****

**SOFTWARE REQUIREMENTS**

**SPECIFICATION (SRS)**

***for***

**FractoScan-"Automated Detection and Classification of Orthopedic Fracture Patterns Using Statistical Learning”**

Prepared by

|  |  |  |  |
| --- | --- | --- | --- |
| S. No | Roll Number | Student Name | |
| 1. | 23N31A6695 | G. AMULYA | |
| 2. | 23N31A66B0 | G. ABHINAV | |
| 3. | 23N31A66C8 | K. YOGENDRA BALAJI | |
|  |  |  | |
|  |  | |  |

|  |  |
| --- | --- |
| Supervisor: | Mr. MOHAMMAD AZHAR |
| Designation: | Assistant Professor |
| Department: | CSE(AIML) |
| Batch ID: | 23CIIP1B05 |
| Date: |  |
| Supervisor Sign. & Date |  |

**Department of Computational Intelligence**



**Title of the Project: FractoScan-"Automated Detection and Classification of Orthopedic Fracture Patterns Using Statistical Learning”**

# Content

Contents I[i](#__RefHeading___Toc397335649)

Revisions I[ii](#__RefHeading___Toc397335649)

1 Introduction [1](#__RefHeading___Toc397335650)

1.1 Document Purpose [1](#__RefHeading___Toc397335651)

1.2 ProJECt Scope [1](#__RefHeading___Toc397335652)

1.3 Existing Systems 2

1.4 Problems with Existing Systems 2

1.5 Proposed Systems 3

1.6 Advantages of Proposed Systems 5

2 Overall Description 6

2.1 Feasibility Study 6

2.2 Product Functionality 7

2.3 Design and Implementation Constraints 8

2.4 Assumptions and Dependencies 9

3 Functional Requirements 10

3.1 Software Requirement Specifications 10

3.2 Hardware Requirements Specifications 10

3.3 Use Case Model 10

4 Other Non-functional Requirements 13

4.1 Performance Requirements 13

4.2 Safety and Security Requirements 13

4.3 Software Quality Attributes 14

5 Other Requirements 16

5.1 Database Requirements..................................................................................................................16

5.2 Internationalization Requirements............................................................................................16

5.3 Legal Requirements........................................................................................................................16

5.4 Reuse Objectives...............................................................................................................................17

5.5 Development Environment Requirements...................................................................................17

5.6 Documentation Requirements.......................................................................................................17

6 REFERENCES 18

# Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
|  |  |  |  |

# Introduction

## The project detects and classifies bone fractures from X-ray images. It uses Convolutional Neural Networks (CNNs) trained on the MURA dataset (7 bone types).Preprocessing improves image quality before training. CNN automatically learns features.The system provides the Fracture severity estimation,Grad-CAM visualization (highlights fracture areas), Automated reports and treatment suggestions, Makes diagnosis faster, accurate, and helpful for doctors.

## Document Purpose

This Software Requirements Specification (SRS) document defines the requirements for **“FractoScan – Automated Detection and Classification of Orthopedic Fractures using Statistical Learning.”** The main purpose of this document is to describe how the system detects and classifies bone fractures from X-ray images using Convolutional Neural Networks (CNNs).

It outlines the system’s main functions, including image preprocessing, fracture classification, severity estimation, and automated report generation. This document serves as a guide for developers and stakeholders to understand the system’s goals and ensure that the project is developed efficiently to provide accurate and faster fracture diagnosis,supportive for patients

The **FractoScan** system aims to assist doctors and radiologists by providing an automated and reliable method for fracture detection. By using deep learning and image processing, it reduces manual effort, minimizes diagnostic errors, and speeds up the overall decision-making process in orthopedic analysis.

## Project/Product Scope

This project aims to develop an intelligent **FractoScan** system that automatically detects and classifies **bone fractures from X-ray images** using **Convolutional Neural Networks (CNNs)**. It utilizes the **MURA dataset**, which contains seven bone types — finger, elbow, hand, forearm, humerus, wrist, and shoulder — divided into **training and validation sets** to ensure accurate and generalized model performance. The system applies **image preprocessing techniques** such as resizing, normalization, and augmentation to enhance image quality before training.

The main goal of this project is to assist doctors and radiologists in **accurately identifying fracture types and severity levels** through automated analysis, reducing manual effort and diagnostic errors. It integrates multiple modules, including **data preprocessing, CNN-based classification, Grad-CAM visualization**, and **automated report generation**, all within a user-friendly interface built using **Streamlit**.

The scope of the project includes:

* **Image Input Module** – Allows doctors to upload X-ray images for analysis.
* **Preprocessing Module** – Performs image resizing, normalization, and augmentation to improve model accuracy.
* **Fracture Classification** – Uses CNNs to detect and classify bone fractures into respective categories.
* **Severity Estimation** – Determines the extent of fracture severity for better treatment planning.
* **Grad-CAM Visualization** – Highlights the affected fracture areas on X-ray images for interpretability.
* **Automated Report Generation** – Produces structured diagnostic reports with treatment suggestions.
* **Performance Evaluation** – Measures model accuracy, reliability, and prediction time using validation data.
* **Future Enhancements** – Can be extended to handle multi-class classification, larger datasets, and real-time X-ray scanning integration in hospitals.

This project aims to make **orthopedic diagnosis faster, more accurate, and efficient** through deep learning automation. By integrating medical image analysis with modern visualization techniques, **FractoScan** provides a reliable and interactive platform that enhances clinical decision-making and supports healthcare professionals in fracture detection and treatment planning.

**1.3 Existing System:**

**1.3.1 Manual Diagnosis:**

X-ray images are examined visually by medical professionals to detect and classify fractures.

**1.3.2 Previous Machine Learning Models:**

• SVM (Support Vector Machine): Improved accuracy over manual diagnosis by separating bone classes using decision boundaries.

•k-NN (k-Nearest Neighbors): Classified bones based on feature similarity, giving simple and quick results.

•These methods worked better than manual checking but required manual feature extraction and could not handle complex fracture patterns.

## 1.4 Problems with Existing System:

* Uses classical machine-learning algorithms such as SVM, k-NN, Random Forest and ANN to assist fracture detection.
* Relies on manual feature extraction from X-ray images to prepare data for these algorithms.
* Designed primarily for handling standard or simpler fracture patterns.
* Provides basic classification results without integrated severity estimation, visualisation or automated reporting.
* Requires separate tools for image analysis and report preparation, rather than a single unified interface.

## 1.5 Proposed System

* Still uses CNN for bone fracture detection, but with improved preprocessing and training.
* Provides better accuracy by focusing on bone-specific characteristics.
* Extends beyond classification with severity estimation, automated reports, and treatment suggestions.
* Includes patient-focused features like follow-up reminders, rehab tracking, and alert system.
* Makes the system more practical, supportive, and useful in real medical workflows.

**1.6 Advantages of Proposed System**

* Uses Convolutional Neural Networks (CNNs) for fracture detection and classification.
* Automatically learns features from X-ray images without manual preprocessing.
* Achieves higher accuracy and better generalization than traditional ML methods.
* Goes beyond classification by providing severity estimation, automated reports, and treatment suggestions.

# Overall Description

## Feasibility Study

The feasibility study evaluates the practicality of implementing **FractoScan – Automated Detection and Classification of Orthopedic Fracture Patterns Using Statistical Learning (CNNs).**This assessment covers **technical, economic, operational, and schedule feasibility** to ensure successful project development and deployment.

**2.1.1 Technical Feasibility**

* **Technology Availability:**The proposed system utilizes well-established **deep learning and image-processing technologies** such as **Convolutional Neural Networks (CNNs), OpenCV, TensorFlow**, and **Grad-CAM visualization**.
* **Computational Requirements:**The model can be efficiently trained and executed on systems with **moderate hardware specifications** (e.g., Intel i5/i7 processors and GPU support). Once trained, the model can perform rapid fracture detection on standard hospital or educational computers.
* **Data Availability:**Publicly available medical image datasets like **MURA** provide high-quality, diverse X-ray samples for training and evaluation, ensuring sufficient data for model development and validation.
* **Scalability:**The system design supports future enhancements such as adding **new bone types, multi-class fracture severity classification, or integration with real-time hospital imaging systems**, demonstrating long-term scalability.

**2.1.2 Economic Feasibility**

* Compared to large-scale commercial diagnostic systems, **FractoScan** can be developed and deployed using **open-source frameworks** like TensorFlow, Streamlit, and OpenCV, making it affordable for academic and institutional use.
* The reliance on **free software tools and public datasets** reduces licensing and maintenance costs, ensuring an economically viable solution.
* By streamlining the detection and reporting process, the system can help **reduce manual workload**, **speed up diagnosis**, and **assist in educational training**, offering long-term cost and time savings for institutions.

## **2.1.3 Feasibility Scheduling**

* **Project Timeline:**The **FractoScan system was successfully designed, developed, and tested in four months.**

The project followed a streamlined process covering:

* **Data preprocessing and model setup**
* **CNN model training and accuracy evaluation**
* **Grad-CAM visualization integration**
* **User interface development using Streamlit**
* **Testing, documentation, and presentation preparation**

## Product Functionality

The **FractoScan system** is designed to analyze orthopedic X-ray images, automatically detect fracture patterns, estimate their severity, and generate clear visual and textual diagnostic outputs.

**2.2.1 Image Input and Preprocessing**

* **X-ray Image Upload:** Users such as doctors, students, or researchers can upload X-ray images for analysis.Supports common formats like JPG, PNG, and DICOM.
* **Preprocessing:** Enhances image clarity and contrast.Normalizes and resizes images for consistent input.Uses data augmentation (rotation, flipping) to improve model accuracy.

**2.2.2 Fracture Detection and Classification**

* **Automatic Detection:** The CNN model identifies fracture areas by learning bone structure patterns from the images.
* **Bone Classification:** Classifies X-rays by bone type (e.g., wrist, elbow, shoulder) and detects if a fracture is present.

**2.2.3 Severity Estimation and Visualization**

* **Severity Analysis:** Estimates how severe the fracture is based on its pattern and spread.
* **Grad-CAM Visualization:** Generates heatmaps highlighting the exact fracture region for better understanding.

**2.2.4 Automated Report Generation**

* **Diagnostic Report:** Creates a structured report showing bone type, fracture status, and severity level.
* **Consistent Output:** Produces accurate, uniform reports for every analyzed image automatically.

**2.2.5 User Interfae:**

* **Interactive Control:** A simple **Streamlit-based interface** lets users upload X-rays, view results, and generate reports easily.
* **Visual Display:** Shows the original image, processed image, and heatmap side by side for clear comparison.

****2.2.6 Optimization and Future Scope****

* **Model Optimization:** Fine-tuned CNN ensures high accuracy with efficient performance.
* **Future Enhancements:** Can be expanded for multi-class fracture detection, real-time use, and larger datasets.

## Design and Implementation Constraints

The development of **FractoScan**, an automated system for orthopedic fracture detection and classification using CNNs, involves several **design and implementation constraints.**These constraints ensure the system remains efficient, accurate, and user-friendly.

**2.3.1 Design Constraints**

**Image Complexity and Size**

* High-resolution X-ray images may increase processing time and memory usage.
* Images with overlapping bones or multiple fracture lines require advanced handling for accurate detection.

**Computational Constraints**

* Deep learning models like CNNs require GPUs or systems with good processing power for faster analysis.
* Memory optimization is needed to handle large image datasets efficiently during training and testing.

**Visualization Constraints**

* Heatmaps and visual overlays must clearly highlight fracture regions without obstructing important image details.
* The user interface layout should ensure clarity, readability, and smooth navigation between input and output images.

**Detection Accuracy Constraints**

* The system must handle variations in X-ray brightness, angle, and image quality while maintaining accuracy.
* Ensuring consistent performance across different bone types and fracture patterns is essential.

**Model Generalization Constraints**

* The CNN must be trained on diverse datasets to avoid overfitting and ensure accurate predictions for new, unseen images.

**2.3.2 Implementation Constraints**

**Hardware Constraints**

* Moderate processing power and memory are required to visualize scripts efficiently.
* High-resolution displays may be needed for clear GUI visualization of program states.

**Software Constraints**

* with Python versions commonly used in educational and professional environments.
* Dependence on GUI frameworks (Tkinter, PyQt) and visualization libraries (Matplotlib, Compatible Graphviz).

**Data Privacy and Security Constraints**

* For scripts containing sensitive information, the visualizer must ensure local execution and data safety.
* Avoid storing or transmitting user code without explicit consent.

**Deployment and Integration Constraints**

* Must be compatible with standard operating systems (Windows, macOS, Linux).
* Optional integration with online learning platforms or coding environments for educational purposes.

**2.3.3 Design Constraints – User Interaction Constraints**

**Usability and Learning Curve**

* The interface must be intuitive and easy to use for beginners and advanced users alike.
* Navigation controls (step forward, step back, pause, jump to line) should be responsive and clearly labeled to avoid confusion during code visualization.

This ensures the system is not only technically sound but also user-friendly for learners and developers.

## Assumptions and Dependencies

The development of **FractoScan** for automated detection and classification of orthopedic fracture patterns is based on several **assumptions and dependencies.**These factors influence the system’s performance, usability, and deployment effectiveness.

**2.4.1 Assumptions:**

* · **Image Input & Quality Assumptions**

X-ray images provided to the system are clear, properly oriented, and of sufficient resolution for accurate analysis.

Images are pre-scanned and verified to be free from severe noise or artifacts that could affect detection.

· **Detection & Visualization Assumptions**

The CNN model can correctly identify bone regions and highlight fracture areas using Grad-CAM visualizations.

The system assumes that the input images correspond to one of the trained bone categories (e.g., wrist, elbow, shoulder).

· **Computational and Hardware Assumptions**

The system operates on a computer with adequate memory and processing capability (e.g., Intel i5/i7 with GPU support).

Execution and prediction times are within acceptable limits for real-time or near real-time image analysis.

· **User and Operational Assumptions**

Users such as doctors, medical students, or researchers can interpret visual heatmaps and report outputs correctly.

The system is used as a **decision-support or educational tool**, assisting human analysis rather than replacing professional judgment.

**2.4.2 Dependencies**

· **Data Dependencies**

Access to reliable and standardized **X-ray datasets** (e.g., MURA) for training, validation, and testing.

Availability of **diverse bone and fracture samples** to ensure model accuracy and generalization.

· **Algorithmic and Software Dependencies**

Dependence on **open-source libraries** such as TensorFlow, OpenCV, NumPy, and Streamlit for model training and visualization.

Availability of stable machine learning frameworks and pre-trained model architectures to ensure efficient implementation.

* **Hardware and Deployment Dependencies**

Requires systems with sufficient **GPU or high-performance CPU** for faster training and prediction.

The deployed environment should support Python 3.x and required dependencies without compatibility issues.

# Functional Requirements

## Software Requirement Specifications

The classification of fractured bones for step-by-step execution requires the following software components:

•**Operating System**: Windows 11

• **Programming Language**: Python 3.7+

• **Libraries:** Streamlit,Tensorflow,Opencv-python,Numpy,Pandas,Scikit-learn,Matplotlib

• **Integrated Development Environment**: Jupyter Notebook, VS Code 1.103 version.

## 3.2 Hardware Requirements Specifications

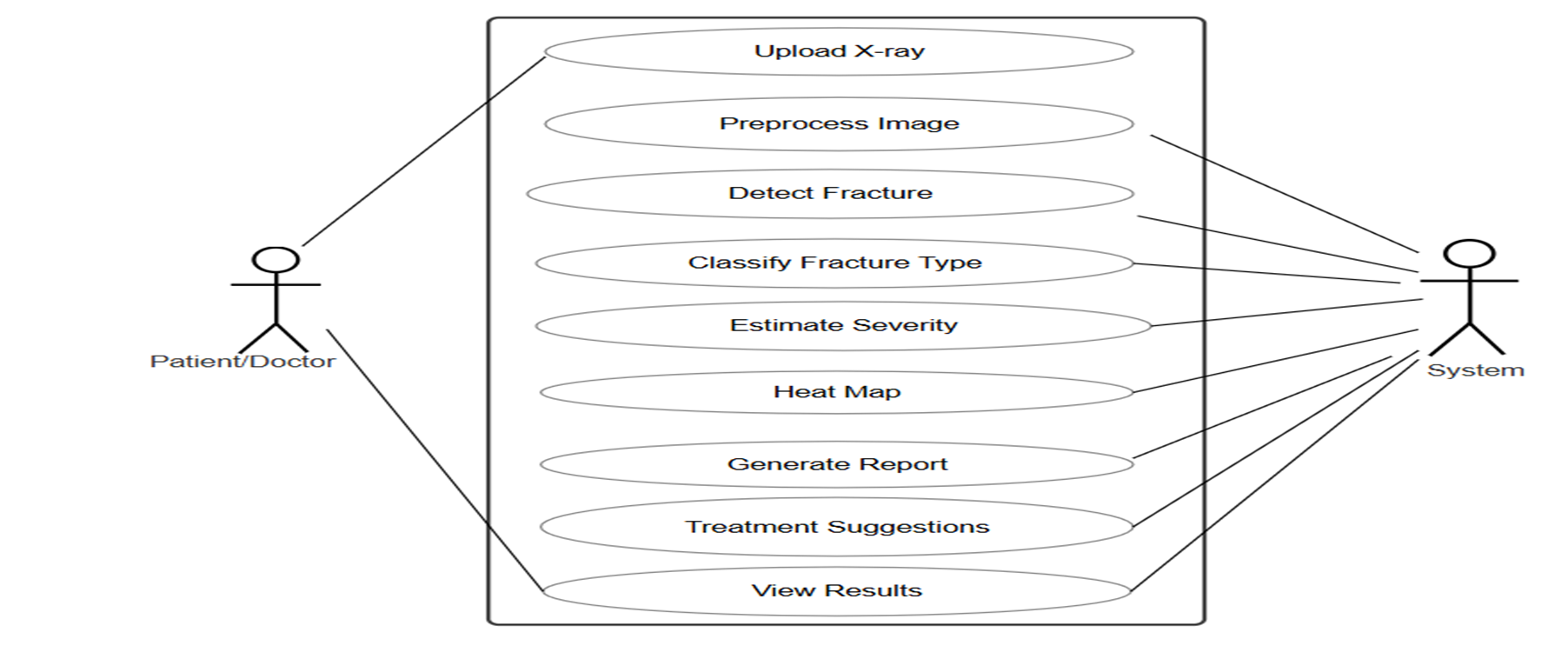
1.**Processor:** Intel Core i5/i7 (Quad-Core, 2.4 GHz )

2.**RAM:** 16 GB DDR

3.**Storage:** 256 GB SSD

4.**GPU:** NVIDIA GPU with CUDA support

## 3.3 Use Case Model



### ****Use Case :****

**Author:** Team FractoScan  
**Purpose:**  
This Use Case Diagram provides an overview of how a user interacts with the FractoScan system to automatically detect and classify orthopedic fractures from X-ray images using CNN-based analysis.

**Requirements Traceability:**

R1: X-ray image input and preprocessing

R5: Automated fracture detection and classification

R10: Fracture severity estimation

R14: Visualization through Grad-CAM heatmaps

R20: Automated report generation

**Priority:** High  
**Preconditions:** The user has uploaded a valid X-ray image.  
**Postconditions:** The system displays the classification result, severity level, heatmap visualization, and diagnostic report.

**Actors:** User (Radiologist or Medical Staff)  
**Extends:** N/A

**Flow of Events**

**1. Basic Flow**

The user uploads an X-ray image through the FractoScan interface.

The system preprocesses the image (resizing, normalization, noise removal).

CNN model analyzes the image to detect and classify any fractures.

Severity estimation is performed and Grad-CAM visualization highlights the fracture area.

The system generates and displays an automated diagnostic report with results and visualization.

**2. Alternative Flow**

The user can select a specific bone type for focused analysis.

The user may review previous reports stored in the system

The user can download or share the generated report securely.

**3. Exceptions**

If the image format or quality is invalid, the system prompts for re-upload.

If processing fails due to incomplete data, an error message is displayed.

For unclear cases, the system may flag the result as “requires review.”

**4. Includes**

Image preprocessing and enhancement module.

CNN-based fracture detection and classification.

Grad-CAM visualization module.

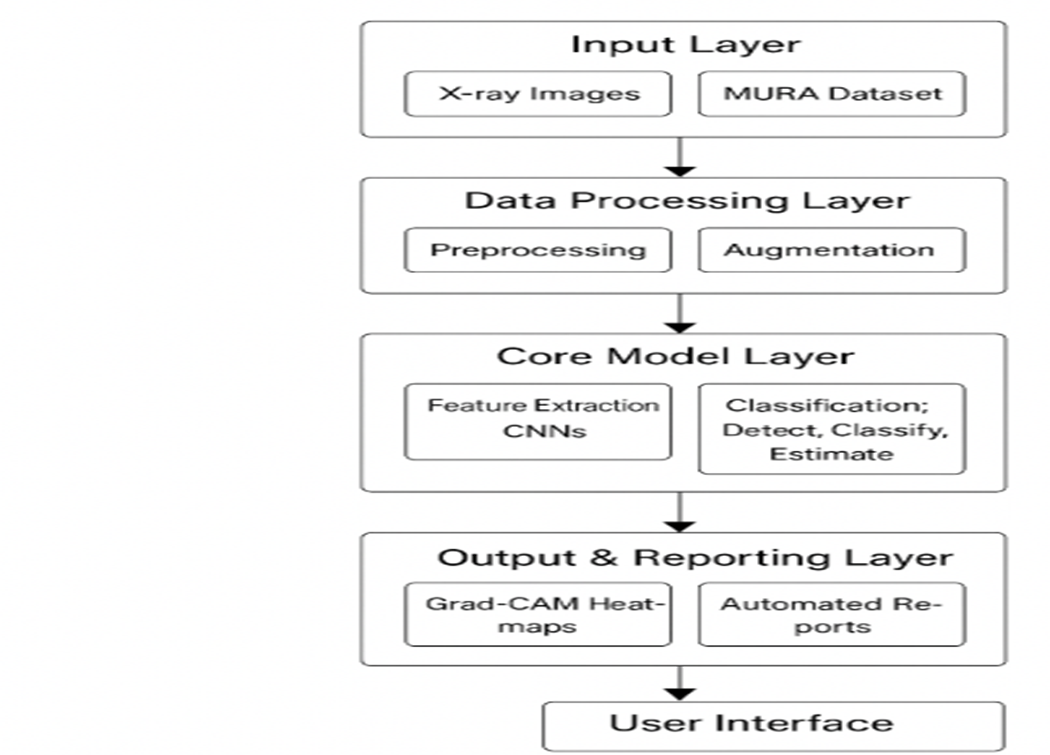
Automated report generation system.

**5. Notes/Issues**

Future versions may include integration with hospital management systems for direct report transfer.

Cloud-based real-time diagnosis and multi-bone fracture detection may be added later.

### **System Archtecture**



# Other Non-functional Requirements

#### ****4.1 Performance Requirements****

**Accuracy and Sensitivity**

* The system should achieve **at least 90% accuracy** in detecting and classifying bone fractures.
* Sensitivity should remain high to ensure that even minor fractures are correctly identified.

**Image Processing and Preprocessing**

* Supports standard X-ray image formats (JPG, PNG).
* Performs preprocessing like **resizing,normalization and augmentation** to improve image clarity and model accuracy.

**AI Model Performance**

* The **CNN model** should detect fractures with high confidence using the **training and validation sets** from the split dataset.
* Inference time should be **under 5 seconds per X-ray** for near real-time diagnosis.

**Computational Requirements**

* Supports **GPU acceleration (CUDA)** for faster training and prediction.
* Uses memory efficiently to handle multiple image analyses simultaneously.

**Integration and Compatibility**

* Compatible with hospital systems for report generation and image upload.
* Can be integrated into existing diagnostic tools or used as a standalone application.

**User Interface and Reporting**

* Provides **Grad-CAM heatmaps** for visual interpretation of fracture areas.
* Automatically generates structured diagnostic reports with **severity level and treatment suggestions**.

**Compliance and Security**

* Ensures patient data confidentiality with **local file handling**.
* Maintains high reliability and uptime during report generation

### ****4.2 Safety and Security Requirements****

**Patient Data Privacy and Security**  
All X-ray images and reports are processed locally to protect patient information.  
Only authorized users can upload and access medical data.

**System Reliability and Fault Tolerance**  
Ensures stable performance during continuous use.  
Automatically saves reports and results to prevent data loss.

**AI Model Safety and Bias Reduction**  
CNN model is trained on diverse data from the MURA dataset to reduce bias.  
Uses Grad-CAM visualization to provide clear and explainable results.

**Secure Image Handling**  
Accepts only valid X-ray formats (JPG, PNG) for analysis.  
Validates and preprocesses images before model prediction.

**Error Handling and Risk Management**  
Displays alerts for uncertain or low-confidence predictions.  
Maintains error logs for future model improvements

## 4.3 Software Quality Attributes

#### ****Performance Efficiency****

* **Fast Processing:** Each X-ray image should be analyzed and classified within **5 seconds**.
* **Optimized Resource Usage:** The system should efficiently use **CPU and GPU**resourcetraining and inference.
* **Scalability:** Can handle large X-ray datasets and multiple image uploads without performance issues.

#### ****Accuracy and Reliability****

* **High Accuracy:** CNN should achieve at least **90% accuracy** in fracture detection and classification.
* **Consistency:** Results should remain stable across training and validation datasets.
* **Minimal Downtime:** The system should maintain reliable uptime during medical use.

#### ****Security and Privacy****

* **Data Protection:** All uploaded X-rays remain stored locally and are not shared externally.
* **Access Control:** Only authorized medical staff can upload and view diagnostic reports.

#### ****Maintainability and Modularity****

* **Modular Design:** System components like preprocessing, CNN model, Grad-CAM, and reporting are independent and easy to update.
* **Code Documentation:** Well-commented code for better readability, testing, and collaboration.

#### ****Usability and User Experience****

* **User-Friendly Interface:** Built using **Streamlit**, allowing doctors to upload X-rays and view results easily.
* **Clear Visualization:** Grad-CAM heatmaps highlight fracture areas clearly for better understanding.
* **Instant Report Generation:** Automatically provides structured diagnostic reports with severity details.

#### ****Interoperability and Integration****

* **Hospital Integration:** Can be linked with local hospital systems for faster X-ray uploads.
* **Dataset Compatibility:** Supports standard image formats (JPG, PNG) and integrates easily with new datasets.
* **API Support (Future):** Open for future integration with external medical tools.

# Other Non-functional Requirements

#### ****5.1 Database Requirements****

* **X-ray Images:** Stored in JPG/PNG format from the MURA dataset, divided into training and validation folders.
* **Fracture Details:** Includes bone type, fracture category, and severity level.
* **Reports:** Stores automatically generated diagnostic reports and prediction results.
* **User Data (Optional):** Doctor name, patient ID, and upload date for reference and tracking.

#### ****5.2 Internationalization Requirements****

* **Dataset Diversity:** Model trained on diverse bone X-rays to handle different bone structures and patient variations.
* **Character Encoding:** Uses UTF-8 for multilingual support in the Streamlit interface.
* **Data Privacy:** Ensures patient data and X-ray images remain confidential and locally stored.

#### ****5.3 Legal Requirements****

* **Medical Use Compliance:** The system is designed as a diagnostic assistance tool, not a replacement for doctors.
* **Data Usage:** Only publicly available MURA dataset images are used for model training.
* **User Consent:** Users must confirm image ownership or permission before upload.

#### ****5.4 Reuse Objectives****

* **Model Reuse:** The trained CNN can be fine-tuned on other medical image datasets.
* **Cross-domain Application:** The same architecture can be extended to detect other bone or organ abnormalities.
* **Reusable Preprocessing Pipeline:** Steps like normalization and augmentation can be reused in future medical AI projects.

#### ****5.5 Development Environment Requirements****

* **Programming Stack:** Python 3.7+, TensorFlow, OpenCV, NumPy, Pandas, Matplotlib, Streamlit.
* **Version Control:** Source code managed using GitHub.
* **Model Testing:** Uses accuracy metrics and validation sets for model performance evaluation.
* **Deployment:** Can be hosted locally or on cloud platforms (AWS/GCP) for remote use.

#### ****5.6 Documentation Requirements****

* **Code Documentation:** Each module (preprocessing, CNN model, Grad-CAM, reporting) is clearly commented.
* **System Architecture:** Includes workflow, dataset structure, and CNN layer design.
* **User Guide:** Provides steps to upload X-rays, view heatmaps, and download reports.
* **Release Notes:** Lists updates such as accuracy improvements or interface enhancements in each version.

# References

* **MURA Dataset – Stanford University:** A large dataset of musculoskeletal radiographs used for training and validating the CNN fracture detection model.
* **Convolutional Neural Networks (CNNs):** Research papers and tutorials on deep learning architectures for medical image classification.
* **TensorFlow & Keras Documentation:** Official libraries used for model building, training, and evaluation.
* **OpenCV and NumPy:** Used for image preprocessing tasks like resizing, normalization, and augmentation.
* **Grad-CAM Visualization Techniques:** Studies on model interpretability and highlighting key image regions in deep learning.
* **Streamlit Framework:** Documentation for building the interactive user interface for uploading X-rays and viewing results

# · SRS DOCUMENT REVIEW

# CERTIFICATION

This Software Requirement Specification (SRS) Document is reviewed and certified to proceed for the project development by the Departmental Review Committee (DRC).

|  |  |
| --- | --- |
| Date of SRS Submitted: |  |
| Date of Review: |  |
| Supervisor Comments: |  |
| Supervisor Sign. & Date. |  |
| Coordinator Sign. & Date |  |
| HOD Sign. & Date |  |
| Dept. Stamp |  |

**Guidelines :**

* Font Style for the Content Page is ”Times New Roman”.
* Font Style for the Remaining Pages is “Calibri(Body)”.
* Font size for Headings is 14 size

(Example – 4.3 Software Quality Attributes).

* Font size for Sub-Headings is 12 size

(Example – 4.3.1 **Usability**).

* Headings and Sub-Headings must be in bold letters.
* Font size for the content is 12 size.
* Line Spacing value is 1.15.