. The business logic is where all of the core processing occurs, including AI-based tumour detection and data handling.

* Role: Manages UI requests, runs AI algorithms on MRI scans, interacts with the database, and returns results to the UI.
* **Key Components:**

AI (CNN Model): The uploaded MRI images are processed using convolutional neural networks (CNNs). The CNN model is trained for image classification with TensorFlow, determining whether the tumour is benign or malignant.

Data Security: Provides secure data transmission and storage. All sensitive data, such as MRI scans and patient information, is encrypted with TLS/SSL protocols.

Role-based Access Control (RBAC): Ensures that the right users (doctors, patients, researchers) have the appropriate permissions to view or upload data.

## 10.1.3 Data Layer (Database)

The data layer stores all application data, such as user information, MRI scan data, processed results, and logs.

* Role: Provides storage and retrieval goods and services for all application data.
* Technologies:

MongoDB (NoSQL Database): MongoDB is used to store large amounts of unstructured data, such as MRI scan images and metadata. Its adaptable schema allows for simple storage and retrieval of MRI scans in a variety of formats.   
SQL Database: An SQL database, such as PostgreSQL or MySQL, stores relational data like user accounts, reports, and model performance metrics (such as AUC and accuracy). These databases offer structured data storage and efficient querying capabilities.

Data Storage Breakdown:

* MongoDB: Stores MRI scan files (image files), user activity logs, and other unstructured data.
* SQL Database: Structured data is stored, including user details, session information, processed results (such as tumour classification and model metrics), and historical reports.

## 10.2 Technologies Used

The following technologies are integrated into the system design for seamless operation:

## 10.2.1 Backend Technologies

* Python Flask: Flask is used for backend development because it is simple and scalable. It accepts HTTP requests, processes data, and communicates with the frontend and database. Flask's minimalistic approach allows for easy integration of machine learning models, user authentication, and API creation.
* **Features:**

Handles the routing of user requests and responses.   
Integrates with machine learning models to detect tumours.  
Connects to MongoDB and SQL databases to retrieve and store information.

* TensorFlow (CNN Models): TensorFlow is an open-source machine learning library used to create and train Convolutional Neural Networks (CNNs) that process MRI images. CNN is the best model for image classification tasks because it can detect spatial hierarchies in images.

**CNN Workflow:**

**Preprocessing:** MRI scans are resized and normalized.

**Training:** The CNN is trained using labeled MRI images (benign or malignant).

**Inference:** The trained CNN model is used to divide uploaded MRI scans into benign and malignant categories.

**MongoDB (NoSQL Database):** MongoDB is a NoSQL database that allows you to store unstructured data such as MRI images and metadata. It supports fast read and write operations, making it ideal for handling large image files.

**Role in the System:**

Stores MRI images and associated metadata.   
Offers a flexible structure that can accommodate a variety of image formats and other unstructured data.

**SQL Database (e.g., PostgreSQL/MySQL):** a relational database that houses structured data, including reports, metadata, and user credentials. For managing structured data with distinct relationships between entities (like users, reports, and models), SQL databases are perfect.

**Role in the System:**

Information about patients and users is stored here.   
Tracks processed results, such as tumour classifications, model accuracy, and historical reports.

**10.2.2 Frontend Technologies**

**React.js**: The application's frontend is constructed with the help of the robust JavaScript library React. It is perfect for displaying MRI results of analysis and interacting with backend services given its interactive and dynamic user interface, which updates in real time.

**Features:**

Efficient rendering of dynamic content, such as real-time tumour classification outcomes.   
Supports single-page application (SPA) architecture, which allows for smooth navigation without the need for full-page reloads.  
Integrates with Flask via RESTful APIs to send and receive data.

**10.2.3 Integration Between Frontend and Backend**

* **RESTful API (Flask and React):** RESTful APIs are used for communication between the Flask backend and React.js frontend. These APIs allow MRI scan data to be sent from the frontend to the backend, allowing machine learning algorithms to process it and produce results.

**10.2.4 Security Measures**

**•** Encryption: SSL/TLS is used to encrypt all sensitive data, including user information and MRI images, before it is stored in the database.   
• Authentication: Using user authentication techniques like JWT (JSON Web Tokens), the system protects platform access.  
• Role-Based Access Control (RBAC): Makes sure that only authorized users, such as researchers and doctors, can upload MRI scans or access sensitive data.

**10.3. System Design**

The system is designed to ensure scalability, security, and efficiency. Here's how the components interact within the architecture:

## Frontend (UI Layer - React.js):

The user interacts with the platform through its React-based frontend.   
Users can upload MRI scans, view the results, and access reports. The frontend communicates with the backend via AJAX and RESTful APIs.

## Backend (Application Layer - Flask, CNN, TensorFlow):

The backend receives UI requests and uses TensorFlow's CNN models to process MRI scans.   
The Flask application acts as a middleware, handling requests and responses between the UI and the database, as well as orchestrating interactions with the CNN model.

## Data Layer (Database Layer - MongoDB, SQL):

SQL databases store structured data (user information, results, and reports), whereas MongoDB stores unstructured data (MRI images).   
The database layer ensures that both types of data are efficiently managed and retrieved in response to user requests.

## Machine Learning (CNN - TensorFlow):

The CNN model is used to process MRI images. The model is trained on a labelled dataset before being used for inference (tumour classification in new MRI scans).   
TensorFlow enables efficient model training and deployment.

* **Observer Pattern (for Real-Time Monitoring and Logging):**

The system monitors actions like uploads, login attempts, and report views for auditing and security purposes.

1. **Algorithms and Data Structures**

The Brain Tumour Detection Website employs a variety of data structure and algorithms to efficiently process MR imaging images, handle information and provide results. The backend data structures and algorithms that support efficient processing and retention of information are discussed in detail below.

**11.1 Algorithms Used**

**11.1.1 Convolutional Neural Network (CNN)**

The Convolutional Neural Network (CNN employed in the background to analyse MRI images and classify brain tumours. CNNs are extremely efficient in image processing because they can detect features such as edges, forms, along with textures directly from the beginning image data, eliminating the need for the manually extraction of features.

* **Role:** Classify MRI scans as benign or malignant based on the patterns found in the images.
* **How CNN Works:**
  1. **Convolutional Layer:**

The convolutional layer uses filters (kernels) to detect patterns in the input MRI image, including edges and textures.

* 1. **Activation Function (ReLU):**

The Rectified Linear Unit (ReLU) function offers nonlinearity, allowing CNNs to learn complicated shapes from images.

* 1. **Pooling Layer:**

Pooling reduces image dimensions while retaining key features. Max pooling is commonly used to extract the most important features.

* 1. **Fully Connected Layer:**

The fully connected layer connects all neurones and contributes to the final classification decision (benign or malignant).

* 1. **Output Layer:**

The output layer typically assigns probabilities to possible classes (benign/malignant) using a softmax activation function.

* **Training Process:**

The CNN is trained using a labelled collection of MRI scans, with each image classified as benign or malignant. During this step, the CNN discovers to connect imagery characteristics with the appropriate class labels. The model's performance is measured using metrics such as accuracy, precision, recall, along with AUC, which show how well the model distinguishes between healthy and dangerous tumours.

**11.1.2 Data Preprocessing Algorithms**

Before feeding MRI scans into the CNN model, data preprocessing is necessary to ensure that the images are in a suitable format and quality for analysis:

* **Resizing:** MRI scans are frequently resized to a standard dimension to reduce computational complexity and ensure consistency in the model's input.
* **Normalization:** The pixel values of the MRI images are normalised to scale them between 0 and 1, accelerating the CNN's convergence during training.
* **Augmentation:** Rotation, flipping, and zooming are techniques used to artificially increase dataset size and improve model generalisation.

**11.1.3 Performance Metrics for CNN**

Accuracy refers to how many MRI scans are correctly classified by the model. Recall and precision aid in determining the model's ability to correctly recognise benign and malignant tumours, particularly when a particular type is more common instead of the other. AUC (Area Under the Curve) measures how well the model distinguishes between the two classifications, with higher AUC suggesting better performance. Simply put, accuracy represents the total amount of accurately recognised MRI images determined by the model.

**11.2. Data Structures Used**

**11.2.1 Arrays (for Image Representation)**

**Purpose:** Arrays are used to represent the pixels in MRI images in a format that the CNN can understand. Each MRI image has been turned into an array with multiple dimensions (tensor), with each element representing a pixel value at the image.

**Example:** A greyscale image with a resolution of 512x512 pixels is represented by a 2D array of dimensions [512, 512]. A colour image (RGB) is represented as a 3D array with dimensions [512, 512, 3].

**11.2.2 Linked Lists (for Dynamic Data Storage)**

**Purpose:** Linked lists are used in scenarios where dynamic data structures need to be maintained, such as managing patient records or keeping track of recent actions.

**Usage Example:** Each patient’s MRI scan history or report history can be stored as a linked list, where each node contains a report or MRI scan reference.

**11.2.3 Trees (for Hierarchical Data Storage)**

**Purpose:** Trees are used to store hierarchical data, such as patient reports and their associated MRI scans. This allows efficient retrieval of MRI scan information based on patient ID or report details.

**Usage Example:** A binary search tree might be used to store information on patients, allowing for quick retrieval and searching based on the patient's ID or name.

**Application Example:**

The tree's root could represent the first patient, while the child nodes could represent the patient's MRI scan or reports.

**11.2.4 Hash Tables (for Quick Lookups)**

**Purpose:** Hash tables are used to store data that must be retrieved quickly, such as session information, user credentials, or tumour classification results.

**Usage Example:** A hash table could store user sessions using the session ID as the key and the associated user data (login credentials, last accessed page) as the value.

**11.3. Data Flow and Integration with Algorithms**

* **Input Flow:** MRI images are initially uploaded using the front-end interface. The images are then preprocessed (resized, normalised) and transformed into arrays.
* **Processing Flow:** The preprocessed images are fed into the CNN model, which classifies tumours using the learnt features.
* **Output Flow:** The results (tumor classification, AUC, accuracy) are stored in the database and displayed on the front-end interface for the user.

**11.4. Data Storage in MongoDB and SQL**

* **MongoDB (NoSQL):**

Stores unstructured data such as MRI images (as binary data) and patient records.

Allows for flexible schema design to accommodate various image formats (JPEG, PNG, DICOM).

* **SQL Database (PostgreSQL/MySQL):**

Stores structured data, such as patient profiles, model performance metrics (e.g., AUC, accuracy), and processed results.

Supports relational queries for quick retrieval of patient information, report generation, and tracking model performance over time.

**11.5. Scalability and Efficiency Considerations**

* **Scalability of CNN:**

The CNN is trained on a scalable cloud infrastructure, using **GPU acceleration** to handle large datasets and improve training time.

The model can be retrained periodically to accommodate new data and improve classification accuracy.

* **Efficient Data Storage:**

MongoDB provides fast retrieval of MRI images by using indexing mechanisms, ensuring low latency for data access.

SQL databases allow for efficient querying of structured data (e.g., patient information, tumor classifications).

* **Batch Processing for Large Datasets:**

The system supports batch processing of MRI datasets uploaded by researchers for algorithm comparison. This helps scale the processing and evaluation across multiple models without compromising performance.

1. **User Interface Design and Implementation Details  
   12.1. Preliminary Design**

The system is designed with three main actors: Patient, Doctor, and ML Researcher. Each actor has a specific navigational path tailored to their unique interactions with the system. Below is a detailed description of these paths:

**First Page User Signup and Login**

The initial landing page features a Signup and Login form. The signup form includes a mandatory field to capture the role of the user (Patient, Doctor, or ML Researcher). Based on the selected role, the system directs the user to their respective dashboard upon successful login.

**Patient Interaction Flow**

Upon logging in as a Patient, the user is navigated to the MRI Upload and Results Page.

Here, the Patient uploads an MRI scan image. The uploaded image is securely stored in the database.

The system then obtains the MRI image and runs it by means of an ML algorithm with a CNN (Convolutional Neural Network) model.

When the analysis is finished, the results—such as AUC (Area Under the Curve) and accuracy metrics—are displayed on the the client interface.

After reviewing the results, the patient can log out.

**Doctor Interaction Flow**

Upon logging in as a Doctor, the user is directed to a page designed for managing patient information.

The Doctor inputs details about the patient and uploads the corresponding MRI scan image.

This data, once submitted, is saved in the database and then retrieved for further processing by the CNN model.

The frontend displays the prediction results along with a static message specifying the type of MRI analysis.

After the necessary review, the Doctor can log out of the system.

**ML Researcher Interaction Flow**

Upon logging in as an ML Researcher, the user is navigated to an advanced results page.

The Researcher uploads MRI scan images, which are then processed by multiple ML algorithms, including CNN.

The front end presents a detailed analysis, including metrics from various algorithms, convolution matrices, precision values, and other advanced results.

After completing the research activities, the ML Researcher can log out of the system.

12.2. Core Backend Implementation

Backend Development:

a. To facilitate MRI file uploads, safe data storage, and machine learning model integration, the group will create the backend infrastructure.   
b. The backend will manage secure file transfers, user authentication, and systemwide data privacy protection.

Machine Learning Model Integration:

a.Integrating machine learning models into the system to evaluate uploaded MRI images is the main objective. This entails configuring an API or integrating TensorFlow/Kerasbased models that have already undergone tumour detection training directly.   
b. The machine learning models will receive the MRI images correctly, process them, and provide the findings to the user thanks to the backend.

Frontend Development

Connecting the Frontend to the Backend:

a. To enable user interaction with the system, the frontend and backend will be linked. This entails turning on secure login, uploading scans, and showing the findings (reports).   
b. The development of the user interface will concentrate on making the system easy to use for patients and medical professionals, with a special focus on accessibility.

Developing Reports:

Users will see reports produced by the machine learning models in an easy-to-understand style. Tumour categorisation (malignant or benign) and any pertinent suggestions will be included in the reports.

1. **Summary of Changes**

**Key Revisions:**

1. Updated **System Sequence Diagrams** to reflect final implementation.
2. Enhanced **Class Diagram** with new attributes and methods.
3. Added specific **design patterns** like Singleton for database connection and Observer for monitoring system events.
4. Elaborated **Nonfunctional Requirements** with performance benchmarks.
5. Revised **Use Cases** to align with demo-ready features.
6. Integrated **Kaggle Dataset** details for user interface demonstration.
7. **Unique Capability**

**14.1 Feature: Secure Data Handling**

Secure data handling is a critical component of any healthcare system, especially one that deals with sensitive information such as MRI scans and patient records. In the **Brain Tumor Detection Website**, secure data handling ensures that all data is protected from unauthorized access, tampering, and breaches.

To ensure the security of the system, two key techniques are implemented:

**Implementation:**

**1. Role-Based Access Control (RBAC)**

**Role-Based Access Control (RBAC)** is a security model that restricts system access based on the user's role within an organization. In the **Brain Tumor Detection Website**, RBAC ensures that users can only access data and perform actions that are relevant to their role. This minimizes the risk of unauthorized access to sensitive information.

**How it Works:**

**Roles:**

**Patients:** Can upload their MRI scans, view their results, and access historical reports. They cannot access or modify any data related to other users.

**Doctors:** Can upload patient MRI scans, view patient reports, generate new reports, and make updates. They have more access than patients but are restricted to their patient records.

**Researchers:** Can upload datasets for model testing, compare machine learning algorithms, and analyze model performance. They cannot access patient-specific data or view individual reports.

**Administrators:** Have full access to the system, allowing them to manage user accounts, update roles, and monitor the system's overall health. They are responsible for ensuring that the platform adheres to security protocols.

* **Access Control:**

Each user is assigned a role during the registration process. Based on this role, the system enforces access restrictions. For instance, a doctor cannot access a researcher’s dataset, and a patient cannot see another patient's MRI scan.

**Enforcement:** Each system function (e.g., viewing reports, uploading scans) is linked to a role. The system checks the role before allowing any operation.

**Benefits:**

* **Reduced Risk:** By restricting access based on roles, RBAC reduces the chance of unauthorized access to sensitive medical data.
* **Compliance:** Ensures the system meets regulatory standards like , which mandate that only authorized individuals can access sensitive healthcare data.

**14.2. Encryption**

**Encryption** is the process of converting data into a secure format that cannot be read by unauthorized users. This is essential for protecting sensitive data, especially during transmission and storage, such as patient information and MRI scan files.

**How it Works:**

* **Data Encryption at Rest (Storage Encryption):**

All sensitive data, such as MRI scan files and patient reports, is encrypted when stored in the database (both MongoDB and SQL databases). Even if a malicious actor gains access to the database, the data remains unreadable without the decryption key.

The system uses **AES (Advanced Encryption Standard)** with a strong key size (e.g., 256-bit) for encryption to ensure that the data is secure.

* **Data Encryption in Transit (Transmission Encryption):**

When users upload MRI scans or download reports, the data is transmitted securely over HTTPS, which uses **TLS (Transport Layer Security)**. TLS ensures that the data sent between the client (e.g., the patient’s browser) and the server is encrypted, preventing interception by third parties (e.g., during a man-in-the-middle attack).

* **End-to-End Encryption:**

For particularly sensitive data (such as patient-specific data), the system could employ **end-to-end encryption**, meaning data is encrypted at the source (e.g., the patient’s device) and only decrypted when it reaches the intended recipient (e.g., the doctor or system administrator).

**Overall Security Flow:**

1. **User Authentication:** When a user logs in (whether a patient, doctor, researcher, or administrator), their credentials are checked against the database. Only authenticated users are allowed to proceed to the next step.
2. **Role Assignment:** Based on the user’s role, the system applies the appropriate access controls. Each user is granted access to certain features of the platform based on their role.
3. **Secure Data Transmission:** When data (such as MRI scans or reports) is transmitted between the client (browser) and the server, it is encrypted using HTTPS (TLS). This protects the data during transit.
4. **Secure Data Storage:** All sensitive data stored in MongoDB or SQL databases is encrypted using AES. This ensures that even if the data is accessed by unauthorized users, it cannot be read without the proper decryption key.
5. **Monitoring and Logging:** The system continuously monitors user actions, especially access to sensitive data. This monitoring is logged, and **Observer Pattern** ensures that any suspicious activity is detected and flagged in real-time.
6. **Conclusions and Future Work**

**15.1 Technical Challenges Encountered**

Throughout the development of the **Brain Tumor Detection Website**, several technical challenges arose, requiring careful consideration and innovative solutions. Below are the key challenges encountered:

1. **Data Preprocessing and Model Integration:**

**Challenge:** Preparing MRI scan images for analysis by the Convolutional Neural Network (CNN) proved to be complex. The original dataset had variations in image formats, sizes, and quality, which made standardization and preprocessing essential before feeding the data into the model.

**Solution:** We employed techniques such as resizing, normalization, and augmentation to standardize the images. Additionally, using a CNN architecture required careful tuning to ensure optimal performance. Selecting the right parameters (like learning rate, epochs) and avoiding overfitting was an ongoing challenge.

1. **AI Model Accuracy:**

**Challenge:** Ensuring the accuracy of the CNN model for classifying MRI scans as benign or malignant required extensive training and testing. With limited labeled data available, it was difficult to train a robust model that generalized well on unseen data.

**Solution:** We used **transfer learning**, where we fine-tuned pre-trained CNN models (such as VGG16 or ResNet) to improve accuracy. This allowed us to leverage a large dataset that had already been trained on image classification tasks, reducing the need for an extensive dataset of our own.

1. **Handling Large Data Volumes:**

**Challenge:** The system needed to efficiently handle large image files and multiple simultaneous requests from different users, such as patients, doctors, and researchers.

**Solution:** To address this, we implemented **asynchronous processing** and utilized **batch processing** techniques. Images were uploaded in batches and processed in the background, preventing delays in the user interface and ensuring that large datasets from researchers could be handled without affecting the system’s performance.

1. **Data Security and Privacy:**

**Challenge:** Handling sensitive patient data, such as MRI scans and medical reports, posed a significant challenge in terms of ensuring data security and compliance with data protection.

**Solution:** We implemented **role-based access control (RBAC)**, ensuring that only authorized users had access to specific parts of the platform. Additionally, all data was encrypted both in transit and at rest, and access logs were maintained to monitor and audit system activity.

1. **Integrating Machine Learning with the Web Application:**

**Challenge:** Integrating machine learning models into a web application created its own set of issues, particularly around managing the flow of data between the front-end (React.js) and the back-end (Python Flask).

**Solution:** We used **RESTful APIs** to connect the front-end and back-end. The backend handled the image processing, and the results were returned to the front-end via these APIs. This allowed the web interface to remain responsive and fast while handling complex tasks on the server side.

**15.2 How Software Engineering Techniques Addressed These Challenges**

The software engineering techniques we learned throughout the course, including **UML Modeling & Object-Oriented Design**, **Requirements Engineering**, **System Modeling**, **Architectural Design**, **JUnit Testing**, and **Design and Implementation**, played a crucial role in addressing the technical challenges outlined above. Here’s how these techniques contributed to the success of the project:

1. **UML Modeling & Object-Oriented Design:**

**Challenge:** Designing the system architecture and ensuring that the components interacted efficiently.

**Solution:** We used **UML (Unified Modeling Language)** diagrams to model the system's components and their interactions. **Class diagrams**, **sequence diagrams**, and **use case diagrams** were created to clearly define the relationships between the actors (patients, doctors, researchers) and the system. This provided a clear blueprint of the system, which helped in object-oriented design and modular implementation.

1. **Requirements Engineering:**

**Challenge:** Gathering and documenting clear functional and non-functional requirements for the system, ensuring that both user needs and system constraints were met.

**Solution:** Through **requirements engineering**, we conducted multiple iterations of discussions with stakeholders (doctors, patients, and researchers) to ensure that the system met their needs. We defined use cases for different user types and ensured that the platform was designed to be scalable, efficient, and secure.

1. **System Modeling and Architectural Design:**

**Challenge:** Designing the system to ensure smooth communication between the front-end, back-end, and database layers.

**Solution:** We followed a **3-tier architecture** approach to separate concerns between the **UI layer**, **application layer**, and **data layer**. This allowed us to manage the interactions between the user interface (React.js), the backend logic (Flask), and the databases (MongoDB and SQL) effectively. This architecture facilitated easy scalability and maintainability.

1. **JUnit Testing:**

**Challenge:** Ensuring that the backend algorithms and APIs were functioning as expected, particularly with large-scale data processing.

**Solution:** We used **JUnit testing** and wrote unit tests for backend components (such as the machine learning models, image processing functions, and API endpoints). This helped us catch bugs early and ensure that each component worked independently before integrating them into the full system.

1. **Design and Implementation:**

**Challenge:** Designing a system that was both functional and user-friendly.

**Solution:** We followed **object-oriented design principles** to ensure that the codebase was modular and reusable. We also employed **agile practices**, iterating on design and development to improve the platform based on continuous feedback. This helped us balance user needs with technical feasibility.

**15.3 Other Knowledge That Could Have Helped**

While the techniques learned in this course were instrumental in overcoming many challenges, additional knowledge in the following areas could have further improved the project:

1. **Cloud Deployment and DevOps Practices:**

**Challenge:** Scaling the system to handle increased traffic and large datasets.

**Solution:** More experience with cloud services (e.g., **AWS**, **Google Cloud Platform**) and **DevOps** practices (e.g., continuous integration/continuous deployment) would have helped automate deployment processes and manage scalability more efficiently.

1. **Advanced Machine Learning Techniques:**

**Challenge:** Improving the CNN model’s accuracy and generalizability.

**Solution:** Additional knowledge in **deep learning techniques** like **transfer learning** and **hyperparameter optimization** could have helped achieve better results. We could have explored more advanced architectures, such as **ResNet** or **DenseNet**, to improve the model's performance.

**15.4 Future Work and Directions for Improvement**

While the Brain Tumour Detection Website is functional and provides reliable tumor detection results, there are several areas for improvement and future work

* **Increasing Data Volume for Model Training:**

**Challenge:** Our model’s accuracy is limited by the amount of training data available.

**Future Work:** To improve the model's accuracy, we can expand the training dataset by incorporating more diverse MRI scans. Techniques like adding noise, rotation, and flipping can artificially increase the dataset size. Additionally, collaborating with medical institutions to acquire more labeled MRI scans can significantly enhance the model's performance.

* **Real-Time Processing:**

**Challenge:** The system processes MRI scans asynchronously, which can lead to delays.

Future Work: To improve real-time capabilities, the system could be optimized for real-time MRI scan processing, especially for emergency situations. Using GPU acceleration for faster processing would significantly reduce latency.

**Advanced User Features:**

**Challenge:** Current user features are basic, focusing primarily on uploading and viewing MRI scans.

**Future Work:** Adding features such as predictive analytics, personalized tumor monitoring, and the ability to track historical data would enhance user experience for both patients and doctors. Integrating a chatbot or virtual assistant could also help users interact with the system more effectively.

* **Deployment and Scalability:**

**Challenge:** The current deployment setup may not be able to handle large-scale usage.

**Future Work:** Moving the system to a cloud-based infrastructure (such as AWS or Google Cloud), with auto-scaling capabilities, would improve its ability to handle high volumes of user requests and MRI data uploads.

**15. References**

**Rajaraman, V., & Gaur, M.** (2019). Machine learning for brain tumor detection. *PMC*. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6562394/>

**Topol, E. J.** (2019). *Deep medicine: How artificial intelligence can make healthcare human again*. Basic Books.

**Litjens, G., et al.** (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42, 6088. <https://doi.org/10.1016/j.media.2017.07.005>

**Chollet, F.** (2018). *Deep learning with Python*. Manning Publications.

**Bashir, A., & Raza, M.** (2022). Brain tumor classification using machine learning: A comprehensive review. *Journal of Biomedical Informatics*, 126, 103984. <https://doi.org/10.1016/j.jbi.2021.103984>

**Khan, M. A., et al.** (2021). A review on machine learning techniques for brain tumor detection in MRI images. *Artificial Intelligence Review*, 54(2), 12311262. [https://doi.org/10.1007/s10462020098091](https://doi.org/10.1007/s10462-020-09809-1)

**Menze, B. H., et al**. (2015). The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE Transactions on Medical Imaging, 34*(10), 1993-2024. <https://doi.org/10.1109/TMI.2014.2377694>

**Pereira, S., et al.** (2016). Brain tumor segmentation using convolutional neural networks in MRI images. *IEEE Transactions on Medical Imaging, 35*(5), 1240-1251. <https://doi.org/10.1109/TMI.2016.2538465>

**Kamnitsas, K., et al.** (2017). Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. *Medical Image Analysis, 36*, 61-78. <https://doi.org/10.1016/j.media.2016.10.004>

**Suzuki, K.** (2017). Overview of deep learning in medical imaging. *Radiological Physics and Technology, 10*(3), 257-273. <https://doi.org/10.1007/s12194-017-0406-7>

**Aderghal, K., et al.** (2018). Classification of Alzheimer’s disease with MRI using convolutional neural networks. *IEEE 15th International Symposium on Biomedical Imaging (ISBI)*, 514-517. <https://doi.org/10.1109/ISBI.2018.8363647>

**Sommerville, I**. (2015). *Software engineering* (10th ed.). Pearson Education.

**Boehm, B. W., & Turner, R.** (2004). *Balancing agility and discipline: A guide for the perplexed*. Addison-Wesley.

**Garlan, D., & Shaw, M.** (1993). An introduction to software architecture. *Advances in Software Engineering and Knowledge Engineering, 1*, 1-39. <https://doi.org/10.1142/9789812798039_0002>

**Bass, L., Clements, P., & Kazman, R**. (2012). *Software architecture in practice* (3rd ed.). Addison-Wesley

**He, K., et al.** (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778. <https://doi.org/10.1109/CVPR.2016.90>

**Szegedy, C., et al.** (2015). Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1-9. <https://doi.org/10.1109/CVPR.2015.7298594>

**Simonyan, K., & Zisserman, A.** (2015). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. <https://arxiv.org/abs/1409.1556>