****Software Requirement Specifications*****

TumorTech AI-powered Brain Tumor Analysis

With Predictive Algorithms :

Submitted by

Bushra Aslam

(F22BINFT1E02075)

7th semester 1E

Submitted to

Ma'am Sara Fareed

Department of Information Technology

The Islamia University of Bahawalpur

Software Requirements Specification

Introduction

The "Brain Tumor Detection Using MRI Images" project aims to develop a sophisticated, AI-powered system that automates the detection and classification of brain tumors from MRI scans. Leveraging Convolutional Neural Networks (CNNs), a form of deep learning, the project seeks to enhance the accuracy and efficiency of brain tumor diagnosis, reducing reliance on manual interpretation by radiologists. The system will classify tumors into key categories—such as glioma, meningioma, and pituitary adenomas—while distinguishing healthy brain tissues. This project addresses the limitations of traditional diagnostic methods by providing a non-invasive, automated solution capable of improving early detection, lowering diagnostic costs, and supporting timely medical interventions. Through the use of advanced machine learning techniques, the project is designed to assist healthcare professionals in making faster, more reliable diagnoses, ultimately improving patient outcomes.

Project Goals & Objectives

Goals

- Develop an AI-based system for accurate tumor detection in MRI and CT scans.
- Enhance diagnostic accuracy to improve patient outcomes and healthcare efficiency.
- Provide an intuitive user interface for healthcare professionals to easily upload and analyze medical images.

Objectives

- Preprocess medical images to prepare them for deep learning algorithms.
- Train a deep learning model for tumor classification using labeled datasets.
- Create a user-friendly web application for image uploads and result retrieval.
- Enable quick and accurate diagnostic results to support timely treatment decisions.

Software Requirements Specifications

Functional Requirements

No.	Requirement	Description
FR1	**Image Upload**	• Users can upload MRI or CT scan images through a user-friendly web interface. • This functionality ensures a seamless and intuitive process for healthcare professionals to submit images for analysis. • The system should support standard medical image formats (e.g., DICOM, PNG, JPEG) and ensure secure file transmission to protect patient confidentiality.
FR2	**Image Preprocessing**	• The uploaded images undergo validation to check for compatibility, size, and format. After validation, the system normalizes these images to ensure consistency in quality and format. • This preprocessing step may include resizing, color normalization, noise reduction, and data augmentation to prepare the images for accurate tumor detection by the deep learning model.
FR3	**Tumor Detection**	• The system employs a pre-trained deep learning model to analyze the uploaded images and detect tumor regions. • The model identifies and highlights potential tumor areas, providing a confidence score for its findings. • This functionality aims to assist healthcare professionals in making precise and timely diagnoses.
FR4	**Report Generation**	• The system employs a pre-trained deep learning model to analyze the uploaded images and detect tumor regions. • The model identifies and highlights potential tumor areas, providing a confidence score for its findings. • This functionality aims to assist healthcare professionals in making precise and timely diagnoses.

FR6	**API Communication**	• The system uses a REST API to handle communication between the frontend and backend components. • The API processes image analysis requests and delivers results back to the frontend in real-time. • This architecture ensures scalability, modularity, and efficient data exchange across different system components.
FR7	**Result Download**	• Users have the option to download the generated analysis report in PDF format. • This feature allows healthcare professionals to save, share, or print the report for record-keeping, consultation, or further study. • The download functionality is designed to ensure compatibility with standard devices and software.

Software Requirements Specifications

Non-Functional Requirements

No.	Attribute	Description
NFR1	**Reliability**	• The system must demonstrate high reliability, ensuring it can handle more than 95% of user requests without failure. • This includes maintaining uptime, error handling, and system recovery in case of unexpected issues. • The system should incorporate fault-tolerant mechanisms, such as redundant servers and error logging, to identify and address problems proactively, minimizing downtime.
NFR2	**Security**	To protect sensitive user and patient data, the system must implement robust security measures: • All data, including uploaded images and analysis results, must be encrypted during transmission and storage using secure encryption standards (e.g., AES, SSL/TLS). • User authentication is required for access to the system, with features such as encrypted passwords, session management, and multi-factor authentication (if applicable). • Access control mechanisms should ensure that only authorized users can access specific features or data.
NFR3	**Performance**	• The system must be optimized to process each image and deliver results within an average time of 10 seconds. • This requirement ensures responsiveness and usability, particularly for healthcare professionals needing quick analysis. • Performance optimization techniques, such as efficient algorithm implementation, caching, and load balancing, should be employed to meet this standard.
NFR4	**Portability**	• The system should be designed to operate seamlessly across major operating systems, including Windows, Linux, and MacOS.

		• This ensures accessibility for a wide range of users and devices. • Compatibility should be achieved through platform-independent development frameworks and thorough testing on each platform.
NFR5	**Scalability**	The system should support simultaneous access and processing requests from up to 50 users without performance degradation. Scalability considerations include: • Implementing load balancing to distribute requests evenly across servers. • Optimizing backend services to handle increased workloads. • Ensuring the architecture can be extended to support more users if needed in the future.

Software Requirements Specifications

Hardware Requirements

Sr#	Component	Specification	Description
1	**Processor (CPU)**	Intel Core i7 or AMD Ryzen 7 (or higher)	A high-performance processor is needed to handle complex computations and support the deep learning model efficiently.
2	**Graphics Card (GPU)**	NVIDIA RTX 3060 or higher with at least 8GB VRAM	A powerful GPU is essential for accelerating deep learning tasks, particularly for training and inference of the model.
3	**RAM**	Minimum 16 GB	Sufficient memory is required to process large medical images and run deep learning frameworks without performance lag.
4	**Storage**	Minimum 512 GB SSD	High-speed SSD storage is required for quick access to datasets, model weights, and temporary processing files.
5	**Operating System**	Windows 10/11, Linux (Ubuntu 20.04 or newer), or macOS	The system should be compatible with widely used operating systems for flexibility in development and deployment.
6	**Display**	Full HD (1920x1080) resolution	A high-resolution display is required for clear visualization of medical images and analysis results.
7	**Internet Connectivity**	Broadband connection with a minimum speed of 10 Mbps	Required for accessing cloud resources, APIs, or downloading datasets and dependencies for the project.
8	**Power Supply**	650W or higher (for desktops)	A reliable power supply unit is essential to support the GPU and other hardware components.
9	**Peripherals**	Standard keyboard and mouse	Basic peripherals for system interaction and testing.

Optional Hardware for Enhanced Performance

Sr#	Component	Specification	Description
1	**External Storage**	1 TB External SSD or HDD	Useful for storing large datasets or backup of medical images and analysis results.
2	**High-Performance GPU**	NVIDIA RTX 3090 or NVIDIA A100	For faster training and handling larger datasets or more complex models.

Software Requirements Specifications

Use Case Diagrams

A simple use case diagram represents the interaction between the end user and the system, illustrating the relationship between various system operations and their functionalities. The use case identifies interactions such as uploading images, preprocessing, tumor detection, report generation, API communication, and result downloading.

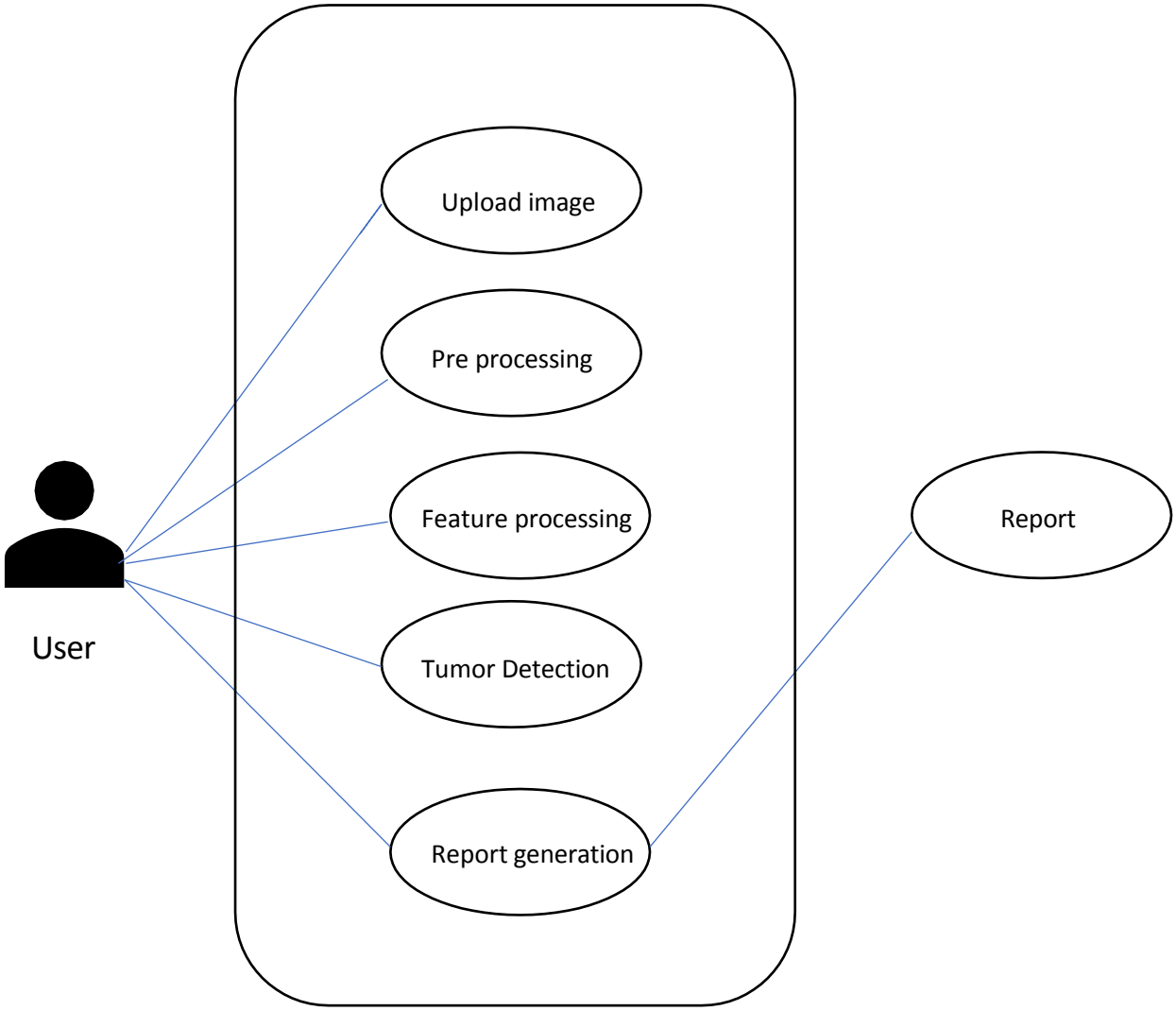

Figure 1.1: Use Case Diagram

Figure 1.1 illustrates the interactions between the **User** and the **Tumor Detection System**. It includes functionalities for uploading images, preprocessing, tumor detection, and generating and saving reports.

Table 1: Usage Scenario of Image Upload Use Case

This use case allows users to upload MRI or CT scan images to the system for analysis.

<table>
<tr><td>Use Case Name: Image Upload</td><td></td></tr>
<tr><td>Entry Condition: The user must select an image file to upload.</td><td>Participating Actors: End User</td></tr>
<tr><td>Flow of Events</td><td></td></tr>
<tr><td colspan="2">Basic Flow
The user clicks on the “Upload Image” button.
The user selects the image file from the local system.
The system validates the file type and size.
The system confirms successful upload.</td></tr>
<tr><td colspan="2">Alternative Flow
A1: Cancel, the system displays the message: “The upload process has been canceled by user demand.”</td></tr>
<tr><td colspan="2">Exceptional Flow
E1: Invalid file type or size, the system displays the message: “Please upload a valid MRI/CT image file.”</td></tr>
<tr><td>Exit Condition</td><td>The image is successfully uploaded and ready for preprocessing.</td></tr>
<tr><td>Constraint</td><td>The image file must be in a valid format (e.g., PNG, JPEG, DICOM) and within size limits.</td></tr>
</table>

Table. 1 showing the image is uploaded successfully if it meets the format and size requirements, ready for preprocessing. If canceled, a message notifies the user of the cancellation. Invalid file types or sizes trigger an error message, preventing further action until corrected.

Table 2: Usage Scenario of Image Preprocessing Use Case

This use case allows the system to preprocess uploaded images for tumor detection.

<table>
<tr><td colspan="2">Use Case Name: Image Preprocessing</td></tr>
<tr><td>Entry Condition: An image must be uploaded to the system.</td><td>Participating Actors: End User</td></tr>
<tr><td>Flow of Events</td><td></td></tr>
<tr><td colspan="2">Basic Flow
The system validates the uploaded image for clarity and compatibility.
The system normalizes the image for analysis (e.g., resizing, color correction).
The system confirms successful preprocessing.</td></tr>
<tr><td colspan="2">Alternative Flow
A1: Cancel, the system displays the message: “Preprocessing process canceled by user demand.”</td></tr>
<tr><td colspan="2">Exceptional Flow
E1: Error in image quality, the system displays the message: “Image preprocessing failed due to low image quality.”</td></tr>
<tr><td>Exit Condition</td><td>The image is preprocessed and ready for tumor detection.</td></tr>
</table>

Table. 2 showing the system preprocesses the uploaded image by validating its clarity and compatibility, followed by normalization for analysis. If the user cancels, a message is displayed indicating the cancellation. In case of low image quality, the system notifies the user that preprocessing failed, and the image is not ready for tumor detection.

Table 3: Usage Scenario of Tumor Detection Use Case

This use case processes the uploaded image to detect tumors using the deep learning model.

Use Case Name: Tumor Detection	
Entry Condition: A preprocessed image must be available.	**Participating Actors:** End User
Flow of Events	

• **Basic Flow**
  • The system sends the preprocessed image to the tumor detection model.
  • The system generates a confidence score and highlights tumor regions.
  • The model analyzes the image and identifies potential tumor regions.	

• **Alternative Flow** • A1: Cancel, the system displays the message: "Tumor detection process canceled by user demand."	
• **Exceptional Flow** • E1: Model error, the system displays the message: "Tumor detection failed. Please retry."	
Exit Condition	Tumor detection results are generated and ready for report creation.

Table. 3 showing the system processes the preprocessed image through the tumor detection model, generating a confidence score and highlighting potential tumor regions. If the user cancels, a message is displayed indicating the cancellation of the process. In case of a model error, the system notifies the user of the failure and prompts them to retry the detection. The tumor detection results are then ready for report generation.

Table 4: Usage Scenario of Report Generation Use Case

This use case generates a detailed analysis report based on tumor detection results.

Use Case Name: Report Generation	
Entry Condition: Tumor detection results must be available.	**Participating Actors:** End User
Flow of Events	

• **Basic Flow**
  • The system compiles tumor detection results, including confidence scores and tumor regions.
  • The system adds recommendations based on analysis.
  • The system generates a report in PDF format.	

• **Alternative Flow** • A1: Cancel, the system displays the message: "Report generation canceled by user demand."	
• **Exceptional Flow** • E1: Report generation error, the system displays the message: "Unable to generate the report. Please retry."	
Exit Condition	A PDF report is generated and ready for download.

The system compiles the tumor detection results, including confidence scores and identified tumor regions, and adds recommendations based on the analysis. If the user cancels the process, a message is displayed indicating the cancellation. In the case of a report generation error, the system alerts the user and prompts them to retry. Once successful, the system generates a PDF report, ready for download.

Class Diagram for Brain Tumor Detection System

Below is the class diagram for your system, representing key classes and their relationships:

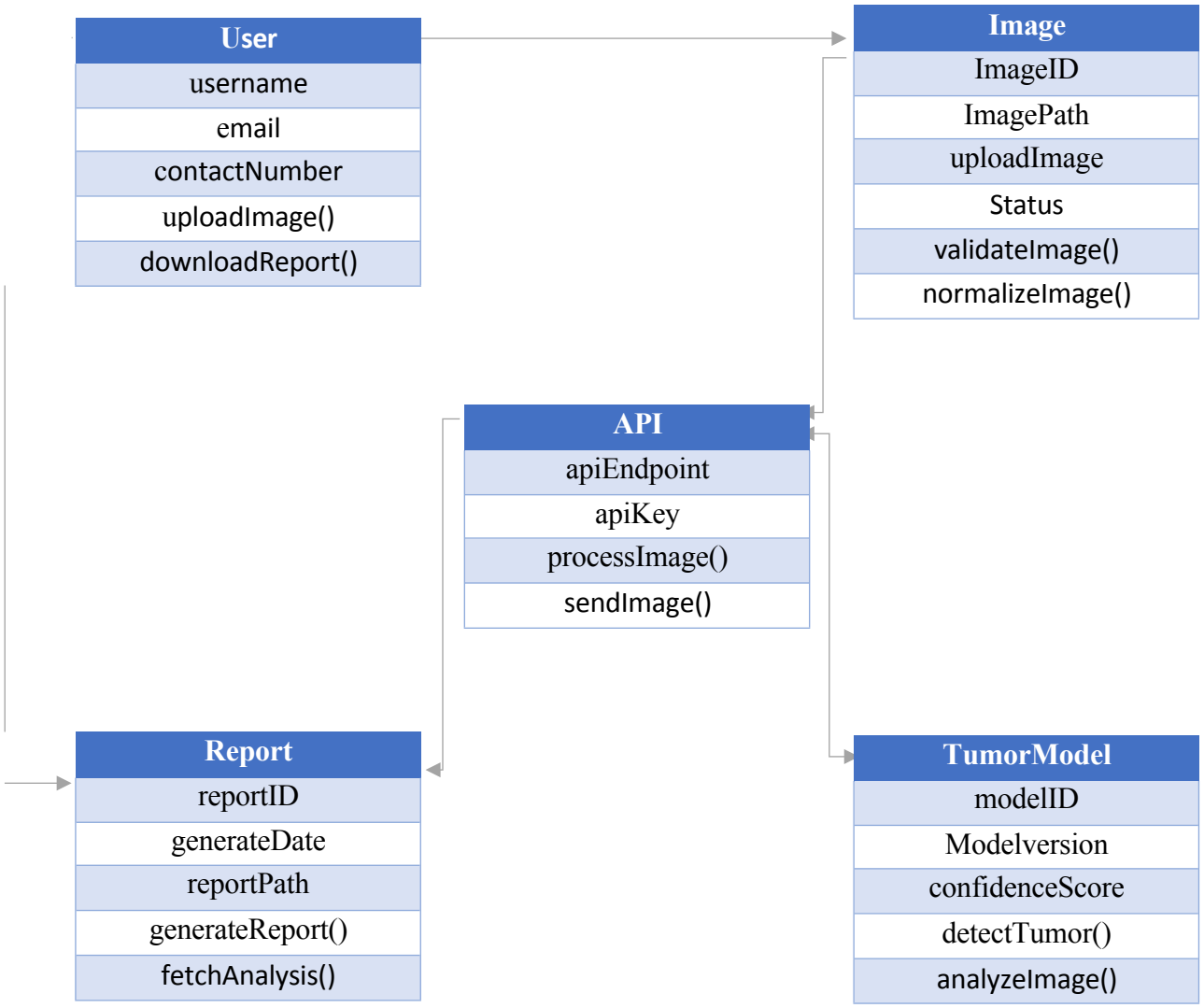

Figure 1.2: Class Diagram for Brain Tumor Detection System

Figure 1.2 shows the structure of the User, Image, API, and TumorModel classes. It outlines the attributes and methods involved in uploading images, processing them through an API, and detecting tumors using a tumor detection model.

Relationships

1. **User and Image**:
    - A **User** can upload **Image**.
2. **Image and TumorModel**:
    - An **Image** is processed by the **TumorModel**.
    - The **TumorModel** processes **Image**.
3. **TumorModel and Report**:
    - The **TumorModel** generates a **Report** for **Image**.
4. **User and Report**:
    - A **User** can download **Report**.
5. **API and TumorModel**:
    - The **API** is used to connect the **TumorModel** to the frontend.

Combination of Classes

Class Name	Associated Classes	Type of Relationship	Description
User	Image, Report	Association (1-to-many)	A user uploads images and downloads reports.
Image	User, TumorModel	Association (1-to-many, 1-to-1)	An image is uploaded by a user and processed by the tumor detection model.
TumorModel	Image, Report, API	Aggregation	The tumor model processes images and generates reports using API communication.
Report	User, TumorModel	Association (1-to-many, 1-to-1)	Reports are generated by the tumor model for specific images and downloaded by users.
API	TumorModel	Association (1-to-1)	The API handles communication between the frontend and the tumor model.

This table. 6 provides a comprehensive overview of class relationships and their combinations in your system.

Sequence Diagram for Brain Tumor Analysis System

The User interacts with the system through the Web Interface, providing input images for tumor detection. The System (Backend) processes the input using the Tumor Detection Model to analyze and detect tumors, then generates a detailed report via the Report Generator for the user's review.

Actors:

- User
- Web Interface
- System (Backend)
- Tumor Detection Model
- Report Generator
- API

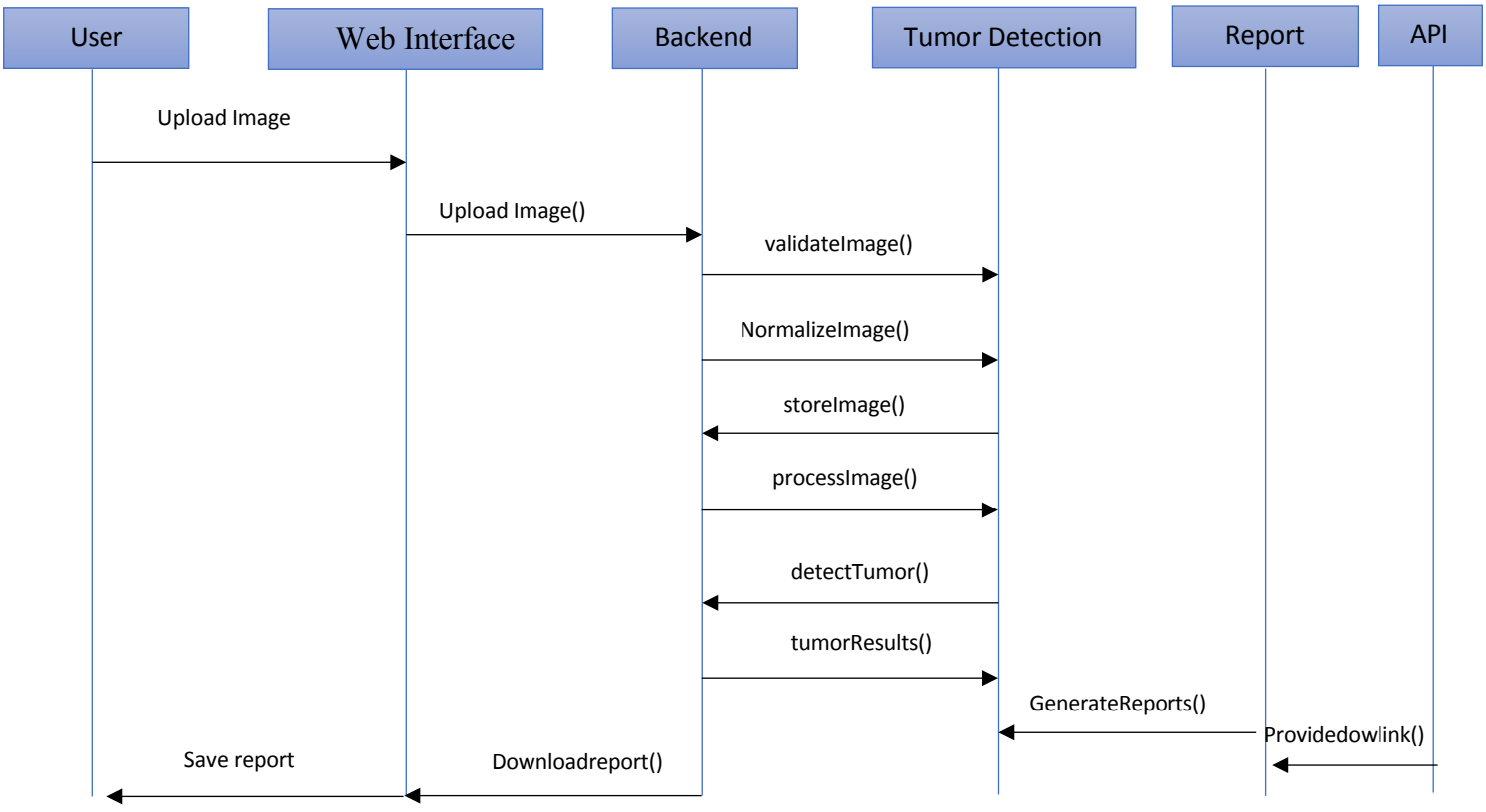

Figure 1.3: Sequence Diagram for Brain Tumor Detection

Steps in Sequence:

1. **User Uploads Image**:
    - The **User** interacts with the **Web Interface** to upload an image.
    - The **Web Interface** calls the uploadImage() method to validate and upload the image to the **System (Backend)**.
2. **System Preprocesses Image**:
    - The **System (Backend)** invokes the validateImage() and normalizeImage() methods on the uploaded image.
    - After validation, the image is processed and stored in the system.
3. **Tumor Detection**:
    - The **System (Backend)** sends the preprocessed image to the **Tumor Detection Model** via the processImage() method of the **API**.
    - The **Tumor Detection Model** detects tumors in the image and returns the detection results, including a confidence score and tumor regions.
4. **Report Generation**:
    - The **System (Backend)** sends the tumor detection results to the **Report Generator** via the generateReport() method.
    - The **Report Generator** creates a detailed analysis report, including tumor identification, confidence score, and recommendations.
5. **Download Report**:
    - The **System (Backend)** provides the **Web Interface** with the link to download the generated report.
    - The **User** then clicks on the "Download" button to download the report from the **Web Interface**.

Entity-Relationship (ER) Diagram for Brain Tumor Detection

Entities and Attributes

1. **User**
    - Attributes:
        - UserID (Primary Key)
        - Name
        - Email
        - ContactNumber
2. **Image**
    - Attributes:
        - ImageID (Primary Key)
        - UserID (Foreign Key)
        - FileName
        - UploadDate
        - Status (e.g., Pending, Processed)
3. **Tumor Detection Result**
    - Attributes:
        - ResultID (Primary Key)
        - ImageID (Foreign Key)
        - TumorDetected (Boolean)
        - ConfidenceScore
        - DetectionDate
4. **Report**
    - Attributes:
        - ReportID (Primary Key)
        - ResultID (Foreign Key)
        - FileName
        - GeneratedDate
5. **APIRequest**
    - Attributes:
        - RequestID (Primary Key)
        - ImageID (Foreign Key)
        - RequestTime

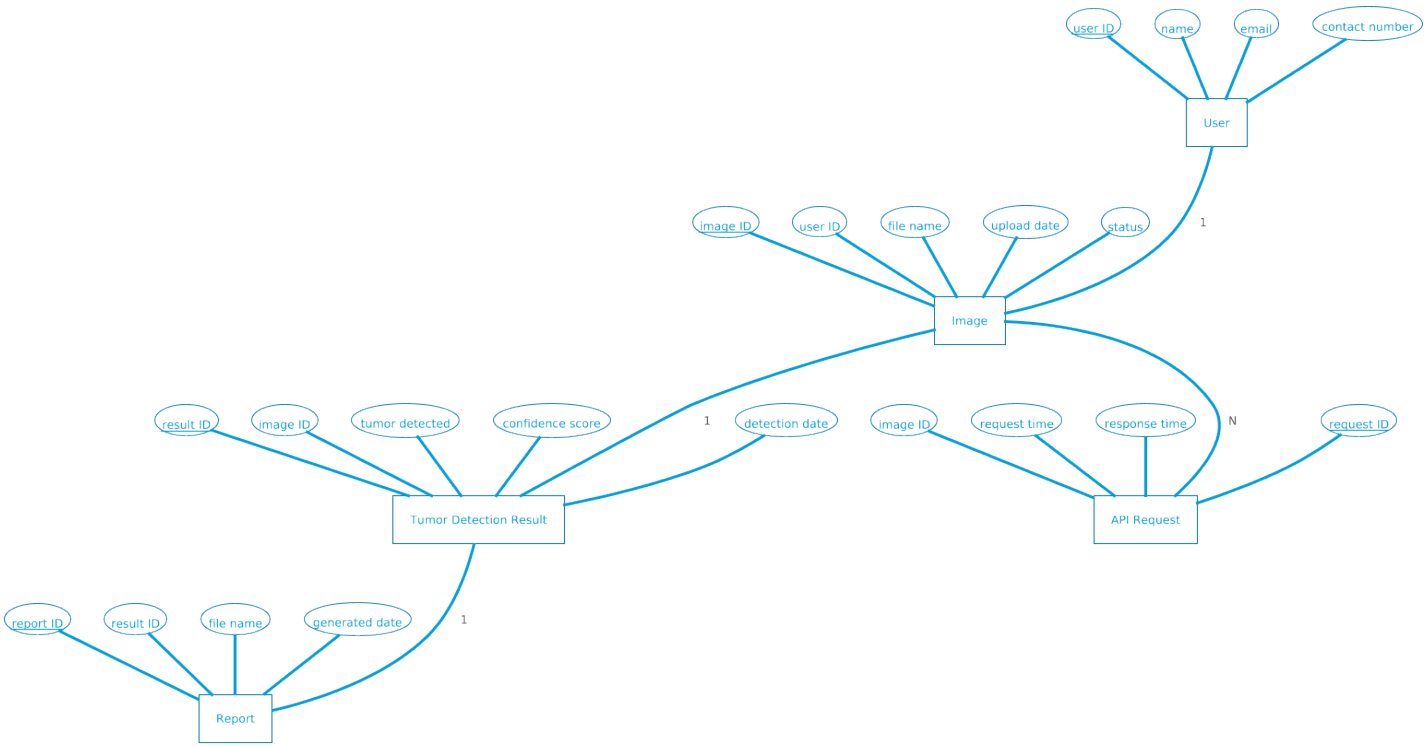

Figure 1.4: ER diagram for the Brain Tumor Detection

Figure 1.4 shows the relationships between **User**, **Image**, **Tumor Detection Results**, and **Report** entities in the brain tumor detection system. It illustrates how users upload images, which are processed to detect tumors, and generate corresponding reports.