### A PROJECT REPORT

on

### MEDVID:VISUALIZATION OF MRI REPORTS

Submitted in partial fulfillment of the requirements for the award of the degree of

### BACHELOR OF TECHNOLOGY

in

#### CSE(Artificial Intelligence & Machine Learning)

Submitted by

**B.Vennela (21UP1A6610)**

**K.Harika (21UP1A6623)**

**P.Deekshitha (21UP1A6639)**

**U.Bala Sri Bhavya Nandini (21UP1A6660)**

Under the Guidance of

#### Dr. D. Shanthi , Ph.D

Professor & Head of the Department, Department of CSE (AI&ML)



**DEPARTMENTOF CSE(ARTIFICIALINTELLIGENCE& MACHINELEARNING)**

### VIGNAN’S INSTITUTE OF MANAGEMENT AND TECHNOLOGY FOR WOMEN

#### (An Autonomous Institution) Approved by AICTE and Affiliated to JNTUH



### DEPARTMENT OF CSE(AI&ML) CERTIFICATE

This is to certify that project work entitled “**MEDVID:VISUALIZATION OF MRI REPORTS**” submitted by B. Vennela (21UP1A6610), K. Harika (21UP1A6623), P. Deekshitha (21UP1A6639) ,U. Bala Sri Bhavya Nandini (21UP1A6660) in the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in CSE(AI&ML) **VIGNAN’S INSTITUTE OF MANAGEMENT AND**

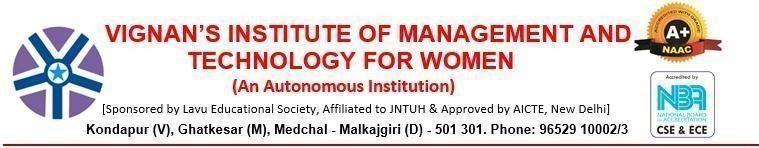
**TECHNOLOGY FOR WOMEN** is a record of Bonafide work carried by them under my guidance and supervision. The results embodied in this project report have not been submitted to any other University or institute for the award of any degree.

**Signature**   **Signature**

**Guide Project Cordinator**

**Signature Signature**

**External HOD(AI & ML)**



### DEPARTMENTOF CSE(AI&ML)

##### DECLARATION

We here by declare that the work reported in the present project entitled “**MEDVID:VISUALIZATION OF MRI REPORTS**” is a record of bonafied work duly completed by us in the Department of CSE (AI&ML) from Vignan’s Institute of Management and Technology for Women, affiliated to JNTU, Hyderabad. The reports are based on the project work done entirely by us and not copied from any other source. All such materials that have been obtained from other sources have been duly acknowledged.

The result embodied in this project report have not been submitted to any other University or Institute for the award of any degree to the best of our knowledge and belief.

| **SI No.** | **Roll Number** | **Student Name** | **Signature of the Student** |
| --- | --- | --- | --- |
| 1. | 21UP1A6610 | B.Vennela |  |
| 2. | 21UP1A6623 | K.Harika |  |
| 3. | 21UP1A6639 | P.Deekshitha |  |
| 4. | 21UP1A6660 | U.Nandini |  |



We would like to express our sincere gratitude and indebtedness to our project Guide **Dr.D. Shanthi**, Ph.D her valuable suggestions and interest throughout the course of this project.

We would like to extend our gratitude to **Mr. S. Sandeep Babu** (Project Coordinator) Department of CSE- Artificial Intelligence and Machine Learning from Vignan's Institute of Management and Technology For Women, for his valuable suggestions and timely help while the project.

We are also thankful to **Dr. D. Shanthi**, Head of the Department of CSE- Artificial Intelligence and Machine Learning from Vignan's Institute Of Management And Technology For Women, Hyderabad for providing excellent infrastructure and a nice atmosphere for completing this project successfully as a part of our B.Tech. Degree (CSM).

We convey our heartfelt thanks to the lab staff for allowing us to use the required equipment whenever needed.

Finally, we would like to take this opportunity to thank our family for their support through the work. We sincerely acknowledge and thank all those who gave directly or indirectly their support in the completion of this work.

#### B Vennela(21UP1A6610) K.Harika(21UP1A6623)

**P.Deekshitha (21UP1A6639)**

**U.Nandini (21UP1A6660)**



Medical reports, especially MRI reports, often contain complex medical terminology that can be difficult for patients to understand, creating a communication gap between healthcare providers and patients. This lack of clarity can lead to confusion, anxiety, and reduced adherence to treatment plans. **MedVid** addresses this challenge by using advanced natural language processing (NLP), machine learning, and video generation technologies to transform intricate MRI reports into clear, engaging, and easy-to-understand animated videos.MedVid begins by analyzing MRI reports using NLP models trained on clinical language to extract key information such as diagnoses, anatomical references, and relevant medical terms. This data is converted into a structured script written in simplified, patient-friendly language. The script can be translated into multiple languages, making the system accessible to a diverse patient population.The generated script is then paired with synchronized audio and visuals to create a personalized animated video. These animations visually depict the affected areas, explain the condition, and outline potential treatments, helping patients understand their health status in a more intuitive and reassuring way. By presenting information through a combination of narration, text, and visuals, MedVid enhances comprehension and engagement.

In addition to improving patient understanding, MedVid strengthens the doctor-patient relationship by promoting transparency and empowering patients to make informed decisions about their care. It is especially valuable in telemedicine, patient education, and multilingual healthcare environments.MedVid not only simplifies medical information but also promotes health literacy, improves patient satisfaction, and contributes to better healthcare outcomes. By bridging the gap between medical professionals and patients, it transforms how complex health data is communicated—making it more human, accessible, and ef

**CONTENTS PAGE NO**

##### **ABSTRACT**

##### INTRODUCTION

2.0 Problem Statement

* 1. Objective 
  2. Motivation 
  3. Existing System 
  4. Proposed System 
  5. Scope and Purpose

##### [LITERATURE REVIEW](#_heading=h.evubl1tt1cq8)

##### [SYSTEM REQUIREMENTS AND SPECIFICATIONS](#_heading=h.sir3n4clebwi)

##### Software Requirements

##### Hardware Requirements

##### SYSTEM DESIGN

* 1. System Architecture 
  2. UML Diagrams
     1. Use Case Diagram 
     2. Class Diagram 
     3. SequenceDiagram 
     4. Activity Diagram 
     5. Deployment Diagram 
     6. ER-Diagram
     7. Data Flow Diagram

**6.Module split-up**

**7.Methodologies**

**8.Testing 26**

8.1 Unit Testing 26

8.2 Black Box Testing 27

8.3 White Box Testing 28

8.4 Test Cases 29

**9.** **RESULT**

**10.** **CONCLUSION** 

* 1. 
  2. 

**8 REFERENCE 43**

**Table of Figures:**

| **S.No** | **Figure No** | **Figure Name** |
| --- | --- | --- |
| **1** | fig:4.1 | System Architecture |
| **2** | fig:4.2.1 | UseCase Diagram |
| **3** | fig:4.2.2 | Class Diagram |
| **4** | fig:4.2.3 | Sequence Diagram |
| **5** | fig:4.2.4 | Activity Diagram |
| **6** | fig:4.2.5 | Deployment Diagram |
| **7** | fig:4.2.2 | ER diagram |
| **8** | fig:4.2.2 | Data Flow Diagram |
| **9** | fig:8.1.1 | Test Case 1 |
| **10** | fig:8.1.2 | Test Case 2 |
| **11** | fig:9.1.1 | Result 1 |
| **12** | fig:9.1.2 | Result 2 |
| **13** | fig:9.1.3 | Result 3 |
| **14** | fig:9.1.4 | Result 4 |

**Tables**

| **S.No** | **Table Name** |
| --- | --- |
| **3.1** | Literature Survey Table |
| **6.1** | Technology Table |

**Introduction**

Medical imaging techniques like MRI (Magnetic Resonance Imaging) have revolutionized healthcare, enabling early detection and precise diagnosis of complex medical conditions. However, the reports generated from these scans often contain technical terminology that is challenging for patients to understand. According to a 2023 article in The New York Times, approximately 60% of patients find medical reports overly complex, leading to confusion and anxiety. Additionally, a study published in the Journal of Medical Internet Research highlighted that poor communication of medical information reduces patient engagement by 40%, often resulting in suboptimal treatment adherence.

The COVID-19 pandemic further emphasized the importance of accessible healthcare communication, as reported by The Lancet, where remote consultations and telemedicine became mainstream. Yet, a significant barrier persisted: patients struggled to interpret diagnostic reports without face-to-face explanations from healthcare providers. This gap underscores the need for patient-centered solutions that simplify medical communication.

MedVid was proposed to address this critical issue by transforming complex MRI reports into visually engaging and patient-friendly animated videos. Leveraging advancements in natural language processing (NLP) and AI- driven video generation, MedVid bridges the communication gap, enabling patients to better comprehend their medical conditions and fostering improved trust and engagement in healthcare.

The problem we are addressing here is the complexity in understanding MRI reports, which is a critical issue for patients. MRI reports often contain intricate medical terminology and detailed imaging descriptions that are difficult for non-medical individuals to comprehend. This lack of clarity creates a communication gap, leaving patients confused and anxious about their health conditions. Effective understanding of these reports is essential for making informed decisions and adhering to treatment plans. Studies show that patients who understand their medical information are more likely to follow prescribed treatments and feel empowered. Bridging this gap is crucial for improving healthcare communication and patient outcomes



* MedVid utilizes automated, scalable technologies like BERT for text understanding, WaveNet for audio synthesis, and Stable Video Diffusion for video generation, integrating multilingual support and advanced optimization techniques to create accessible, high-quality, and efficient patient- friendly videos.
* MedVid simplifies complex MRI reports into patient-friendly videos using NLP, video generation, and narration to make medical information engaging, understandable, and empowering for patients.
* MedVid addresses limitations of existing solutions by offering efficient, pathology-specific content with real-time and 3D imaging, guided by detailed system workflows and compliant with HIPAA to ensure data security.
* MedVid’s modular, user-centric design ensures adaptability to emerging technologies, prioritizing accessibility and engagement to enhance health literacy, trust in medical care, and health outcome.

# 

In the modern healthcare landscape, effective communication between medical **MedVid** leverages cutting-edge technologies—Natural Language Processing (NLP), AI-driven script generation, automated video creation, and multilingual voiceovers—to convert complex MRI reports into clear, visually engaging, and easily comprehensible animated videos. By doing so, MedVid not only demystifies medical terminology but also empowers patients by making them active participants in their healthcare journey.professionals and patients is crucial for delivering high-quality, patient-centered care. However, medical reports—particularly MRI reports—are often written in highly technical language that is difficult for the average patient to understand. This creates a significant barrier to comprehension, contributing to patient confusion, anxiety, and a lack of engagement in their own treatment. In many cases, patients leave medical consultations with only a vague understanding of their condition, which can negatively impact their ability to follow treatment plans or make informed decisions.

The motivation behind MedVid is rooted in the desire to reduce patient stress, foster greater trust in medical professionals, and support informed decision-making. By delivering information in a format that is accessible, personalized, and engaging, MedVid improves health literacy and patient satisfaction. This project aims to transform how medical data is communicated—shifting from static, technical reports to dynamic, patient-friendly narratives that truly make a difference.



Current systems designed to explain medical reports often fall short in delivering patient- specific, comprehensive, and engaging information. These systems primarily focus on general or broad medical content rather than tailoring explanations to specific reports like MRIs. As a result, they fail to address the unique complexities and detailed nuances of individual medical conditions, leaving patients with limited understanding.

Many of these systems rely heavily on manual intervention, where medical professionals are required to simplify the content or create educational materials. This manual process is time-consuming, resource-intensive, and not scalable for widespread implementation. Furthermore, the Natural Language Processing (NLP) tools employed in these systems are generally generic and not fine-tuned for medical terminology. This lack of specialization often results in explanations that are overly simplistic or fail to provide clarity on intricate medical details, which are critical for patient comprehension.

The content generated by these systems is typically presented in static formats, such as plain text or simple images, which lack the visual and interactive appeal necessary to engage patients effectively. Generic videos, if included, often fail to personalize the content based on the patient's specific medical report. The absence of dynamic media formats, such as interactive or animated videos, limits the system's ability to create an impactful and memorable learning experience for patients.

Security and compliance with standards such as HIPAA are often overlooked or inadequately addressed in these systems. This poses a significant risk to patient privacy and data protection, which are paramount in healthcare applications. Additionally, the scalability of these systems is hindered due to their dependence on manual processes and limited automation. This restricts their ability to serve a larger patient base efficiently and effectively.



1. Time-Consuming Process:

MRI scans, though powerful, often involve a lengthy workflow—from patient preparation to scanning, post-processing, and interpretation by radiologists. This can lead to delays in diagnosis and treatment, particularly in busy healthcare settings. Additionally, patients may need to wait days before they receive a readable report, which adds to anxiety and uncertainty.

1. Lack of Real-Time Imaging: Most MRI systems are not designed to deliver real-time imaging. This limitation makes it difficult to use MRI in scenarios that require immediate feedback, such as during surgical planning or interventional procedures. The inability to observe dynamic physiological changes limits its utility in real-time decision-making.
2. No 3D or Volumetric Views: Traditional MRI outputs are typically in the form of 2D slice images, which require radiologists to mentally reconstruct the 3D structure of organs or abnormalities. This not only increases the complexity of diagnosis but also makes it difficult for patients to visualize or understand their condition, as there is no holistic or intuitive representation of their internal anatomy.
3. Non-Specific to Pathology: MRI reports are generally technical and focused on raw findings, often lacking contextual explanation of how those findings relate to the patient's symptoms or diagnosis. This makes the reports less meaningful or understandable to patients, especially those without a medical background. Furthermore, a lack of customization in reporting makes it harder to tailor communication to individual patient needs.
4. High Noise and Artifacts: MRI images can suffer from various types of noise and artifacts due to patient movement, equipment limitations, or external interference. These distortions can obscure critical diagnostic details, leading to potential misinterpretations. Noise also lowers the clarity of visualization, especially for small or subtle abnormalities.



MedVid distinguishes itself from other projects in medical communication and patient education by offering a range of innovative features designed to enhance usability and effectiveness. Unlike generic systems, MedVid is specifically tailored for MRI reports, ensuring that the explanations are accurate, detailed, and relevant to the patient’s medical condition. This specialization allows it to address the unique complexities of MRI findings, making it more effective than broad, general-purpose solutions.

A key advantage of MedVid is its end-to-end automation, which eliminates the need for manual intervention. From input processing to video generation, the system automates every step, ensuring efficiency, scalability, and consistent quality. This automation is further complemented by the use of advanced Natural Language Processing (NLP) techniques that are fine-tuned for medical terminology. By leveraging state-of-the-art NLP models, MedVid can accurately interpret complex medical data and convert it into clear, understandable language that patients can easily grasp.

MedVid also stands out for its ability to generate multimedia videos. Unlike traditional systems that rely on static images or plain text, MedVid creates dynamic, visually engaging videos that include audio narration, animations, and synchronized visuals. This multimedia approach not only makes the information more accessible but also enhances patient engagement and retention of medical knowledge.



The scope of MedVid encompasses the transformation of complex medical information, specifically MRI reports, into accessible, patient-friendly formats. It aims to address the communication gap between healthcare providers and patients by delivering visually engaging, personalized videos that simplify intricate medical data. MedVid is designed to cater to a diverse audience, including patients with limited medical knowledge, those facing linguistic barriers, and individuals with varying levels of digital literacy. Its applications extend beyond patient education, as it can also be used by medical professionals to streamline communication, reduce the time spent on explaining reports, and enhance overall patient satisfaction. The system integrates advanced NLP, video generation, and multilingual support, making it scalable for healthcare organizations globally.

Additionally, MedVid prioritizes compliance with security and privacy standards, ensuring it can be deployed in sensitive medical environments.The primary purpose of MedVid is to empower patients by making complex MRI reports understandable, reducing their anxiety, and enabling them to make informed healthcare decisions. It addresses the challenges posed by technical medical jargon and the lack of personalized communication in existing systems. MedVid seeks to foster trust and engagement between patients and healthcare providers by delivering tailored, multimedia-rich content that is both informative and visually appealing. The project aims to improve health literacy, enhance patient outcomes, and set a benchmark for innovative healthcare communicationtools.



1. Visualizing Scanner Utilization From MRI Metadata and Clinical Data Authors: Pradeeban Kathiravelu , Nishchal Singi Published in: IEEE Computer, 2023. This study presents a visual approach to understanding MRI scanner usage within a healthcare network. By analyzing MRI metadata and clinical data, the research offers insights into scanner utilization patterns, which can inform resource allocation and operational efficiency.
2. Patient 3D Data Visualization with AR-based Interactive Technology for Brain MRI,Authors: Vishakha Pareek , Shreyansh Sharma Published in: IEEE Conference Publication, 2023 .This research explores the use of augmentedreality (AR) for interactive 3D visualization of brain MRI data. The AR- based system enhances surgical planning and in-situ guidance during brain tumor resection, demonstrating the potential of immersive technologies in medical visualization.
3. A Web-Based Medical Video Indexing Environment, Authors: Engin Mendi; Coskun Bayrak Published in: IEEE Conference Publication, 2010.The paper presents a web- based system for indexing medical videos, providing functionalities for efficient retrieval and management of video content. Such systems are crucial for organizing and accessing large repositories of medical videos, supporting both educational and clinical needs.
4. Medical Image Processing, Analysis and Visualization in Clinical Research, Authors

M.J. McAuliffe;F.M. Lalonde Published in: IEEE Conference Publication, 2001. This work introduces an extensible , platform-independent program designed for medical image processing and visualization. Tailored for clinical research, the program offers tools for analyzing and visualizing medical images, supporting various research and diagnostic applications.

1. Visualizing MRI & CT Scans Using Mixed Reality,Authors Naeema Ziyad; Salih Yoosaf Published in: IEEE Xplore, 2023. The article outlines methodologies for visualizing MRI and CT scans through mixed reality, enhancing the interpretability of complex medical images. This approach facilitates better understanding for both medical professionals and patients, aligning with MEdVid's objective of making medical information more accessible.
2. Vid2Pix-A Framework for Generating High-Quality Synthetic Videos Authors:Oda O.Nedrejord;Vajira Thambawita Publised in: IEEE International Symposium on Multimedia,2020.This paper presents a Generative Adverarial Network(GAN)-based framework for creating high-quality synthetic videos.The techniques discussed could be leveraged in MEdVid to generate realistic visual content from textual MRI reports.

| S.NO | Title | Author & Year | Metholodogies | Achievements | Limitations/Drawbacks |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Visualizing Scanner Utilizatio From MRI Metadata and Clinica Data | Pradeeban Kathiravelu,Nishchal Singi  [2023] | Analyzed MRI metadata an clinical data using statistica techniques and develope interactive visualizations with  tools like Matplotlib and Plotly | Identified scanner usag nefficiencies, enablin  optimized resource allocation and reduced patien  .wait times. | Lacked real-time data integration an focused on a single healthcare networ limiting generalizability. |  |
| 2 | Patient 3D Data Visualization with AR-based Interactive Technology for Brain MRI | VishakhaPareek Shreyansh Sharma  [2024] | Utilized augmented reality (AR) technology top create interactive 3  visualizations of brain MR i data, enabling surgeons to manipulate and analyze detailed brain structures in real time. | Enhanced surgical lanning and in-situ guidancea for brain tumor resections r mproving precision an reducing risks. | Limited to specific surgical scenario nd requires specialized hardware estricting accessibility and scalability. |  |
| 3 | A Web-Based Medical Video Indexing Environment | Engin Mendi;Cosku Bayrak  [2010] | Developed a web-based system using database managemen and indexg techniques t enable efficient retrieval ande organization of medical video content. | Facilitated streamlined acces to large medical vide o repositories,  nhancing usability fo educational and clinica applications. | Focused primarily on indexing and oretrieval, lacking advanced features lik content analysis or automatic tagging. |  |
| 4 | Medical Image Processing, Analysi  and Visualization in Clinica Research | M.J. McAuliffe;F.M  Lalonde  [2001] | Developed a platform  independent program fo medical image processing an visualization, utilizing modula tools for analysis and visua representation. | Supported clinical  research and diagnostic dapplications by providing flexible, extensible system fo processing and visualizin  medical  images | Limited by the computationa capabilities of the time, making it les asuitable for handling modern high resolution datasets and real-++++tim processing. |  |

,



## Software Requirements

## This section describes the software requirements essential for developing and running the project "MEDVID:Visualization of MRI reports". These requirements are classified into system software, libraries, APIs, and development tools.

## **3.1.1Operating system:**

## The system must run on a modern operating system that supports Python and its required libraries. The application is cross-platform and can be executed on any OS that supports Python 3 and the associated libraries.

* Windows 10 / 11 (64-bit): Windows is recommended for users who prefer a GUI-based interface and compatibility with common IDEs like Visual Studio Code or PyCharm. Most Python libraries used in the project are well-supported on Windows
* Linux (Ubuntu 20.04 or newer): Linux provides a powerful development environment and is ideal for server-based deployment. It offers high performance for model execution and supports terminal-based operations for package installations and script executions.
* macOS 10.15 (Catalina) or newer: macOS is suitable for developers using Apple hardware. It supports Python development natively and allows seamless integration with open-source tools and virtual environments.

**3.1.2 Programming Language:**

Python 3.10 is used for the entire development due to its simplicity, vast ecosystem, and strong support for natural language processing, web development, and multimedia processing.

Python's mature ecosystem offers modules for every stage of the project, from web app development (Flask) to summarization (Transformers) and video generation (MoviePy).

**3.1.3 Libraries and Frameworks:** This project leverages various open-source Python libraries for handling different tasks:

| Library/Framework | Purpose | Details |
| --- | --- | --- |
| Flask | Web framework | Used to build the front-end and handle routes, file uploads, and HTTP POST/GET requests. |
| PyMuPDF (fitz) | PDF text extraction | Extracts raw textual content from the uploaded MRI reports in PDF format. |
| Transformers | Summarization | Uses a pre-trained NLP model (facebook/bart-large-cnn) from HuggingFace to generate concise summaries of lengthy medical texts. |
| gTTS (Google Text-to-Speech) | Text-to-speech | Converts the generated summary into speech in English or Hindi (based on user preference). |
| deep\_translator | Translation | Supports language translation, e.g., translating English summaries into Hindi using Google Translate. |
| moviepy | Video processing | Merges images and audio to generate a complete video summarizing the report. |
| Pillow (PIL) | Image processing | Opens and resizes image files before compiling them into a video. |
| keybert | Keyword extraction | Extracts the most relevant medical terms from the summarized report using BERT embeddings. |
| torch | Model execution | Required backend for the HuggingFace transformers library to run models locally. |

**3.1.4 External APIs and Services**

This project utilizes several external APIs and online services to enhance functionality and streamline various tasks, particularly in areas like summarization, translation, image retrieval, and multimedia generation. The **Google Custom Search API** is integrated for fetching relevant images based on extracted medical keywords. It allows the system to dynamically retrieve illustrative images that visually represent the content being discussed. The **Google Text-to-Speech (gTTS)** service is employed to convert the summarized medical report into natural-sounding speech, supporting multiple languages, including English and Hindi. For translation purposes, the project uses the **GoogleTranslator** module from the deep\_translator library, which facilitates the conversion of English text into regional languages like Hindi to support multilingual accessibility. Additionally, the **Hugging Face Transformers pipeline** is used for local summarization using the pre-trained BART model (facebook/bart-large-cnn), offering high-quality text summarization capabilities. These APIs and services together ensure that the system is intelligent, multilingual, and media-rich, delivering a user-friendly and engaging experience from medical report input to audiovisual output.

**3.1.5 Development Tools and Environment :**The development of this project involved several tools and technologies that supported web application design, audio-visual processing, and natural language processing. The following development tools and environments were used:

* **Programming Language:** Python was chosen for its simplicity, extensive library support, and strong community. It is ideal for building AI/ML-backed applications and integrates well with APIs and multimedia processing tools.
* **Code Editor**: VS Code was used as the primary integrated development environment due to its lightweight design, extensions for Python development, debugging features, and Git integration.
* **Web Framework**: Flask Flask, a micro web framework for Python, was used to build the web interface of the application. It handles routing, form submission, file uploads, and backend logic seamlessly.
* **Virtual Environment**: Python A virtual environment was set up to manage dependencies, avoid conflicts, and ensure consistency across different development machines.
* **Package Manager**: pip All necessary Python packages were installed and managed using pip. This included libraries for summarization, translation, speech synthesis, and video generation.

##### Processor

Minimum Quad-core CPU (Intel i5 or AMD Ryzen 5 equivalent) for efficient processing of AI algorithms and real- time feedback.

##### RAM

Minimum 8 GB for smooth execution of backend services and frontend applications.

##### Storage

##### Minimum 256 GB SSD for faster data retrieval and storage of training content, user data, and logs.

##### Network

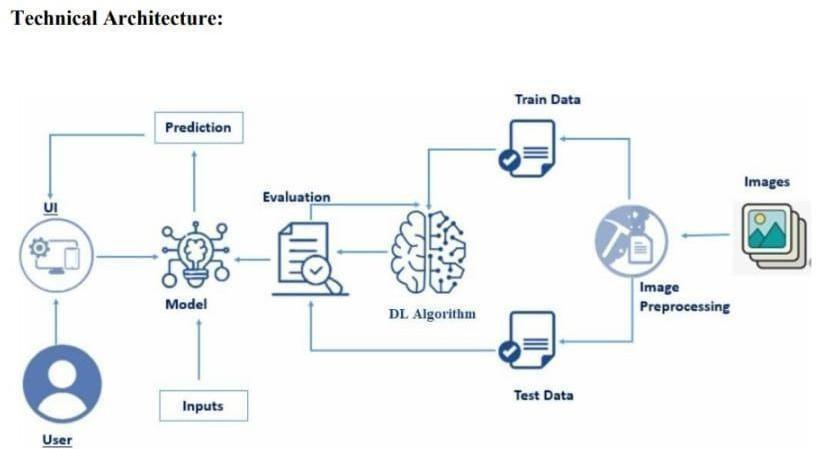
##### High-speed internet connection (minimum 10 Mbps) for real-time data synchronization and cloud- based operations.

##### GPU (Optional)

##### Dedicated GPU (NVIDIA GTX 1060 or higher) for accelerated training of machine learning models if required for advanced AI functionalities.

# 4 SYSTEM DESIGN

## 4.1 System Architecture or Block Diagram



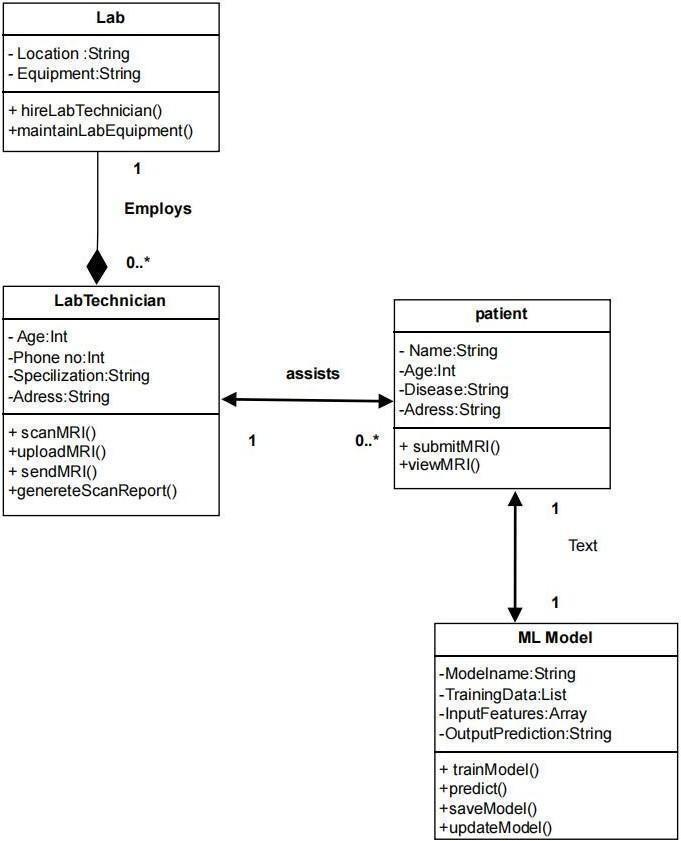
The Architecture depicted in the image represents a workflow integrating a deep learning (DL) model for image-based analysis. Users provide inputs via a user interface (UI), which are then processed by the DL model. Images undergo preprocessing, including transformation and cleaning, to create structured train and test datasets. The DL algorithm trains on this data, optimizing its parameters to predict outcomes effectively. Evaluations assess the model's accuracy and robustness. Once validated, the model can process new user inputs for predictions. The cyclical process ensures continuous refinement and adaptability, making it suitable for scalable, data-driven applications

## 4.2 UML Diagrams

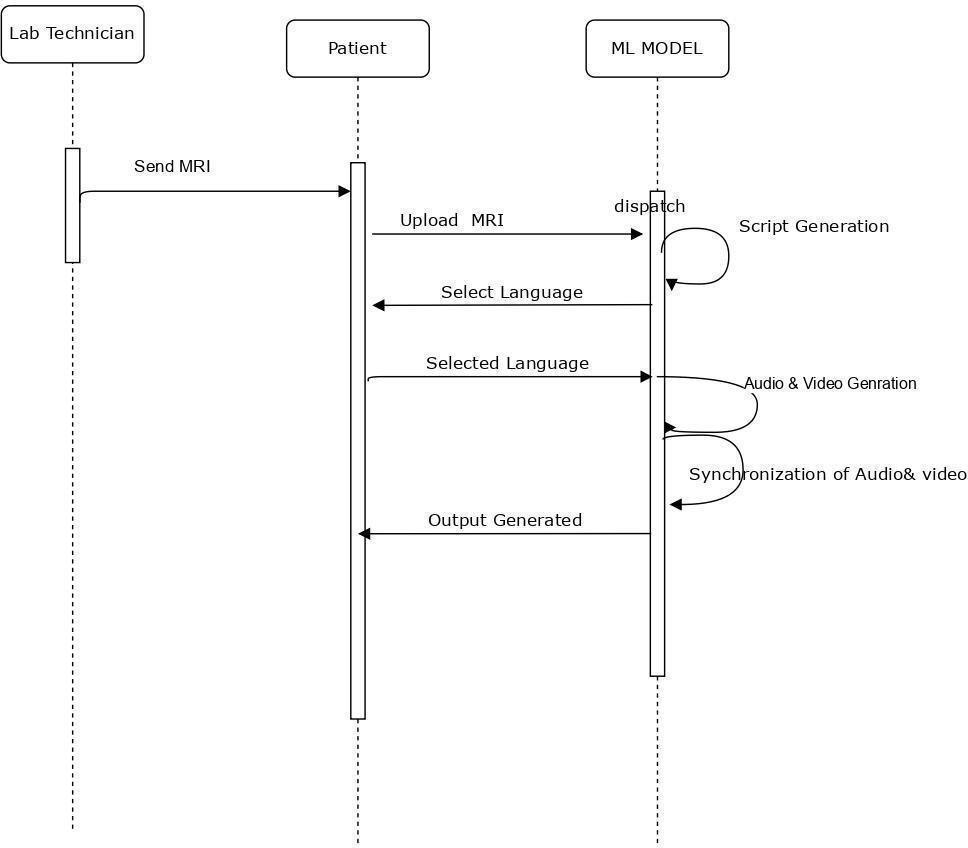
#### 4.2.1 Use Case Diagram



The Use case diagram depicts the process of visualizing medical reports using MRI data with three actors: Patient, Lab Technician, and ML Model. The patient undergoes an MRI scan assisted by the lab technician, who also uploads the MRI data to the system. The patient selects the preferred language for the report. The ML model processes the data to generate a script, create audio in the selected language, and produce a synchronized video visualizing the MRI findings. Finally, the patient receives a user- friendly, animated medical report. The system highlights seamless collaboration between the actors, leveraging advanced ML techniques to make complex medical data more accessible and comprehensible for patients.



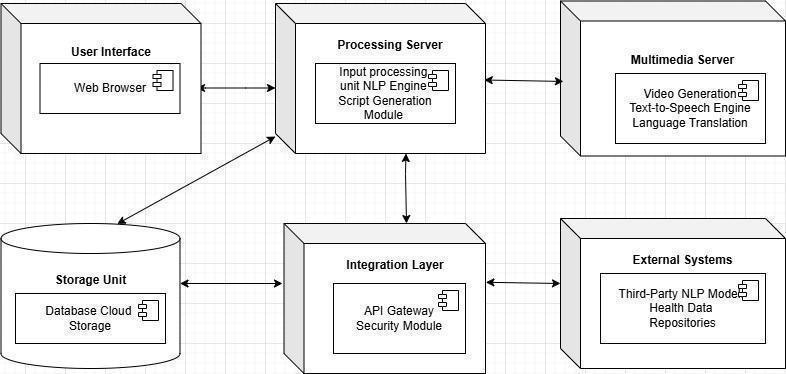
The provided class diagram represents the structure of an MRI Scan Management System integrated with a Machine Learning model, illustrating the key classes and their relationships. The system includes four main classes: **Lab**, **LabTechnician**, **Patient**, and **ML Model**. The **Lab** class manages lab equipment and employs multiple **LabTechnicians**, who are responsible for scanning, uploading, sending MRIs, and generating scan reports. Each **LabTechnician** assists multiple **Patients**, who can submit MRI requests and view their results. The **Patient** class stores personal and medical details. The **ML Model** class is linked to the **Patient** and processes MRI data to provide disease predictions using methods like training, prediction, model saving, and updating. This class diagram effectively captures the object-oriented design of the system, defining attributes, methods, and interactions necessary for MRI processing and intelligent medical analysis.



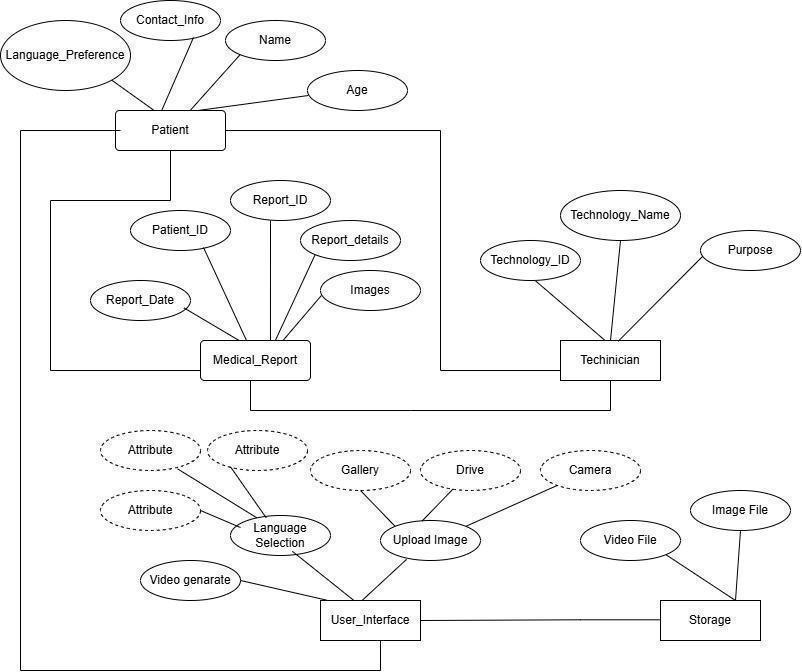
The Sequence diagram outlines the step-by-step interactions between the Lab Technician, Patient, and ML Model, ensuring clarity in the workflow. It maps key processes, including MRI submission, language selection, script generation, and synchronized audio- video creation, which are central to generating user-friendly, multilingual animated MRI reports. By outlining this structured workflow, the sequence diagram ensures that our system’s implementation aligns with the project’s objectives, ultimately improving usability and user experience while simplifying complex medical information.



This Activity diagram outlines the sequence of actions from scanning the patient to generating and viewing the video report, clarifying responsibilitiesBy highlighting decision points, like language selection and the steps for script, audio, and video generation, it ensures that the workflow is well- defined and easy to follow. This promotes better communication among stakeholders, identifies potential inefficiencies, and helps in debugging or refining the system design to improve performance and usability.

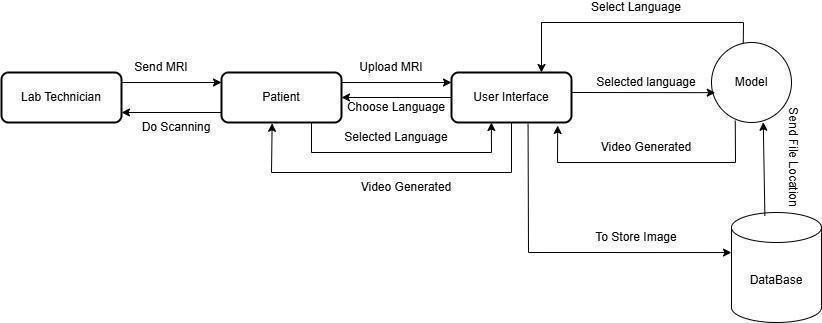


This Deployment diagram outlines the architecture of your project, showcasing how various components interact. Users access the system via a web browser to upload inputs like MRI scans and view outputs. The Processing Server handles input processing, script generation, and NLP tasks, while the Multimedia Server generates audio, video, and translations for multilingual support. Data is securely stored in the Storage Unit (cloud database). The Integration Layer manages communication through APIs and ensures security. External systems, such as third-party NLP models and health repositories, enhance functionality. This setup ensures efficient data processing, multimedia generation, and seamless, secure integration.



The Entity-Relationship (ER) diagram visually represents the relationships between entities in your project. It includes key entities like Patient, Technician, Medical\_Report, and User\_Interface, along with their attributes such as patient contact details, language preference, and medical report details (e.g., images, report date). The Technician entity connects to technology details and purpose, while the User\_Interface handles image and video uploads, gallery access, and video generation. Relationships link these entities to Storage, ensuring proper data flow and management. This ER diagram serves as a blueprint for database design, ensuring the data structure is well- organized and supports efficient system functionality.





The diagram illustrates the workflow of the MedVid system, which transforms MRI reports into easy- to-understand video explanations for patients. The process begins with a lab technician conducting the MRI scan and providing the results to the patient. The patient then interacts with the system through a user-friendly interface, where they upload their MRI scan and select their preferred language for the video explanation. The user interface serves as a central hub, facilitating communication between the patient and the backend components. The uploaded MRI file is stored in a database, while the selected language and file location are sent to a specialized model for processing. The model analyzes the MRI data and generates a personalized video explanation in the chosen language. This video is then sent back to the user interface, where the patient can access it easily. This end-to-end automated process ensures that complex medical data is converted into accessible, personalized, and engaging multimedia content, enhancing patient understanding and empowering informed healthcare decisions.

**5 IMPLEMENTATION:**

**5.1 Module Split-Up:**

1. **Input Processing:**

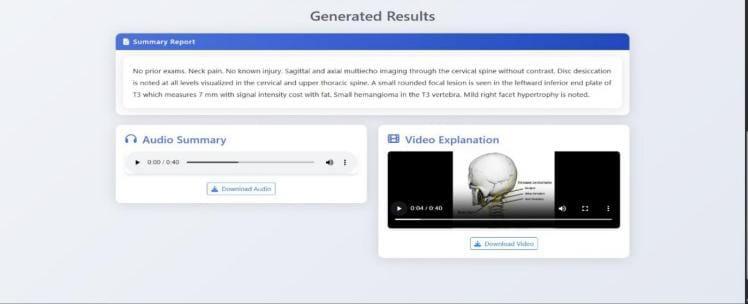
In this module, the uploaded MRI report in PDF format is processed to extract the raw text. Using PyMuPDF (fitz) library, the function extract\_text\_from\_pdf() reads and retrieves all textual content for further processing. Basic cleaning like removing unnecessary symbols or headers is also performed.

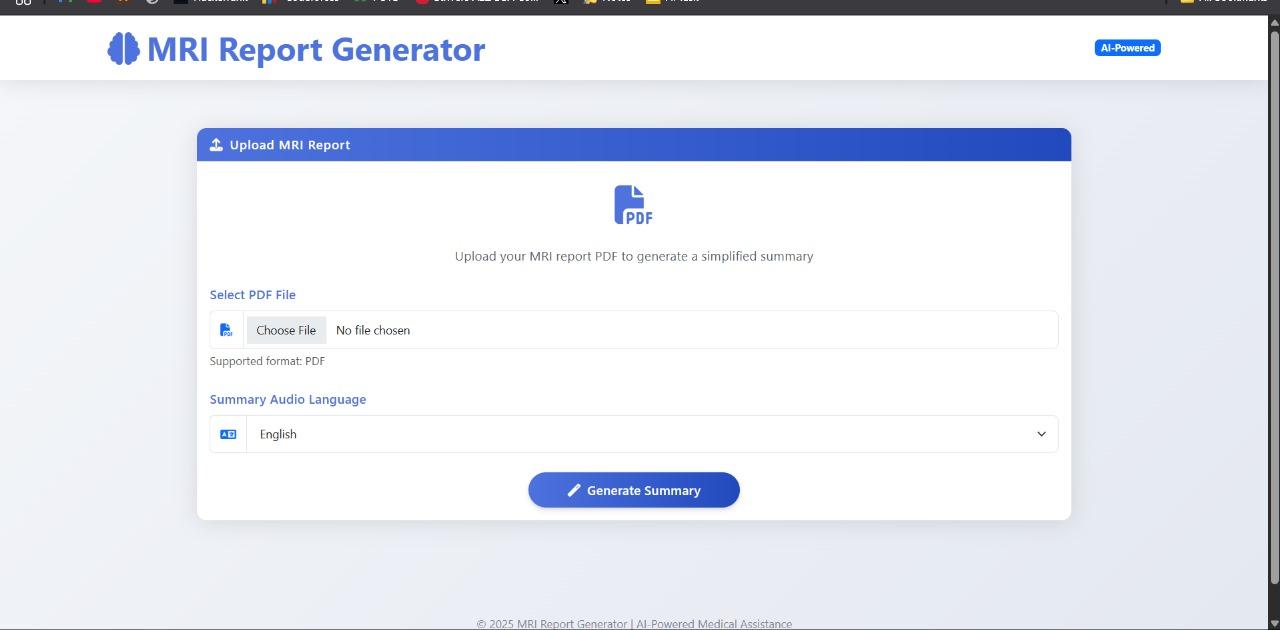
1. **Script Generation:** This module generates a concise and meaningful script from the extracted raw text. It involves two sub-steps: Text Cleaning:Removing unwanted signatures, page numbers, and other noise using Regular Expressions(re). Text Summarisation:Using the BART model (facebook/bart-large-cnn) to rewrite the content in a summarized, easy-to-understand format.
2. **Text-to-Speech:** The summarized text is converted into speech using the gTTS (Google Text-to-Speech) library. The text\_to\_speech() function generates a natural-sounding audio file narrating the MRI report's findings.
3. **Converting into Multiple Languages**: To make the system multilingual, the deep\_translator library is used to translate the English summary into other languages like Hindi, Tamil, etc., before text-to-speech conversion. This module increases accessibility for non-English speakers.
4. **Video Generations & Synchronization:** In the final module, relevant medical images are combined into a video using the moviepy library. The generated audio is synchronized with the image sequence to produce a final explanatory video that is both informative and visually appealing.

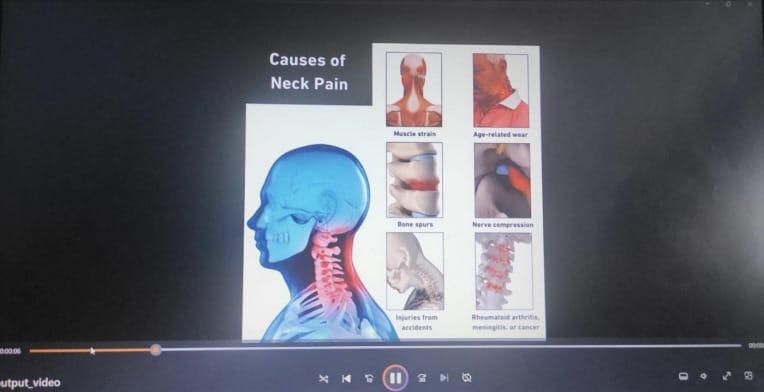
**5.2** **Methodologies:**

1. **Text extraction:**  Library: PyMuPDF (fitz) Process: The primary goal of this step is to extract raw textual content from MRI report PDFs that users upload. PyMuPDF (fitz) is chosen for its remarkable efficiency, lightweight nature, and its ability to handle complex and structured PDF layouts accurately. Function: extract\_text\_from\_pdf(). The extraction process involves loading each MRI report PDF file and parsing it page by page. PyMuPDF reads each page’s content while maintaining the logical and spatial layout of the document, ensuring that important structural elements like paragraph breaks, headings, and sections are not lost.The function extract\_text\_from\_pdf() systematically processes each page to collect the text sequentially and combines them to form a coherent body of text.Using PyMuPDF offers the advantage of dealing with diverse formatting, images embedded with text, and various font styles without losing the integrity of the information. This step is crucial because the raw extracted text forms the foundation for all subsequent text cleaning, processing, and summarization tasks.
2. **Text Cleaning:** Tool: Regular Expressions (re). process:After text extraction, the text often contains noisy elements like page numbers, doctor names, timestamps, report footers, hospital logos, and other non-essential metadata that could interfere with the analysis. Functions:Clean\_unwanted\_text(). After extraction, the text often contains unnecessary elements such as page numbers, doctor signatures, timestamps, and footers. These irrelevant details are removed using Regular Expressions (re). The clean\_unwanted\_text() function identifies and eliminates specific patterns, resulting in a cleaner, more meaningful text body. This step ensures that only medically relevant information is passed to the summarization module.
3. **Text Summarization:** Model:facebook/bart-large-cnn Library:transformers from Hugging Face. Function:summarize\_locally() Algorithm:Abstractive Summarization using Transformer-based BART architecture. This stage focuses on reducing the extracted and cleaned text into a compact, easily understandable form. The summarization uses the BART (Bidirectional and Auto-Regressive Transformer) model, specifically the facebook/bart-large-cnn variant, which is fine-tuned for summarization tasks.  
   The summarize\_locally() function operates using **abstractive summarization**, meaning it does not simply extract phrases from the input text but **rephrases and generates entirely new sentences** that capture the core ideas.The model reads the entire input MRI report, understands its meaning, and then re-articulates it in a concise and patient-friendly format. Important findings, diagnoses, and conclusions are highlighted in the summarized output, helping clinicians quickly grasp the report’s essence without wading through lengthy technical details.This step is critical for medical applications where brevity, precision, and clarity can enhance the efficiency of clinical decision-making.
4. **Keyword Extraction:** Tool:KeyBERT Uses BERT embeddings to e xtract top N keywords. Function: extract\_keywords(). After summarizing the report, the next step is to extract significant terms that represent the most critical aspects of the medical report. The tool KeyBERT is employed for this task.KeyBERT works by leveraging the power of BERT embeddings to understand the **semantic meaning** of the text and pinpoint the most meaningful phrases and keywords. The extract\_keywords() function identifies the top N keywords or key phrases that encapsulate the major topics and findings discussed in the MRI report.These extracted keywords are vital because,They represent medical concepts crucial to the patient's diagnosis, They serve as efficient "anchors" or "tags" for retrieving related content, They make it easier for automated systems to organize and index reports.
5. **Image Retrieval**: API:Google Custom Search API. For each keyword, the top relevant image is downloaded. Function: google\_search\_image() + download\_images(). For every extracted keyword, relevant medical images are fetched using the Google Custom Search API. The functions google\_search\_image() and download\_images() automate the process of searching and downloading the top matching images for each keyword. These images are crucial for building a synchronized, visually informative video that complements the summarized audio narration.
6. **Text-to-Speech(TTS):**  Library:gTTS(Google Text-to-Speech). Optional: Uses deep\_translator to convert summary to Hindi or any selected language before generating audio. Function:text\_to\_speech(). To make the summarized MRI report more accessible and engaging, the summarized text is converted into audio format using the gTTS (Google Text-to-Speech) library. The function text\_to\_speech() is responsible for this conversion. It takes the summarized textual content and generates natural-sounding speech, primarily in English. An optional enhancement involves using the deep\_translator library, which translates the summarized text into any regional language like Hindi before generating the audio output. This multilingual support ensures that the system can cater to a wider demographic, including patients who are more comfortable with their native languages. By providing audio output, the project significantly enhances accessibility, especially for visually impaired individuals or those who prefer auditory learning. Moreover, the ability to translate the summary into different languages before speech generation increases the flexibility and user-friendliness of the system
7. **Video Generation:** Library:moviepy. Merges images into a video and synchronizes it with the generated audio. Function: generate\_video(). After generating the audio narration, the next stage is to create an informative video that synchronizes the narration with relevant medical images. This is achieved using the **moviepy** library, which provides a range of tools for video editing, animation, and composition. The generate\_video() function is responsible formerging the sequence of images retrieved based on keywords with the generated audio. It aligns eachimage appropriately with the corresponding segments of the narration to maintain a logical flow and create an engaging experience. Additional features like smooth transitions, text overlays, and background music can be optionally added to enhance the video's visual appeal. This video acts as a comprehensive tool that not only explains the MRI findings audibly but also visually supports the explanation, thus offering a multi-sensory learning experience for patients and healthcare providers. The final output is a polished, easy-to-understand video that conveys complex MRI results in a simplified, patient-friendly format.

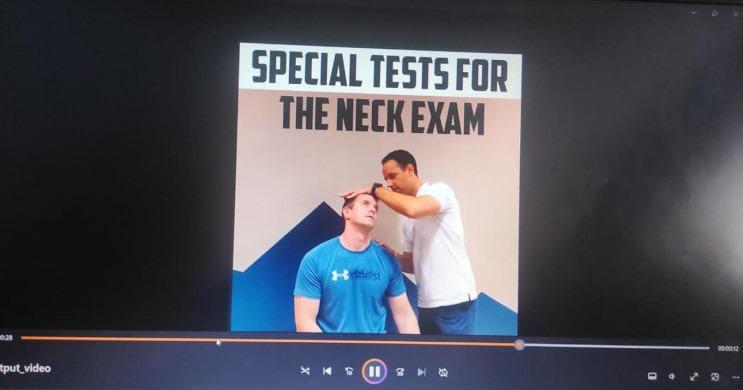
**9.Result**

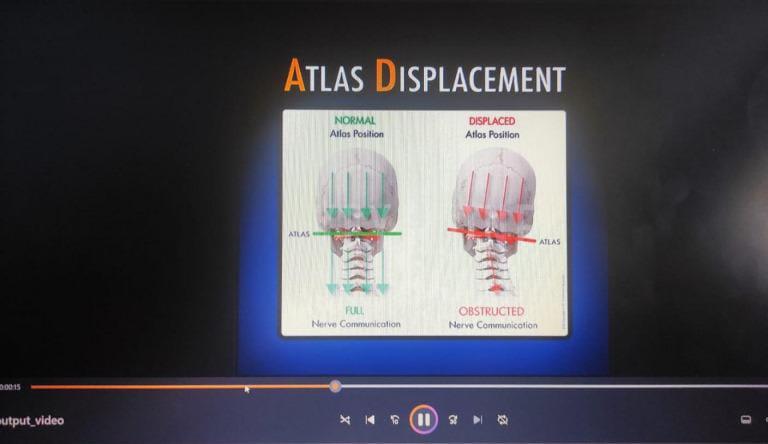












**7.Testing:**

Testing is process of evaluating and validating whether a software application or system meets the defined requirements and functions as expected. It involves executing the software with the intent of identifying bugs, errors, or inconsistencies and ensuring that all components work together seamlessly. Testing plays a vital role in the software development lifecycle by verifying both the functional and non-functional aspects of the system, such as performance, usability, security, and compatibility. There are various types of testing, including unit testing, integration testing, system testing, and user acceptance testing, each targeting different levels of the application. Conducting thorough testing helps developers identify and fix issues early, reducing development costs and avoiding costly failures after deployment. It also ensures that the application is reliable, stable, and secure, ultimately leading to higher user satisfaction and trust. By maintaining a robust testing strategy, teams can deliver high-quality software products that are efficient, maintainable, and ready for real-world use.To ensure the reliability and correctness of the system, two major types of testing were conducted:

**7.1 Black Box Testing:** In Black Box Testing, the system was tested from the user's point of view without accessing the internal source code. Different inputs were provided, and outputs were verified to match expected results.

* Uploaded various MRI reports (in different formats and sizes) to test the entire workflow.
* Verified if the text extraction, cleaning, summarization, keyword extraction, image retrieval, and video generation worked correctly.
* Tested invalid inputs such as non-PDF files and corrupted files to ensure proper error handling.
* Checked the audio generation and synchronization with video output.
* Evaluated multilingual translation functionality for accuracy.

**7.2** **White Box Testing:** In White Box Testing, the internal code structure, logic, and workflow of the system were analyzed and tested thoroughly. Individually tested all major functions like:

* extract\_text\_from\_pdf()
* clean\_unwanted\_text()
* summarize\_locally()
* extract\_keywords()
* google\_search\_image()
* text\_to\_speech()
* generate\_video()
* Verified function outputs for different types of inputs, including edge cases like empty text or extremely large PDFs.
* Checked for logical errors, code optimization, and exception handling.
* Tested data flow between different modules to ensure smooth integration.
* Performed performance testing during video generation to avoid memory leaks and time delays.

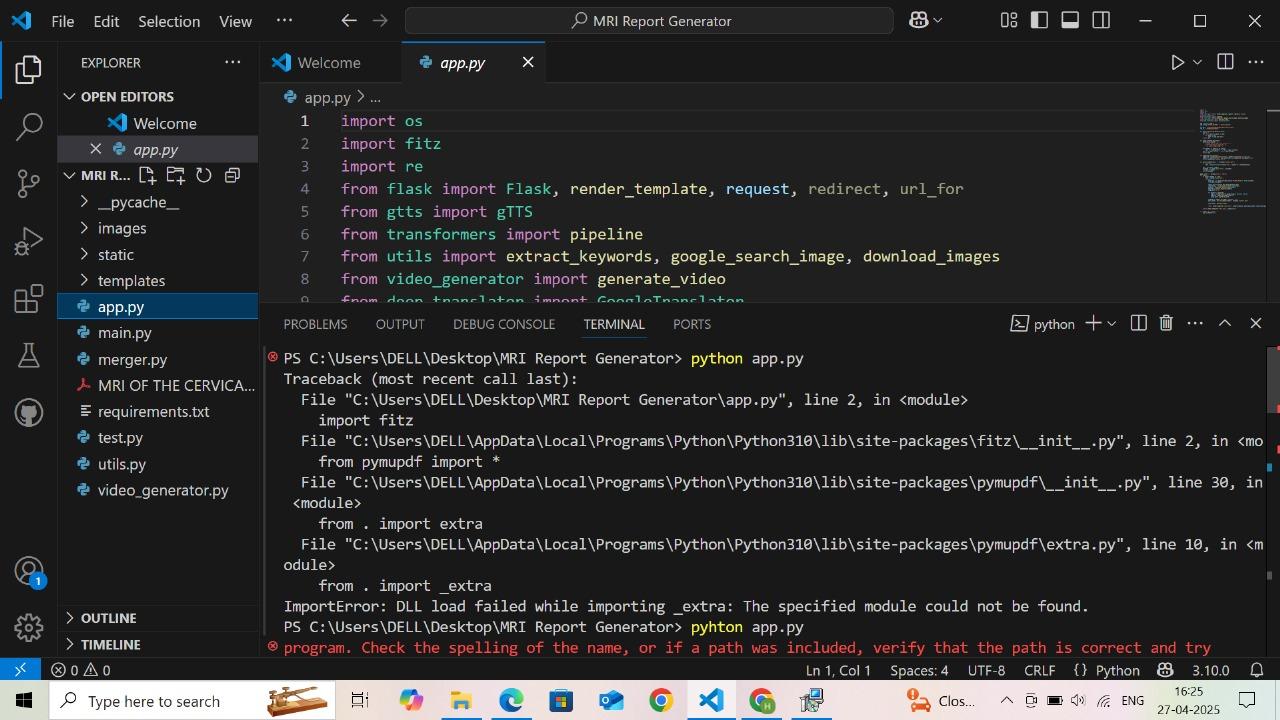
7**.3** **Unit testing:**

In Unit Testing, each individual component and function of the system was tested independently to validate its correctness and robustness. This testing helped to quickly identify and fix defects at an early stage of development, ensuring that each module performed reliably before integration

The following activities were performed during Unit Testing:

* Designed and implemented unit test cases using the unittest framework in Python to validate functionality at the function level
* Tested all critical functions individually, including:
* extract\_text\_from\_pdf()-Verified accurate extraction of text from various PDF structures, including scanned images and multi-column layouts.
* clean\_unwanted\_text()-Ensured effective removal of noise elements such as headers, footers, and unwanted symbols from the extracted text.
* summarize\_locally()-Evaluated the quality, coherence, and relevance of locally generated text summaries.
* extract\_keywords()-Tested the precision of keyword extraction for different types of input documents.google\_search\_image()-Checked that relevant images were consistently fetched based on extracted keywords.
* text\_to\_speech()-Verified the conversion of extracted summaries into natural-sounding audio files.
* generate\_video()-Confirmed the correct compilation of images and audio into synchronized video output.

**7.4 TEST CASE:**

****

While running the project, an issue was encountered related to the importing of the fitz module, which is used for PDF text extraction. Although the code specified import fitz, an ImportError occurred due to missing DLL files. Upon investigation, it was found that fitz is not a standalone package but a module provided within the pymupdf library. To resolve the issue, the pymupdf package was installed using the command pip install pymupdf. After correctly installing the package, the fitz module could be imported without errors, and PDF text extraction functionalities were verified successfully. This testing step ensured that the system could reliably extract text from uploaded PDF reports and reinforced the importance of proper dependency management during system setup.

In conclusion, MedVid addresses a critical issue in healthcare communication by transforming complex MRI reports into clear, personalized, and easily understandable videos. This innovative solution leverages advanced Natural Language Processing (NLP) and video generation technologies to bridge the communication gap between healthcare providers and patients. By providing multilingual, engaging, and visually rich content, MedVid reduces patient anxiety, promotes better understanding of medical conditions, and empowers patients to make informed decisions about their health. Through its structured approach, which integrates various custom algorithms and technologies, MedVid enhances health literacy and improves patient satisfaction. Ultimately, this project represents a significant step towards making healthcare information more accessible, fostering better health outcomes, and contributing to a more patient-centric healthcare system.

#### 10 Future Enhancement:

Future enhancements for the MedVid project aim to expand functionality and impact. Key advancements include personalized video explanations tailored to patient age, education, culture, and medical history, and real-time video generation upon report upload. Improved video quality with advanced algorithms like Stable Video Diffusion 2.0, interactive 3D visualizations, and multilingual support using tools like Multilingual BERT will enhance accessibility and clarity. Mobile apps for Android and iOS will provide on-the-go access, while integration with healthcare systems will ensure secure, compliant data retrieval. Feedback mechanisms, AR features, support for CT scans and X-rays, and cloud computing for faster processing and storage will position MedVid as a scalable, patient- centric healthcare communication tool.

# 11 REFERENCES

1. **Pradeeban Kathiravelu, Nishchal Singi**. "Visualizing Scanner Utilization From MRI Metadata and Clinical Data." *IEEE Computer*, 2023.
2. **Vishakha Pareek, Shreyansh Sharma**. "Patient 3D Data Visualization with AR- based Interactive Technology for Brain MRI." *IEEE Conference Publication*, 2023.
3. **Engin Mendi, Coskun Bayrak**. "A Web-Based Medical Video Indexing Environment." *IEEE Conference Publication*, 2010.
4. **M.J. McAuliffe, F.M. Lalonde**. "Medical Image Processing, Analysis and Visualization in Clinical Research." *IEEE Conference Publication*, 2001.
5. **Naeema Ziyad, Salih Yoosaf**. "Visualizing MRI & CT Scans Using Mixed Reality." *IEEE Xplore*, 2023.
6. **Oda O. Nedrejord, Vajira Thambawita**. "Vid2Pix - A Framework for Generating High-Quality Synthetic Videos." *IEEE International Symposium on Multimedia*, 2020.
7. **MedVidCL Dataset**: This dataset comprises 6,617 videos categorized into 'medical instructional', 'medical non-instructional', and 'non-medical' classes, designed to aid in medical video classification tasks.
8. **MedVidQA Dataset**: Containing 3,010 health-related questions with corresponding video timestamps, this dataset supports research in medical visual answer localization
9. **HealthVidQA Datasets**: These datasets, HealthVidQA-CRF and HealthVidQA-Prompt, are designed for answering health-related questions using medical videos, facilitating research in medical visual answer localization.
10. **Deepak Gupta, Kush Attal, Dina Demner-Fushman**. "A Dataset for Medical Instructional Video Classification and Question Answering." arXiv preprint arXiv:2201.12888, 2022
11. **Syed Abdul Mateen, Niharika Malvia, Syed Abdul Khader, Danny Wang, Deepti Srinivasan, Chi-Fu Jeffrey Yang, Lana Schumacher, Sandeep Manjanna**. "Thoracic Surgery Video Analysis for Surgical Phase Recognition." arXiv preprint arXiv:2406.09185, 2024.
12. **Deepak Gupta, Kush Attal, Dina Demner-Fushman.** "Towards Answering Health-related Questions from Medical Videos: Datasets and Approaches." arXiv preprint arXiv:2309.12224, 2023.
13. **David C. Wong, Stefan Williams.** "Artificial Intelligence Analysis of Videos to Augment Clinical Assessment: An Overview." *Journal of Medical Artificial Intelligence*, vol. 1, no. 1, pp. 1-10, 2024
14. "**A Review of Medical Diagnostic Video Analysis Using Deep Learning Techniques.**" *Applied Sciences*, vol. 13, no. 11, article 6582, 2023.
15. **Medical Imaging Datasets**: A curated list of medical imaging datasets available on GitHub, useful for research and development in medical image analysis.

**SAMPLE CODE**

**App.py:**

import os

import fitz

import re

from flask import Flask, render\_template, request, redirect, url\_for

from gtts import gTTS

from transformers import pipeline

from utils import extract\_keywords, google\_search\_image, download\_images

from video\_generator import generate\_video

from deep\_translator import GoogleTranslator

app = Flask(\_\_name\_\_)

app.config['UPLOAD\_FOLDER'] = 'static/uploads'

API\_KEY = 'AIzaSyCmVoVbMIEUaARPCyKK5t\_6DXLsCljKKYI'

CSE\_ID = '26565ead217784ed2'

def extract\_text\_from\_pdf(file\_path):

    text = ""

    with fitz.open(file\_path) as pdf:

        for page in pdf:

            text += page.get\_text()

    return text

def clean\_unwanted\_text(text):

    patterns\_to\_remove = [

        r"Thank you for your kind referral.\*",

        r"-Electronically Signed by:.\*",

        r"On \d{2}/\d{2}/\d{4}.\*",

    ]

    for pattern in patterns\_to\_remove:

        text = re.sub(pattern, '', text, flags=re.DOTALL)

    text = re.sub(r'\n\s\*\n', '\n\n', text).strip()

    return text

def summarize\_locally(text):

    summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

    summary = summarizer(text, max\_length=250, min\_length=100, do\_sample=False)

    return summary[0]['summary\_text']

def text\_to\_speech(text, l, filename="summary.mp3"):

    if l == "hi":

        text = GoogleTranslator(source='auto', target='hi').translate(text)

    tts = gTTS(text, lang=l)

    filepath = os.path.join('static', filename)

    tts.save(filepath)

    return filepath

@app.route('/', methods=['GET', 'POST'])

def index():

    if request.method == 'POST':

        file = request.files['pdf']

        if file:

            file\_path = os.path.join(app.config['UPLOAD\_FOLDER'], file.filename)

            file.save(file\_path)

            report\_text = extract\_text\_from\_pdf(file\_path)

            cleaned\_text = clean\_unwanted\_text(report\_text)

            summary = summarize\_locally(cleaned\_text)

            keywords = extract\_keywords(summary)

            image\_paths = []

            for keyword in keywords:

                img\_url = google\_search\_image(keyword, API\_KEY, CSE\_ID)

                img\_path = download\_images(img\_url)

                image\_paths.append(img\_path)

**MAIN.PY:**

import fitz

import re

from gtts import gTTS

from transformers import pipeline

from utils import extract\_keywords,google\_search\_image,download\_images

from video\_generator import generate\_video

API\_KEY = 'AIzaSyCmVoVbMIEUaARPCyKK5t\_6DXLsCljKKYI'

CSE\_ID = '26565ead217784ed2'

def extract\_text\_from\_pdf(file\_path):

    text = ""

    with fitz.open(file\_path) as pdf:

        for page in pdf:

            text += page.get\_text()

    return text

def clean\_unwanted\_text(text):

    patterns\_to\_remove = [

        r"Thank you for your kind referral.\*",

        r"-Electronically Signed by:.\*",

        r"On \d{2}/\d{2}/\d{4}.\*",

    ]

    for pattern in patterns\_to\_remove:

        text = re.sub(pattern, '', text, flags=re.DOTALL)

    text = re.sub(r'\n\s\*\n', '\n\n', text).strip()

    return text

def summarize\_locally(text):

    summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

    summary = summarizer(text, max\_length=250, min\_length=100, do\_sample=False)

    return summary[0]['summary\_text']

def text\_to\_speech(text, filename="summary.mp3"):

    tts = gTTS(text)

    tts.save(filename)

    print(f"Audio summary saved as {filename}")

# === EXECUTION ===

file\_path = 'MRI OF THE CERVICAL SPINE WO CONTRAST.pdf'

report\_text = extract\_text\_from\_pdf(file\_path)

cleaned\_text = clean\_unwanted\_text(report\_text)

# Summarize the report

summary = summarize\_locally(cleaned\_text)

print("\nSUMMARY:\n", summary)

keywords = extract\_keywords(summary)

for i in keywords:

    image = google\_search\_image(i, API\_KEY, CSE\_ID)

    download\_images(image)

# Generate the audio file

text\_to\_speech(summary, "mri\_summary.mp3")

generate\_video()

**MERGER.PY:**

from moviepy.editor import VideoFileClip, AudioFileClip

video = VideoFileClip("output\_video.mp4")

audio = AudioFileClip("mri\_summary.mp3").subclip(0, video.duration)

final = video.set\_audio(audio)

final.write\_videofile("merged\_output.mp4", codec="libx264", audio\_codec="aac", fps=24)

**TEST.PY:** from deep\_translator import GoogleTranslator

from gtts import gTTS

import os

def text\_to\_speech(text, l, filename="summary.mp3"):

    if l == "hi":

        text = GoogleTranslator(source='auto', target='hi').translate(text)

    tts = gTTS(text, lang=l)

    filepath = os.path.join('static', filename)

    tts.save(filepath)

    return filepath

print(text\_to\_speech("Hello, how are you?", "hi", "summary.mp3"))

**UTILS.PY:**

import requests

import wikipedia

from keybert import KeyBERT

import os

def google\_text\_search(search\_term, api\_key, cse\_id, num=1):

    search\_url = "https://www.googleapis.com/customsearch/v1"

    params = {

        "q": search\_term,

        "cx": cse\_id,

        "key": api\_key,

        "num": num

    }

    response = requests.get(search\_url, params=params)

    return response.json()

def get\_wikipedia\_summary(keyword, sentences=20):

    try:

        # Get a detailed summary with the specified number of sentences

        summary = wikipedia.summary(keyword, sentences=sentences)

        return summary

    except wikipedia.DisambiguationError as e:

        return f"Disambiguation error: {e.options}"

    except wikipedia.PageError:

        return f"No page found for {keyword}"

    except Exception as e:

        return f"An error occurred: {str(e)}"

def google\_search\_image(search\_term, api\_key, cse\_id, num=3):

    search\_url = "https://www.googleapis.com/customsearch/v1"

    params = {

        "q": search\_term,

        "cx": cse\_id,

        "key": api\_key,

        "searchType": "image",

        "num": num

    }

    response = requests.get(search\_url, params=params)

    return response.json()

def extract\_keywords(text, num\_keywords=10):

    kw\_model = KeyBERT()

    keywords = kw\_model.extract\_keywords(text, keyphrase\_ngram\_range=(1, 2), stop\_words='english', top\_n=num\_keywords)

    return [kw[0] for kw in keywords]

import uuid

def download\_images(response\_json, save\_folder='images'):

    if not os.path.exists(save\_folder):

        os.makedirs(save\_folder)

    items = response\_json.get('items')

    if not items:

        print("No images found.")

        return

    for idx, item in enumerate(items):

        image\_url = item.get('link')

        if image\_url:

            try:

                img\_data = requests.get(image\_url).content

                file\_ext = image\_url.split('.')[-1].split('?')[0]

                if len(file\_ext) > 5 or '/' in file\_ext:

                    file\_ext = 'jpg'

                # Generate a unique filename using uuid

                unique\_id = uuid.uuid4().hex[:8]  # short unique ID

                file\_name = f'image\_{unique\_id}.{file\_ext}'

                file\_path = os.path.join(save\_folder, file\_name)

                with open(file\_path, 'wb') as handler:

                    handler.write(img\_data)

                print(f"Downloaded: {file\_path}")

            except Exception as e:

                print(f"Failed to download {image\_url}. Reason: {e}")

**VIDEO\_GENERATOR:**

from PIL import Image, UnidentifiedImageError

from moviepy.editor import ImageClip, concatenate\_videoclips, AudioFileClip

import numpy as np

import os

def generate\_video():

    images\_folder = 'images'

    audio\_file = 'static/summary.mp3'

    image\_files = sorted([

        os.path.join(images\_folder, img)

        for img in os.listdir(images\_folder)

        if img.lower().endswith(('.png', '.jpg', '.jpeg', '.webp'))

    ])

    audio = AudioFileClip(audio\_file)

    audio\_duration = audio.duration

    valid\_clips = []

    num\_images = len(image\_files)

    if num\_images == 0:

        raise Exception("No images found in the folder.")

    image\_duration = audio\_duration / num\_images

    for img\_path in image\_files:

        try:

            img\_pil = Image.open(img\_path).convert('RGB')

            img\_np = np.array(img\_pil)

            clip = ImageClip(img\_np).set\_duration(image\_duration).resize(height=720)

            valid\_clips.append(clip)

            print(f"Added image: {img\_path}")

        except UnidentifiedImageError:

            print(f"Skipped invalid image: {img\_path}")

        except Exception as e:

            print(f"Error processing {img\_path}: {e}")

    if not valid\_clips:

        raise Exception("No valid images found to create a video.")

    video = concatenate\_videoclips(valid\_clips, method="compose")

    # Attach audio and match duration explicitly

    video = video.set\_audio(audio)

    video = video.set\_duration(audio.duration)

    video.write\_videofile(

        "static/output\_video.mp4",

        fps=24,

        codec='libx264',

        audio\_codec='aac'

    )

    folder = 'images'

    if os.path.exists(folder):

        for f in os.listdir(folder):

            file\_path = os.path.join(folder, f)

            if os.path.isfile(file\_path):

                os.remove(file\_path)