DESIGN AND DEVELOPMENT OF LUMBAR SPINE DEGENERATIVE CLASSIFICATION

**A**

**MAJOR PROJECT-I REPORT**

Submitted in partial fulfillment of the requirements

for the degree of

**BACHELOR OF TECHNOLOGY**

in

**CSE-ARTIFICIAL INTELLIGENCE & DATA SCIENCE**

By

**GROUP NO. 10**

|  |  |
| --- | --- |
| Sparsh Sahu | 0187AD211039 |
| Mahak Mirza | 0187AD211020 |
| Eashan Tiwari | 0187AD211015 |

Under the guidance of

**Prof. Ruchi Jain**

(ASSISTANT PROFESSOR)

****

**Department of CSE-Artificial Intelligence & Data Science**

**Sagar Institute of Science & Technology (SISTec), Bhopal (M.P.)**

**Approved by AICTE, New Delhi & Govt. of M.P.**

**Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.**

**DECEMBER-2024**

***Sagar Institute of Science & Technology (SISTec), Bhopal***

*Department of* *CSE*-*Artificial Intelligence & Data Science*

******

***CERTIFICATE***

I hereby certify that the work which is being presented in the B.Tech. Major Project-I Report entitled **DESIGN & DEVELOPMENT OF LUMBAR SPINE DEGENERATIVE CLASSIFICATION,** in partial fulfillment of the requirements for the award of the degree of ***Bachelor of Technology***, submitted to the Department of *CSE* with *Artificial Intelligence & Data Science*, *Sagar Institute of Science & Technology (SISTec)****,*** Bhopal (M.P.) is an authentic record of my own work carried out during the period from July-2024 to Dec-2024 under the supervision of **Prof. Ruchi Jain (Assistant Professor).**

The content presented in this project has not been submitted by me for the award of any other degree elsewhere.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Sparsh Sahu** | **Mahak Mirza** | **Eashan Tiwari** |
| **0187AD211039** | **0187AD211020** | **0187AD211015** |

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

***Date:***

**Dr. D.K. Rajoriya**

**Principal, SISTec**

**Dr. Vasima Khan**

**HOD, CSE-AI&DS**

**Prof. Ruchi Jain**

**Project Guide**

**ACKNOWLEGMENT**

We would like to express our sincere thanks to **Dr. D.K. Rajoriya, Principal, SISTec** and **Dr. Swati Saxena, Vice Principal, SISTec** Gandhi Nagar, Bhopal for giving us an opportunity to undertake this project.

We also take this opportunity to express a deep sense of gratitude to **Dr. Vasima Khan, HOD**, **Department of CSE-Artificial Intelligence & Data Science** for her kindhearted support.

We extend our sincere and heartfelt thanks to our Guide, **Prof. Ruchi Jain** for providing us with the right guidance and advice at the crucial junctures and for showing us the right way.

I am thankful to the Project Coordinator**, Prof. Ruchi Jain** who devoted her precious time in giving us the information about the various aspect and gave support and guidance at every point of time. I am thankful to their kind and supportive nature. His inspiring nature has always made my work easy.

I would like to thank all those people who helped me directly or indirectly to complete my project whenever I found myself in any issue.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **TITLE** | | **PAGE NO.** |
| Abstract | | i |
| List of Abbreviation | | ii |
| List of Figures | | iii |
| Chapter 1 | Introduction | 1 |
|  | 1.1 About Project | 2 |
|  | 1.2 Purpose | 2 |
|  | 1.3 Project Objective | 2 |
|  | 1.4 Significance of Project | 3 |
| Chapter 2 | Software & Hardware Requirements | 4 |
|  | 2.1 Hardware requirements | 5 |
|  | 2.2 Software requirements | 5 |
|  | 2.3 Client Side requirements | 5 |
| Chapter 3 | Problem Description | 6 |
|  | 3.1 Current Scenario and Challanges | 7 |
|  | 3.2 Importance of Automation | 7 |
|  | 3.3 Objective of the Project | 8 |
|  | 3.4 Scope of the Project | 8 |
|  | 3.5 Summary | 9 |
| Chapter 4 | Literature Survey | 10 |
|  | 4.1 Automated detection, labelling and radiological grading of clinical spinal MRIs | 11 |
|  | 4.2 Landmarking and feature localization in spine x-rays | 12 |
|  | 4.3 Automatic Detection and Measurement of Spinous Process Curve on Clinical Ultrasound Spine Images | 12 |
|  | 4.4 AI in Spinal Imaging: Current Status and Future Directions | 13 |
| Chapter 5 | Software Requirements Specification | 14 |
|  | 5.1 Functional Requirements | 15 |
|  |  |  |
|  |  |  |
|  | 5.2 Non-Functional Requirements | 16 |
| Chapter 6 | Software Design | 17 |
|  | 6.1 Overview | 18 |
|  | 6.2 Use Case Diagram | 18 |
|  | 6.3 Project Flow Diagram | 19 |
| Chapter 7 | Machine Learning Module | 20 |
|  | 7.1 Dataset Description | 21 |
|  | 7.2 Pre-processing Steps | 22 |
|  | 7.3 ML Model Analysis | 23 |
|  | 7.4 Result Analysis | 24 |
| Chapter 8 | Front End Connectivity | 26 |
|  | 8.1 HTML Code | 27 |
|  | 8.2 Connectivity | 28 |
| Chapter 9 | Coding | 32 |
|  | 9.1 Image Preprocessing | 33 |
|  | 9.2 Model Training | 34 |
|  | 9.3 Prediction and Evaluation | 35 |
| Chapter 10 | Output Screens | 38 |
| Appendix-1: Glossary of Terms | | 40 |
| References | | 43 |
| Project Summary | | 44 |

**ABSTRACT**

This project focuses on the development of a deep learning-based system for the classification of lumbar spine degenerative conditions using MRI images. The system utilizes advanced image processing techniques to extract relevant features from MRI scans, enabling the identification of degenerative conditions in the lumbar spine. The project employs a Convolutional Neural Network (CNN) to detect key coordinates within the MRI images, highlighting regions of interest associated with spinal degeneration.

Preprocessing steps, including image normalization and slicing, are applied to ensure consistent input data for the model. The model is trained on a large dataset of annotated MRI images, leveraging transfer learning with a pre-trained ResNet-18 architecture for enhanced accuracy and generalization.

The system's current functionality includes displaying the coordinates of spinal regions detected in the MRI images, with potential future improvements aimed at incorporating disease classification based on these coordinates. This project demonstrates the application of deep learning for medical image analysis, offering a valuable tool for automatic diagnosis and aiding in the detection of lumbar spine degeneration. The system can be integrated into clinical workflows to assist healthcare professionals in diagnosing spinal disorders more efficiently.

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **ACRONYM** | **FULL FORM** |
| MRI | Magnetic Resonance Imaging |
| ROI | Region of Interest |
| DICOM | Digital Imaging and Communications in Medicine |
| PT | PyTorch Model File Format |
| FLASK | Fast Lightweight Application Stack Kit |
| ML | Machine Learning |
| CNN | Convolutional Neural Network |
| NIFTI | Neuroimaging Informatics Technology Initiative |
| HTML | Hyper Text Markup Language |
| CSS | Cascading Style Sheets |
| IDE | Integrated Development Environment |
| GPU | Graphics Processing Unit |
| CSV | Comma Separated Values |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIG. NO.** | **TITLE** | **PAGE NO.** |
| 6.1 | Use Case Diagram | 18 |
| 6.2 | Project Flow Diagram | 19 |
| 7.1 | Software Life Cycle | 25 |
| 10.1 | Home Page | 39 |
| 10.2 | Prediction Page | 39 |

**CHAPTER – 1**

**INTRODUCTION**

**CHAPTER 1**

**INTRODUCTION**

* 1. **ABOUT PROJECT**

This project focuses on the development of a deep learning-based system designed to identify and classify degenerative conditions in the lumbar spine from MRI images. Spinal degenerative diseases, such as disc herniation and spondylosis, can significantly impact a patient’s mobility and quality of life. Early detection and accurate diagnosis are essential for effective treatment. The project leverages Convolutional Neural Networks (CNNs) to process and analyze MRI scans, aiming to automate the detection of key spinal regions associated with degeneration. By extracting key coordinates from MRI images, the system identifies and highlights areas of concern. The system currently focuses on detecting spinal regions but aims to expand to disease classification in future iterations, offering a comprehensive tool for the diagnosis of lumbar spine degenerative conditions.

* 1. **PURPOSE**

The purpose of this project is to develop a robust, automated system for identifying degenerative conditions in the lumbar spine using MRI images. By applying advanced image processing techniques and deep learning models, the project seeks to streamline the process of diagnosing spinal disorders, thus improving diagnostic accuracy and efficiency. With the ability to automatically analyze MRI scans, the system can assist healthcare professionals in identifying potential areas of degeneration, enabling earlier intervention and more personalized treatment. Ultimately, the project aims to contribute to the advancement of medical imaging technologies, supporting clinical practices and helping reduce the diagnostic burden on healthcare systems.

* 1. **PROJECT OBJECTIVE**

The primary objective of this project is to create an automated system that can accurately detect degenerative changes in the lumbar spine from MRI images using deep learning techniques. The system will focus on identifying key regions of the spine that show signs of degeneration, such as disc bulging, narrowing, and osteophyte formation. By implementing a Convolutional Neural Network (CNN), the project aims to automate the detection and classification process, allowing for faster and more consistent analysis of MRI scans. Additionally, the project will explore future expansions that could incorporate the classification of different spinal diseases based on these detected regions, enhancing the system's diagnostic capabilities.

* 1. **SIGNIFICANCE OF PROJECT**

The significance of this project lies in its potential to revolutionize the diagnosis and management of lumbar spine degenerative conditions. With the increasing reliance on medical imaging in diagnostic practices, accurate and efficient analysis of MRI images is critical. This system helps reduce the time and effort required for manual examination, providing quick and reliable results that can assist doctors in making timely decisions. The tool has the potential to support healthcare professionals in diagnosing spinal degenerative diseases with greater precision, improving patient outcomes. Furthermore, by automating the analysis of MRI images, this project contributes to the advancement of AI-based medical technologies, making them more accessible to healthcare settings, particularly in regions with limited access to expert radiologists.

**CHAPTER – 2**

**SOFTWARE & HARDWARE REQUIREMENTS**

**CHAPTER 2**

**SOFTWARE & HARDWARE REQUIREMENTS**

**2.1 HARDWARE REQUIREMENTS**

* **Processor:** High-performance CPU (i5 12th gen or higher) for training large models efficiently.
* **Memory:** The recommended RAM is 8GB or 16GB, to handle the data processing tasks smoothly. For larger datasets, more RAM may be beneficial.
* **Storage:** Generally, at least 10GB of free space is recommended for data storage and model saving.
* **GPU:** If using deep learning models, a dedicated GPU (such as NVIDIA’s GPUs) can significantly speed up model training.
* **Internet Connectivity:** Internet connectivity is required for data collection and for deploying the web application online
* **Display:** Full HD or higher resolution display (1920 x 1080 pixels or better) for clear visual inspection of MRI images and model outputs.

**2.2 SOFTWARE REQUIREMENTS**

* **IDE:** Visual Studio Code, a lightweight, cross-platform source-code editor developed by Microsoft.
* **Python:** The main programming language for implementing the language identification models, leveraging libraries like OpenCV, PyTorch NumPy, Pandas, Matplotlib and Scikit-learn.
* **Flask-Cors**: A Flask extension used to handle Cross-Origin Resource Sharing (CORS), enabling the frontend and backend to communicate seamlessly.

**2.3 CLIENT-SIDE REQUIREMENTS**

* **Browser:** A compatible browser to run the web application.

**CHAPTER – 3**

**PROBLEM DESCRIPTION**

**CHAPTER 3**

**PROBLEM DESCRIPTION**

In this chapter, we explore the core aspects of the "Lumbar Spine Degenerative Classification" project. We outline the problem domain, the need for an automated solution, and the objectives we aim to achieve using deep learning techniques. Additionally, we describe the significance of this project in the medical field and how the proposed system could improve the accuracy of lumbar spine degenerative condition detection using MRI images.

**3.1 CURRENT SCENARIO AND CHALLENGES**

In the current medical field, diagnosing lumbar spine degenerative conditions typically involves manual interpretation of MRI images by radiologists. This process is time-consuming and highly dependent on the expertise of the individual interpreting the images. Moreover, human error, especially in complex cases with unclear or borderline conditions, can lead to misdiagnosis. Additionally, manual evaluation is not always feasible in regions with a shortage of qualified healthcare professionals.

Challenges such as inconsistencies in MRI image quality, variation in imaging techniques, and the presence of subtle degenerative signs make it difficult for traditional methods to achieve consistent and accurate results. These limitations pose significant challenges in diagnosing conditions such as disc herniation, spondylosis, and spinal stenosis, affecting the overall efficiency of diagnosis and treatment planning.

**3.2 IMPORTANCE OF AUTOMATION**

Automating the process of diagnosing lumbar spine degenerative conditions using deep learning provides a significant opportunity to improve the accuracy and efficiency of the healthcare industry. The ability to automatically identify degenerative conditions in MRI scans can help overcome the limitations of manual interpretation, especially in resource-limited settings.

Automated systems offer several advantages:

* **Speed**: Automated diagnosis can be much faster than manual interpretation, enabling quicker decision-making.
* **Consistency**: Machine learning models can provide consistent results, minimizing human errors in diagnosis.
* **Scalability**: With an automated system, large volumes of MRI images can be processed efficiently, allowing for faster clinical assessments and supporting healthcare professionals in their decision-making process.

The integration of an automated diagnosis system into clinical workflows has the potential to revolutionize the speed, accuracy, and accessibility of lumbar spine condition diagnostics, ultimately improving patient outcomes.

**3.3 OBJECTIVES OF THE PROJECT**

The primary objectives of this project are as follows:

* The system will be designed to automatically identify key coordinates and features from MRI scans of the lumbar spine, highlighting potential degenerative areas.
* CNNs are known for their effectiveness in image classification tasks. By using pre-trained models and fine-tuning them for this specific task, the project aims to achieve high accuracy in detecting degenerative conditions.
* The model will be tested on multiple datasets, including publicly available lumbar spine MRI images, to ensure that it can generalize well across different image types and medical conditions.
* The system will undergo iterative improvements in to increase diagnostic accuracy, especially in challenging cases such as subtle conditions or poor-quality images.

**3.4 SCOPE OF THE PROJECT**

This project focuses specifically on diagnosing degenerative conditions of the lumbar spine using MRI scans. The system is designed to process MRI images in common formats, including DICOM and NIfTI, and identify key coordinates corresponding to degenerative regions of the spine.

The scope of the project is limited to the following:

* The system is not designed for diagnosing other spinal conditions, such as fractures or infections.
* The project will focus on MRI images and will not process other medical imaging modalities, such as X-rays or CT scans.
* The model will be trained and evaluated using a limited set of publicly available MRI datasets, which may not encompass all variations of degenerative conditions across diverse populations.

Despite these limitations, the system will be highly valuable for supporting medical professionals in diagnosing lumbar spine conditions, particularly in cases where rapid diagnosis is essential or in settings with limited access to specialized healthcare professionals.

**3.5 SUMMARY**

In summary, this chapter introduces the challenges faced in diagnosing lumbar spine degenerative conditions and outlines the objectives and scope of the project. The proposed deep learning-based system aims to automate and improve the accuracy of lumbar spine degeneration detection from MRI images, offering a promising solution to support healthcare professionals. By leveraging advanced deep learning techniques, such as CNNs and transfer learning, the system will contribute to more efficient, consistent, and scalable diagnostics, ultimately enhancing patient care and outcomes in the medical field.

**CHAPTER – 4**

**LITERATURE SURVEY**

**CHAPTER 4**

**LITERATURE SURVEY**

This chapter reviews existing research and techniques related to the automatic detection of lumbar spine degenerative conditions using medical imaging, particularly focusing on machine learning and deep learning techniques applied to MRI scans.

# 4.1 Automated detection, labelling and radiological grading of clinical spinal MRIs

Spinal magnetic resonance (MR) scans are a vital tool for diagnosing the cause of back pain for many diseases and conditions. However, interpreting clinically useful information from these scans can be challenging, time-consuming and hard to reproduce across different radiologists. In this paper, we alleviate these problems by introducing a multi-stage automated pipeline for analyzing spinal MR scans. This pipeline first detects and labels vertebral bodies across several commonly used sequences (e.g. T1w, T2w and STIR) and fields of view (e.g. lumbar, cervical, whole spine). Using these detections it then performs automated diagnosis for several spinal disorders, including intervertebral disc degenerative changes in T1w and T2w lumbar scans, and spinal metastases, cord compression and vertebral fractures. To achieve this, we propose a new method of vertebrae detection and labelling, using vector fields to group together detected vertebral landmarks and a language-modelling inspired beam search to determine the corresponding levels of the detections. We also employ a new transformer-based architecture to perform radiological grading which incorporates context from multiple vertebrae and sequences, as a real radiologist would. The performance of each stage of the pipeline is tested in isolation on several clinical datasets, each consisting of 66 to 421 scans. The outputs are compared to manual annotations of expert radiologists, demonstrating accurate vertebrae detection across a range of scan parameters. Similarly, the model’s grading predictions for various types of disc degeneration and detection of spinal metastases closely match those of an expert radiologist.

#### 4.2 **Landmarking and feature localization in spine x-rays**

The general problem of developing algorithms for the automated or computer-assisted indexing of images by structural contents is a significant research challenge. This is particularly so in the case of biomedical images, where the structures of interest are commonly irregular, overlapping, and partially occluded. Examples are the images created by digitizing film x-rays of the human cervical and lumbar spines. We have begun work toward the indexing of 17 000 such spine images for features of interest in the osteoarthritis and vertebral morphometry research communities. This work requires the segmentation of the images into vertebral structures with sufficient accuracy to distinguish pathology on the basis of shape, labeling of the segmented structures by proper anatomical name, and classification of the segmented, labeled structures into groups corresponding to high level semantic features of interest, using training data provided by biomedical experts. In this paper, we provide a technical characterization of the cervical spine images and the biomedical features of interest, describe the evolving technical approach for the segmentation and indexing problem, and provide results of algorithms to acquire basic landmark data and localization of spine regions in the images.

# 4.3 Automatic Detection and Measurement of Spinous Process Curve on Clinical Ultrasound Spine Images

#### The ultrasound (US) imaging technique has been applied to scoliosis assessment, and the proxy Cobb angle can be acquired on the US coronal images. The spinous process angle (SPA) is a valuable parameter to indicate 3-D deformity of spine. However, the SPA cannot be measured on US images since the spinous process (SP) is merged in the soft tissue layer and impossible to be identified on the coronal view directly. A new method based on the gradient vector flow (GVF) snake model was proposed to automatically locate SP position on the US transverse images, and the density-based spatial clustering of application with noise (DBSCAN) was used to remove the outliers out of the detected location results. With marking the SP points on the US coronal image, the SP curve was interpolated and the SPA was measured. The algorithm was evaluated on 50 subjects with various severity of scoliosis, and two raters measured the SPA on both US images and radiographs manually. The mean absolute differences (MADs) of SPAs obtained from the two modalities were 3.4° ± 2.4° and 3.6° ± 2.8° for the two raters, respectively, which were less than the clinical acceptance error (5°), and the results reported a good linear correlation ( r > 0.85) between the US method and radiography. It indicates that the proposed method can be a promising approach for SPA measurement using the US imaging technique.

# 4.4 Artificial Intelligence in Spinal Imaging: Current Status and Future Directions

#### Spinal maladies are among the most common causes of pain and disability worldwide. Imaging represents an important diagnostic procedure in spinal care. Imaging investigations can provide information and insights that are not visible through ordinary visual inspection. Multiscale in vivo interrogation has the potential to improve the assessment and monitoring of pathologies thanks to the convergence of imaging, artificial intelligence (AI), and radiomic techniques. AI is revolutionizing computer vision, autonomous driving, natural language processing, and speech recognition. These revolutionary technologies are already impacting radiology, diagnostics, and other fields, where automated solutions can increase precision and reproducibility. In the first section of this narrative review, we provide a brief explanation of the many approaches currently being developed, with a particular emphasis on those employed in spinal imaging studies. The previously documented uses of AI for challenges involving spinal imaging, including imaging appropriateness and protocoling, image acquisition and reconstruction, image presentation, image interpretation, and quantitative image analysis, are then detailed. Finally, the future applications of AI to imaging of the spine are discussed. AI has the potential to significantly affect every step in spinal imaging. AI can make images of the spine more useful to patients and doctors by improving image quality, imaging efficiency, and diagnostic accuracy.

**CHAPTER – 5**

**SOFTWARE REQUIREMENT SPECIFICATIONS**

**CHAPTER 5**

**SOFTWARE REQUIREMENT SPECIFICATIONS**

**5.1 Functional Requirements**

Functional requirements define the specific behaviors and actions the system must perform to meet the needs of its users. These requirements describe how the system will function when handling specific inputs, tasks, and outputs.

* **Data Input:** The system should allow users to upload MRI images in various formats such as .jpg, .mha, and .dcm. Users should also be able to submit images through batch processing.
* **Image Preprocessing:** The system should preprocess MRI images by converting them into a consistent format, normalizing pixel values, and performing necessary operations such as resizing and cropping to focus on relevant regions.
* **Region Extraction:** The system should detect and extract the lumbar spine region from the input MRI images using deep learning techniques, such as a pre-trained Convolutional Neural Network (CNN).
* **Coordinate Detection:** The system should identify and output the coordinates of regions in the lumbar spine, marking key points that may indicate degeneration or other conditions.
* **Model Prediction:** The system should classify the detected lumbar spine regions based on degenerative conditions, outputting the predicted condition (e.g., normal, mild degeneration, severe degeneration).
* **Model Training and Evaluation:** The system should allow for training models using labeled MRI data and provide functionality for evaluating the model’s accuracy and performance on a test set.
* **User Interface:** The system should feature a web-based interface, allowing users to upload MRI images and view the output prediction of degenerative conditions, along with visualizations of the detected regions.
* **Results Display:** After processing, the system should display the results of the condition prediction on a separate page, showing the degenerative classification, coordinates, and an overview of the image with relevant regions highlighted.

**5.2 Non-Functional Requirements**

Non-functional requirements define the overall quality attributes of the system. These focus on the performance, reliability, and user experience of the application.

* **Performance:** The system should process and identify lumbar spine regions in MRI images within a reasonable time frame, with optimized performance even when handling high-resolution images or large batches of data.
* **Accuracy:** The system should achieve high classification accuracy for degenerative conditions in the lumbar spine, with an expected accuracy rate above 85%. It should handle diverse MRI scan quality and detect even minor signs of degeneration.
* **Usability:** The web-based interface should be intuitive, easy to navigate, and designed for both medical professionals and non-technical users. The user interface should include step-by-step instructions, clear error messages, and accessible help sections.
* **Reliability:** The system must be stable and reliable, with low downtime and proper recovery mechanisms in case of errors. It should ensure continuous operation, even in the event of a network failure or unexpected input errors.
* **Maintainability:** The system should be built with maintainability in mind, enabling easy updates, bug fixes, and the integration of future enhancements without disrupting the existing functionality.

**CHAPTER – 6**

**SOFTWARE DESIGN**

**CHAPTER 6**

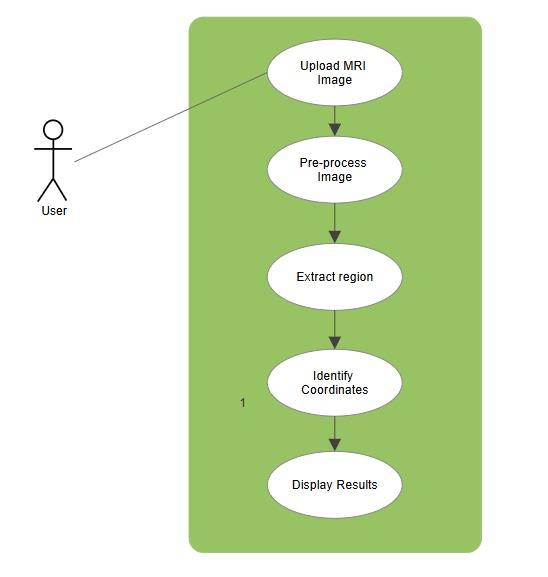
**SOFTWARE DESIGN**

**6.1 OVERVIEW**

This chapter outlines the software design for the "Design and Development of Language Identifier" project.

**6.2 USE CASE DIAGRAM**

The Use Case Diagram for the Lumbar Spine Degenerative Classification project illustrates the interactions between the User and the system. It outlines key functionalities, including uploading MRI images, preprocessing the data, detecting lumbar spine regions, and classifying degenerative conditions. The diagram highlights how radiologists and administrators interact with the system to analyze and classify spine-related issues efficiently.

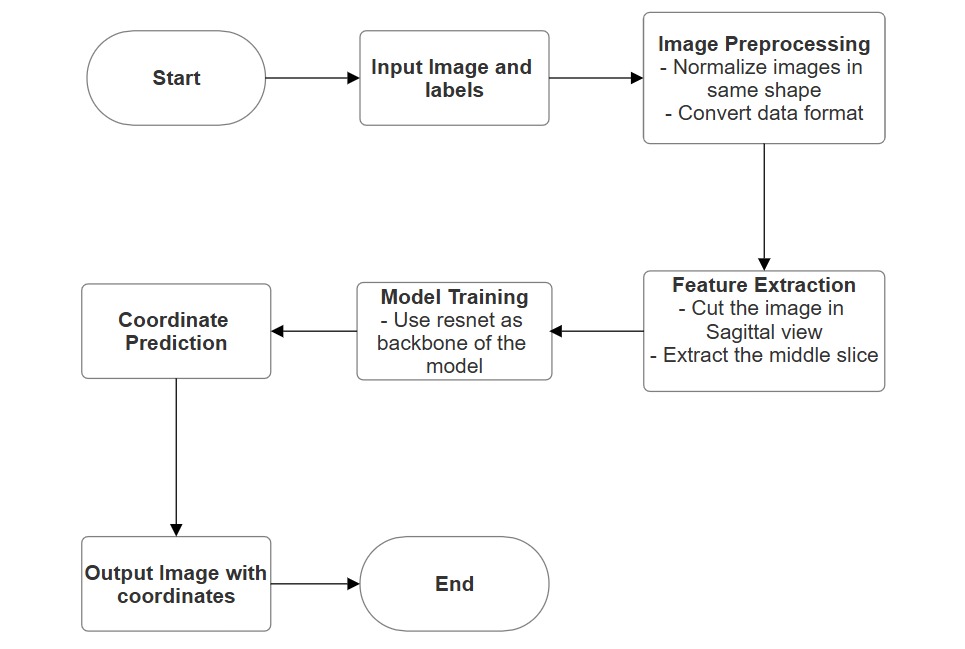


**Figure 6.1 Use Case Diagram**

**6.3 PROJECT FLOW DIAGRAM**

The Project Flow Diagram for the Lumbar Spine Degenerative Classification project provides a step-by-step view of the system’s processing workflow. It begins with input MRI image and follows through data preprocessing, feature extraction, model training, and coordinates prediction, ultimately leading to the coordinates of the target area.

5

****

**Figure 6.2: Project Flow Diagram**

**CHAPTER – 7**

**MACHINE LEARNING MODULE**

**CHAPTER 7**

**MACHINE LEARNING MODULE**

This chapter provides an overview of the Machine Learning module for the “Design and Development of Lumbar Spine Degenerative Classification”. It aims to explain the workflow and techniques used to build a robust system.

**7.1 DATASET DESCRIPTION**

**train.csv** Labels for the train set.

* study\_id - The study ID. Each study may include multiple series of images.
* [condition]\_[level] - The target labels, such as spinal\_canal\_stenosis\_l1\_l2, with the severity levels of Normal/Mild, Moderate, or Severe. Some entries have incomplete labels.

**train\_label\_coordinates.csv**

* study\_id
* series\_id - The imagery series ID.
* instance\_number - The image's order number within the 3D stack.
* condition - There are three core conditions: spinal canal stenosis, neural\_foraminal\_narrowing, and subarticular\_stenosis. The latter two are considered for each side of the spine.
* level - The relevant vertebrae, such as l3\_l4
* [x/y] - The x/y coordinates for the center of the area that defined the label.

**sample\_submission.csv**

* row\_id - A slug of the study ID, condition, and level such as 12345\_spinal\_canal\_stenosis\_l3\_l4.
* [normal\_mild/moderate/severe] - The three prediction columns.

**[train/test]\_images/[study\_id]/[series\_id]/[instance\_number].dcm** The imagery data.

**[train/test]\_series\_descriptions.csv**

* study\_id
* series\_id
* series\_description The scan's orientation.

**Source**: This dataset is publicly available on Kaggle and can be accessed and downloaded for research and educational purposes.

**Link:** https://www.kaggle.com/competitions/rsna-2024-lumbar-spine-degenerative-classification/data

**7.2 PREPROCESSING STEPS**

#### 7.2.1 IMAGE PREPROCESSING

Image preprocessing is the initial step in preparing raw MRI images for analysis, ensuring consistency and improving the quality of the input data. The following techniques are used in this project:

**DICOM to JPEG Conversion:** MRI images in DICOM format are converted to JPEG format for compatibility with the deep learning model. This conversion ensures that the images are suitable for processing and visualization.

**Resizing and Normalization:** Images are resized to a fixed dimension (e.g., 224x224 pixels) to standardize input size, making them compatible with the pre-trained models. Pixel values are normalized to a range of 0 to 1, improving numerical stability and accelerating the training process.

**Middle Slice Extraction:** For 3D MRI images, the middle slice of the scan is extracted. This slice often provides the most relevant information about the spine region, simplifying the analysis while retaining diagnostic details.

**Data Augmentation:** Techniques such as rotation, flipping, zooming, and contrast adjustment are applied to the images. These augmentations artificially expand the dataset, improving the model's ability to generalize to unseen data.

**7.2.2 FEATURE EXTRACTION**

Feature extraction involves transforming processed MRI images into a format suitable for deep learning models. The following steps are performed:

1. **Region of Interest (ROI) Detection:** The coordinates of the lumbar spine region are identified to isolate the area relevant for classification. This step reduces noise and focuses on the most informative parts of the image.
2. **Feature Maps via CNNs:** Convolutional Neural Networks (CNNs) extract hierarchical feature maps, capturing patterns such as edges, textures, and complex structures in the lumbar spine region. These feature maps are used to classify degenerative conditions.
3. **Dimensionality Reduction:** To manage computational complexity, only the most significant features are retained. This reduces redundancy, improves efficiency, and focuses on features critical to identifying degenerative conditions.

**7.3 ML MODEL ANALYSIS**

#### 7.3.1 DATA SPLITTING

The dataset is split into training and testing subsets. The training set comprises 80% of the data and is used for model learning, while the testing set (20%) is reserved for evaluation. This split ensures the model is validated on unseen data, reducing overfitting and improving generalization.

#### 7.3.2 MODEL SELECTION

A ResNet-18 architecture is chosen for this project. ResNet-18, a residual neural network, is selected for its proven performance in image classification tasks. The use of skip connections in ResNet helps mitigate vanishing gradient problems, making it effective for processing complex MRI data.

**7.3.3 MODEL TRAINING**

The ResNet-18 model is trained using the preprocessed MRI images. Key steps in training include:

1. **Loss Function:** Cross-entropy loss is used to measure the difference between predicted and actual labels, guiding the optimization process.
2. **Optimizer:** Adam optimizer is employed for its adaptive learning rate and efficiency in handling sparse gradients**.**
3. **Epochs and Batch Size:** The model is trained for a specified number of epochs, with a fixed batch size, to balance computational efficiency and learning stability.

**7.4 RESULT ANALYSIS**

**7.4.1 PREDICTION**

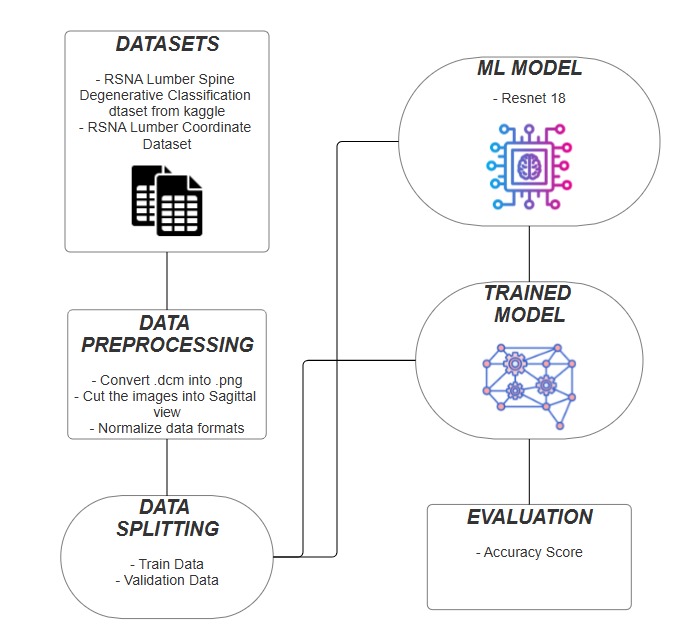
The trained ResNet-18 model is used to classify new MRI images into specific categories of degenerative conditions. The model applies the learned features to predict the presence and severity of degeneration in the lumbar spine.

**7.4.2 EVALUATION OF RESULTS**

The model's performance is evaluated by comparing its predictions with actual labels from the test dataset. Metrics such as accuracy, precision, recall, and F1 score are calculated to assess the model's reliability.

**7.4.3 ACCURACY SCORE**

The model achieves an accuracy score of 87%, indicating its effectiveness in classifying lumbar spine degenerative conditions. This score reflects the model's ability to identify patterns in MRI images and demonstrates its potential for aiding diagnostic processes in medical imaging.



**Figure 7.1: Software Life Cycle**

**CHAPTER – 8**

**FRONT-END**

**CONNECTIVITY**

**CHAPTER 8**

**FRONT-END CONNECTIVITY**

This chapter provides an overview of the front-end integration, detailing how users can interact with the model through these pages to make.

**8.1 HTML CODE**

<!-- templates/index.html -->

{% extends "base.html" %}

{% block content %}

<section class="intro">

    <h2>Welcome to the Lumbar Spine Classification Tool</h2>

    <p>Our tool allows you to upload MRI images and classify lumbar spine degenerative conditions with the power of AI.</p>

</section>

<section class="upload">

    <h3>Upload MRI Image</h3>

    <form action="/upload" method="post" enctype="multipart/form-data">

        <input type="file" name="file" accept=".dcm,.nii,.nmc,image/\*" required>

        <button type="submit">See Results</button>

    </form>

</section>

<section class="result">

    {% if result %}

        <h3>Classification Result:</h3>

        <p>{{ result }}</p>

    {% else %}

        <h3>Awaiting Image Upload...</h3>

    {% endif %}

</section>

{% endblock %}

**8.2 CONNECTIVITY**

import os

import cv2

import torch

import pydicom

import numpy as np

from PIL import Image

import io

import base64

import matplotlib.pyplot as plt

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

from werkzeug.utils import secure\_filename

import timm

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

# Define folder for saving uploaded files

UPLOAD\_FOLDER = 'uploads'

ALLOWED\_EXTENSIONS = {'dcm', 'nii'}

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

# Load the model

def load\_weights\_skip\_mismatch(model, weights\_path, device):

    state\_dict = torch.load(weights\_path, map\_location=device, weights\_only=False)

    model\_state\_dict = model.state\_dict()

    new\_state\_dict = {k: v for k, v in state\_dict.items() if k in model\_state\_dict}

    model\_state\_dict.update(new\_state\_dict)

    model.load\_state\_dict(model\_state\_dict)

model = timm.create\_model('resnet18', pretrained=True, num\_classes=10)

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

model = model.to(device)

weights\_path = r'E:\CodeSpace\Major Project\resnet18.pt'

load\_weights\_skip\_mismatch(model, weights\_path, device)

# Check allowed file extensions

def allowed\_file(filename):

    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS

# Convert DICOM to PNG

def convert\_dcm\_to\_png\_sagittal(dcm\_file\_path):

    dicom\_data = pydicom.dcmread(dcm\_file\_path)

    pixel\_array = dicom\_data.pixel\_array

    if len(pixel\_array.shape) == 3:

        middle\_index = pixel\_array.shape[0] // 2

        pixel\_array = pixel\_array[middle\_index]

    pixel\_array = (pixel\_array - pixel\_array.min()) / (pixel\_array.max() - pixel\_array.min()) \* 255

    pixel\_array = pixel\_array.astype(np.uint8)

    image = Image.fromarray(pixel\_array)

    return image

# Predict on the image

def predict(image, model, device, target\_size=(256, 256)):

    image = image.resize(target\_size)

    image\_rgb = np.array(image.convert("RGB"))

    image\_transposed = np.transpose(image\_rgb, (2, 0, 1)).astype(np.float32) / 255.0

    image\_tensor = torch.tensor(image\_transposed).unsqueeze(0).to(device)

    model.eval()

    with torch.no\_grad():

        preds = model(image\_tensor)

        preds = torch.sigmoid(preds)

    return preds.cpu().numpy()

# Plot predictions on the image

def plot\_predictions\_with\_labels(image, predictions):

    image\_rgb = np.array(image.convert("RGB"))

    for i in range(0, len(predictions[0]), 2):

        x = int(predictions[0][i] \* image\_rgb.shape[1])

        y = int(predictions[0][i + 1] \* image\_rgb.shape[0])

        cv2.circle(image\_rgb, (x, y), radius=10, color=(255, 0, 0), thickness=-1)

        label = f"({x}, {y})"

        font\_scale = 0.6

        thickness = 2

        cv2.putText(image\_rgb, label, (x + 10, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX, font\_scale, (0, 255, 0), thickness)

    plt.figure(figsize=(6, 6))

    plt.imshow(image\_rgb)

    plt.axis('off')

    # Save the plot to a static folder

    output\_image\_path = os.path.join('static', 'output\_image.png')

    plt.savefig(output\_image\_path, format="png")

    plt.close()  # Close the figure to avoid warnings

    return output\_image\_path  # Return the path instead of encoded image

# Flask routes

@app.route('/')

def home():

    return render\_template('index.html')

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

def upload\_image():

    if 'file' not in request.files:

        return "No file part"

    file = request.files['file']

    if file.filename == '':

        return "No selected file"

    if file and allowed\_file(file.filename):

        filename = secure\_filename(file.filename)

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

        file.save(input\_image\_path)

        try:

            image = convert\_dcm\_to\_png\_sagittal(input\_image\_path)

            preds = predict(image, model, device)

            output\_image\_path = plot\_predictions\_with\_labels(image, preds)

            output\_image\_path = plot\_predictions\_with\_labels(image, preds)

            output\_image\_path = output\_image\_path.replace("\\", "/")

            # Render result.html with the output image path

            return render\_template('result.html', output\_image\_path=output\_image\_path, prediction="Your model prediction here")

        except Exception as e:

            return f"Error processing image: {str(e)}"

    return redirect(url\_for('home'))

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

**CHAPTER – 9**

**CODING**

**CHAPTER 9**

**CODING**

This chapter provides an overview of the front-end integration, detailing how users can interact with the model through these pages to make.

**9.1 IMAGE PREPROCESSING**

def convert\_dcm\_to\_png\_sagittal(dcm\_file\_path, output\_folder):

# Load the DICOM file

dicom\_data = pydicom.dcmread(dcm\_file\_path)

pixel\_array = dicom\_data.pixel\_array

# Handle 2D and 3D data cases

if len(pixel\_array.shape) == 3:

slices, rows, cols = pixel\_array.shape

elif len(pixel\_array.shape) == 2:

rows, cols = pixel\_array.shape

slices = 1

pixel\_array = pixel\_array[np.newaxis, :, :] # Add a new axis for consistency

else:

raise ValueError("Unexpected DICOM data shape. Only 2D or 3D images are supported.")

# Normalize pixel values to the range [0, 255] without distortion

pixel\_array = pixel\_array.astype(np.float32) # Convert to float for normalization

pixel\_array = (pixel\_array - pixel\_array.min()) / (pixel\_array.max() - pixel\_array.min()) \* 255

pixel\_array = pixel\_array.astype(np.uint8) # Convert back to uint8

# Ensure the output folder exists

os.makedirs(output\_folder, exist\_ok=True)

saved\_image\_paths = [] # List to store paths of saved images

# Convert each slice to a PNG

for i in range(slices):

img = Image.fromarray(pixel\_array[i]) # Convert the array to an image

image\_path = os.path.join(output\_folder, f'slice\_{i}.png')

img.save(image\_path)

saved\_image\_paths.append(image\_path)

return saved\_image\_paths # Return the paths of saved images

**9.2 MODEL TRAINING**

for epoch in range(cfg.epochs + 1):

    # Training loop

    loss = torch.tensor([0.]).float().to(cfg.device)  # Initialize training loss as 0

    if epoch != 0:  # Skip training for epoch 0

        model = model.train()  # Set the model to training mode

        for batch in tqdm(train\_dl):  # Iterate over batches in the training data loader

            batch = batch\_to\_device(batch, cfg.device)  # Move batch data to the specified device

            optimizer.zero\_grad()  # Zero the gradients

            x\_out = model(batch["img"].float())  # Forward pass to get model output

            x\_out = torch.sigmoid(x\_out)  # Apply the Sigmoid function to map output to [0, 1]

            loss = criterion(x\_out, batch["label"].float())  # Compute the loss

            loss.backward()  # Backpropagate to compute gradients

            optimizer.step()  # Update model parameters

    # Validation loop

    val\_loss = 0  # Initialize validation loss as 0

    with torch.no\_grad():  # No gradient computation is needed during validation

        model = model.eval()  # Set the model to evaluation mode

        for batch in tqdm(val\_dl):  # Iterate over batches in the validation data loader

            batch = batch\_to\_device(batch, cfg.device)  # Move batch data to the specified device

            pred = model(batch["img"].float())  # Forward pass to get model predictions

            pred = torch.sigmoid(pred)  # Apply the Sigmoid function to map predictions to [0, 1]

            val\_loss += criterion(pred, batch["label"].float()).item()  # Compute and accumulate validation loss

        val\_loss /= len(val\_dl)  # Calculate average validation loss

    # Visualization

    visualize\_prediction(batch, pred, epoch)  # Visualize the current batch predictions

    print(f"Epoch {epoch + 1}, Training Loss: {loss.item()}, Validation Loss: {val\_loss}")

    # Print the current epoch's training loss and validation loss

print("Training completed...")

**9.3 PREDICTION AND EVALUATION**

def predict(image\_path, model, device, target\_size=(256, 256)):

"""

Predict the output for a single image using the model.

Args:

image\_path (str): Path to the image file.

model (torch.nn.Module): The trained model.

device (torch.device): The device (CPU or GPU) to run the model on.

target\_size (tuple): Target size for resizing the image.

Returns:

np.ndarray: Model prediction.

"""

# If image\_path is a list, take the first element (assuming it's a single file)

if isinstance(image\_path, list):

if len(image\_path) == 0:

print("Error: Provided list for image\_path is empty.")

return None

image\_path = image\_path[0] # Convert to string path

# Load and preprocess the image

image = cv2.imread(image\_path)

if image is None:

print(f"Error: Unable to load image at path: {image\_path}")

return None

# Resize, convert to RGB, and normalize

image\_resized = cv2.resize(image, target\_size)

image\_rgb = cv2.cvtColor(image\_resized, cv2.COLOR\_BGR2RGB)

image\_transposed = np.transpose(image\_rgb, (2, 0, 1)).astype(np.float32) / 255.0

image\_tensor = torch.tensor(image\_transposed).unsqueeze(0).to(device) # Add batch dimension and move to device

# Set the model to evaluation mode and make prediction

model.eval()

with torch.no\_grad():

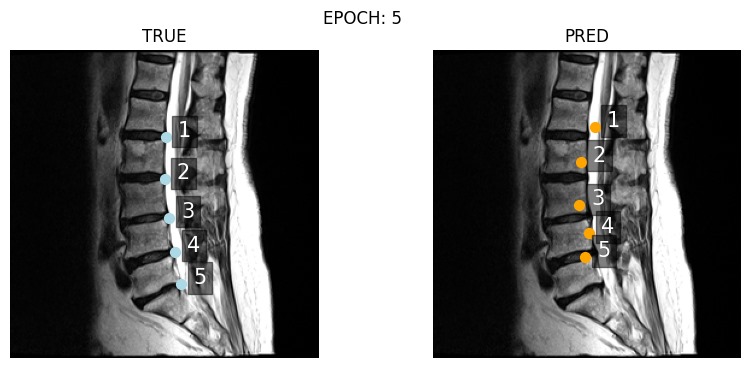
preds = model(image\_tensor)

preds = torch.sigmoid(preds) # Apply sigmoid for probability outputs

# Return the prediction (as numpy array)

return preds.cpu().numpy()

**OUTPUT:**

****

Epoch 6, Training Loss: 0.0005586033803410828, Validation Loss: 0.0032719985798134337

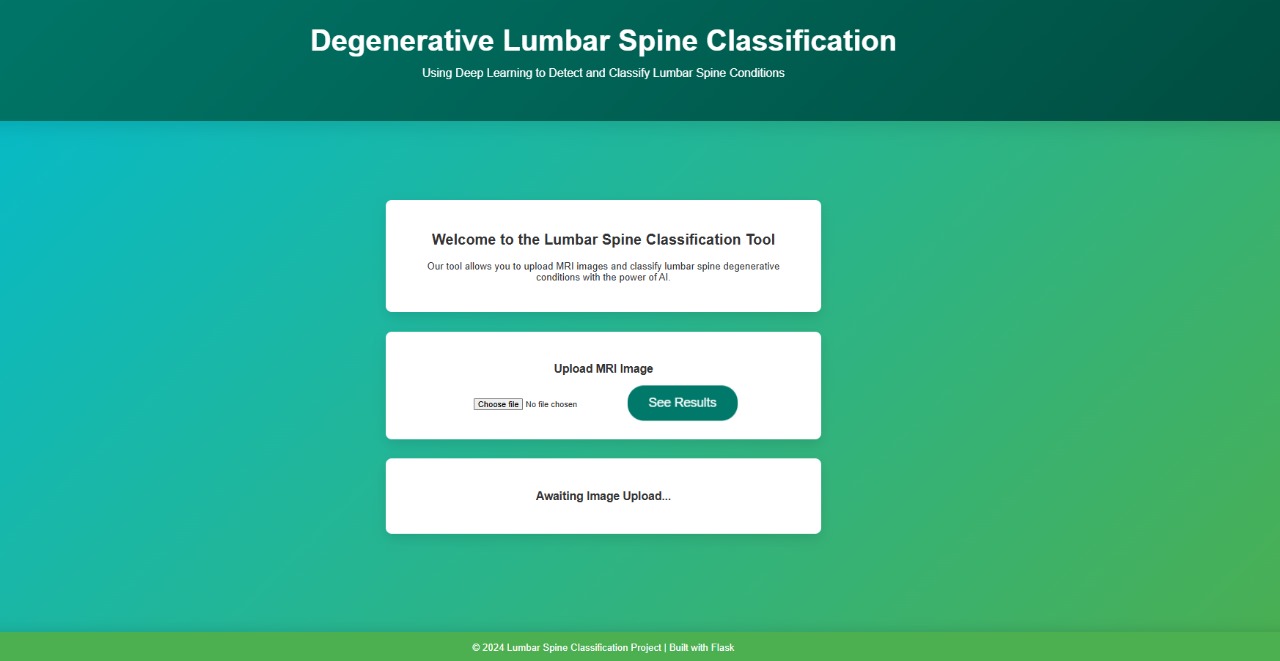
**CHAPTER – 10**

**OUTPUT**

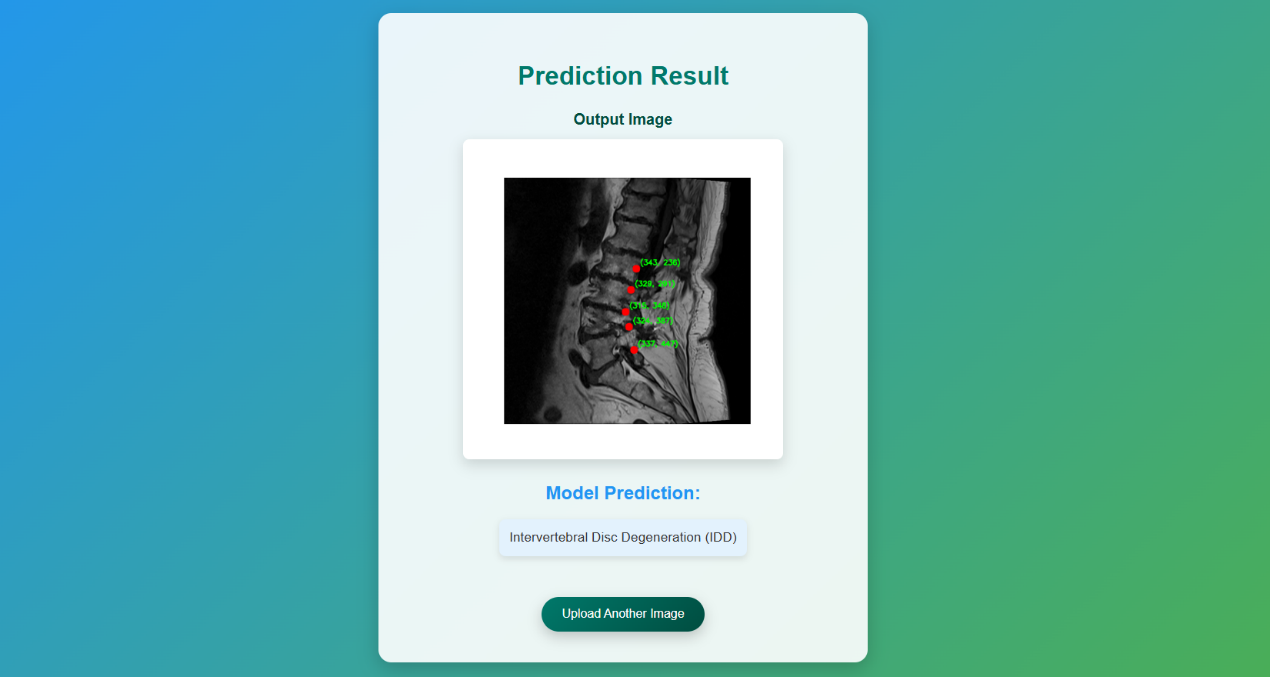
**SCREENS**

**CHAPTER 10**

**OUTPUT SCREENS**

****

**Figure 10.1: Home Page**

****

**Figure 10.2: Prediction Page**

**APPENDIX-1**

**GLOSSARY OF TERMS**

**(In alphabetical order)**

|  |  |
| --- | --- |
| **B** |  |
| **Bounding Box** | **Bounding Box is** a rectangular region in an image used to specify the location of the object of interest, such as specific areas in the lumbar spine in MRI images. |
| **C** |  |
| **CSS** | Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a corner stone technology of the World Wide Web, alongside HTML. |
| **CNN** | A deep learning model primarily used in image processing, adapted in NLP to capture spatial hierarchies and patterns in text data for tasks like sentence classification and language identification. |
| **D** |  |
| **DICOM** | Digital Imaging and Communications in Medicine, a standardized format for storing, sharing, and transmitting medical imaging data, such as MRI scans. |
| **F** |  |
| **Flask** | **Flask is a** lightweight and versatile web framework for building web applications. It provides essential tools and libraries for developing the backend of the application, such as handling HTTP requests, rendering templates, and integrating machine learning models. |
| **H** |  |
| **HTML** | The standard language used to create and structure content on the web. It defines the elements and layout of web pages through tags and attributes. |
| **I** |  |
| **IDE** | A software application that provides tools for software development, including a code editor, debugger, and compiler. It simplifies the process of writing, testing, and debugging code. |
| **L** |  |
| **Lumbar Spine** | The lower region of the spine, consisting of five vertebrae, often susceptible to degenerative conditions such as herniated discs or stenosis. |
| **M** |  |
| **MRI** | Magnetic Resonance Imaging, a medical imaging technique used to visualize internal structures of the body, such as the lumbar spine, in detail. |
| **P** |  |
| **Preprocessing** | The series of steps taken to clean and prepare raw data for analysis or modeling. |
| **R** |  |
| **Resnet** | A deep learning architecture used in this project to analyze and classify lumbar spine MRI images by learning residual connections to improve performance. |
| **ROI** | Specific areas in the MRI image (e.g., vertebrae or discs) that are analyzed for signs of degeneration. |
| **S** |  |
| **Spinal Degeneration** | The gradual wear and tear of spinal discs and vertebrae, leading to conditions like disc herniation or stenosis, which this project aims to classify. |
| **T** |  |
| **Timm** | A built-in Python library used to handle time-related tasks. It provides functions to measure program execution time, pause execution for a specified duration, and fetch the current system time |
| **U** |  |
| **UI** | The space where interactions between humans and machines occur. It includes the design and layout of elements like buttons, menus, and screens that allow users to interact with software or hardware in a way that is intuitive and efficient. |
| **V** |  |
| **VS CODE** | An open-source code editor developed by Microsoft which supports a wide range of programming languages and features such as debugging, syntax highlighting, version control integration, etc. |

**REFERENCES**

**JOURNALS / RESEARCH PAPERS**

1. Windsor, R., Jamaludin, A., Kadir, T. et al. Automated detection, labelling and radiological grading of clinical spinal MRIs. Sci Rep 14, 14993 (2024). <https://doi.org/10.1038/s41598-024-64580-w>

2. L. Rodney Long, George R. Thoma, "Landmarking and feature localization in spine x-rays," J. Electron. Imag. 10(4) (1 October 2001) <https://doi.org/10.1117/1.1406503>

3. H. -Y. Zeng, E. Lou, S. -H. Ge, Z. -C. Liu and R. Zheng, "Automatic Detection and Measurement of Spinous Process Curve on Clinical Ultrasound Spine Images," in IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 68, no. 5, pp. 1696-1706, May 2021, doi: 10.1109/TUFFC.2020.3047622.3.6. Zampieri, M., Malmasi, S., Nakov, P., Ali, A., Shon, S., & Glass, J. (2020). Predicting the languages of the world: A data-driven approach. \*Proceedings of the 14th Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis\*, 56-65.

4. Cui, Y.; Zhu, J.; Duan, Z.; Liao, Z.; Wang, S.; Liu, W. Artificial Intelligence in Spinal Imaging: Current Status and Future Directions. Int. J. Environ. Res. Public Health 2022, 19, 11708. <https://doi.org/10.3390/ijerph191811708>

**PROJECT SUMMARY**

***About Project***

|  |  |
| --- | --- |
| **Title of the project** | Degenerative Lumbar Spine Classification |
| **Semester** | 7th |
| **Members** | 1. Mahak Mirza |
| 1. Sparsh Sahu 2. Eashan Tiwari |
| **Team Leader** | Sparsh Sahu |
| **Describe role of every member in the project** | Sparsh Sahu: Model Trainer + Backend  Mahak Mirza: Model Trainer + Frontend  Eashan Tiwari: Backend |
| **What is the motivation for selecting this project?** | The motivation for selecting this project arises from the critical need to improve the early detection of degenerative spine conditions using advanced deep learning techniques. Manual analysis of MRI scans is time-consuming and prone to errors, making it essential to develop an automated system. This project aims to leverage deep learning to assist medical professionals in accurately identifying degenerative lumbar spine conditions, ultimately enhancing diagnostic efficiency and patient care. |
| **Project Type**  **(Desktop Application, Web Application, Mobile App, Web)** | Web Application |

***Tools &Technologies***

|  |  |
| --- | --- |
| **Programming language**  **used** | Python |
| **Compiler used**  **(with version)** | NA |
| **IDE used**  **(with version)** | Microsoft Visual Studios Code (1.51.1) |
| **Front End Technologies**  **(With version, wherever**  **Applicable)** | HTML & CSS |
| **Back End Technologies**  **(with version, wherever**  **applicable)** | Flask (Python) |
| **Database used**  **(with version)** | NA |

***Software Design& Coding***

|  |  |
| --- | --- |
| **Is prototype of the software developed?** | NA |
| **SDLC model followed**  **(Waterfall, Agile, Spiral etc.)** | Agile Methodology |
| **Why above SDLC model is followed?** | Agile model has a set of guidelines that are: small, highly motivated project team and supports changing requirements. We need both guidelines to develop our project. |
| **Justify that the SDLC model mentioned above is followed in the project.** | Since the demand (functionalities) of the website kept on changing every now and then therefore we used the Agile model, so that we could make desired changes whenever needed. |
| **Software Design approach followed**  **(Functional or Object-oriented)** | Functional oriented approach |
| **Name the diagrams**  **developed**  **(According to the Design**  **approach followed)** | Class Diagram, Data Flow Diagram & Use Case Diagram |
| **In case Object Oriented**  **approach is followed, which of the OOPS principles are**  **covered in design?** | NA |
| **No. of Tiers**  **(example 3-tier)** | NA |
| **Total no. of front-end pages** | 2 |
| **Total no. of tables in database** | NA |
| **Database is in which Normal Form?** | NA |
| **Are the entries in database encrypted?** | NA |
| **Front end validations applied (Yes / No)** | No |
| **Session management done**  **(in case of web applications)** | NA |
| **Is application browser compatible**  **(in case of web applications)** | Yes |
| **Exception handling done**  **(Yes / No)** | No |
| **Commenting done in code**  **(Yes / No)** | Yes |
| **Naming convention followed**  **(Yes / No)** | Yes |
| **What difficulties faced during deployment of project?** | * Faced issues in finding appropriate and diverse dataset. * Faced issues in handling medical image dataset |
| **Total no. of Use-cases** | 4 |
| **Give titles of Use-cases** | 1. Input Image 2. Model Analysis 3. Detect Coordinate 4. Show Result |

***Project Requirements***

|  |  |  |
| --- | --- | --- |
| **MVC architecture followed**  **(Yes / No)** | | Yes |
| **If yes, write the name of**  **MVC architecture followed**  **(MVC-1, MVC-2)** | **MVC-2** | |
| **Design Pattern used**  **(Yes / No)** | No | |
| **If yes, write the name of**  **Design Pattern used** | No | |
| **Interface type**  **(CLI / GUI)** | GUI | |
| **No. of Actors** | 1 | |
| **Name of Actors** | User | |
| **Total no. of Functional**  **Requirements** | 6 | |
| **List few important non-**  **Functional Requirements** | Accuracy, Maintainability, Performance, Compatibility | |

***Testing***

|  |  |
| --- | --- |
| **Which testing is performed?**  **(Manual or Automation)** | Manual |
| **Is Beta testing done for this**  **project?** | No |

***Write project narrative covering above mentioned points***

|  |
| --- |
| The **Lumbar Spine Degenerative Classification** project aims to assist healthcare professionals by leveraging deep learning to analyze MRI images and identify degenerative spine conditions. The system processes uploaded MRI images, extracts relevant features, and uses a trained model to predict affected spine regions and conditions. Users can view the results with annotated spine areas and download the processed images for further analysis. This project addresses the need for efficient, accurate, and automated diagnostic tools in medical imaging, improving diagnosis workflows and patient outcomes. |

|  |  |  |  |
| --- | --- | --- | --- |
| Mahak Mirza | Sparsh Sahu | Eashan Tiwari | Guide Signature |
| 0187AD211020 | 0187AD211039 | 0187AD2110015 | Prof. Ruchi Jain |