Prepared by : Ayesha Arab

Software Requirements Specification (SRS)

COVID-19 Detection System with Continuous Learning

1. Introduction

1.1 Purpose

The purpose of this system is to develop an AI-based **COVID-19 detection system** that can analyze chest X-ray images and classify them as **COVID-positive** or **Normal**. The system will also have a **continuous learning feature**, meaning it can improve itself automatically when new images are added.

1.2 Scope

This system will:

- Allow users to upload **chest X-ray images**.
- Use a **trained deep learning model** to classify images.
- Provide a confidence score for each prediction.
- Continuously learn from newly added images without requiring full retraining.
- Offer a simple **web interface using Streamlit**.

1.3 Audience

This document is intended for:

- Developers working on the AI model.
- Healthcare professionals interested in AI-based diagnostics.
- Researchers studying AI applications in medical imaging.

2. Functional Requirements

2.1 Image Upload & Preprocessing

- Users can upload **JPEG, PNG, or JPG** images.
- The system will resize and normalize images before making predictions.

2.2 Model Prediction

- The system loads a trained deep learning model.
- It classifies an image as **COVID-positive** or **Normal**.
- Provides a confidence percentage for each prediction.

2.3 Continuous Learning

- If new images are added to the **new_data/** folder, the model will retrain itself automatically.
- The model updates without starting the training from scratch.
- User feedback (if implemented) will help the model improve over time.

2.4 Web Interface

- Users can upload images through a **simple web interface**.
- The interface displays results, including predictions and confidence scores.
- It shows a message for **COVID-positive** or **Normal** classifications.

3. Non-Functional Requirements

3.1 Performance

- The system should predict results **within a few seconds**.
- Training updates should be completed **within a reasonable time** (e.g., minutes to an hour).

3.2 Usability

- The web interface should be **simple and easy to use**.
- Users should not need technical knowledge to upload images.

3.3 Security

- The system should not store patient data.
- Images should be processed in real-time and not saved.

3.4 Compatibility

- The system should work on **Windows, Linux, and macOS**.
- It should support **Google Colab** for training and **VS Code** for local deployment.

4. System Architecture

4.1 Project Folder Structure

covid_19_detection/

```
├── dataset/            # Initial dataset (COVID & Normal images)
├── new_data/           # New images for continuous learning
├── model/              # Stores trained AI model
├── app.py              # Streamlit web app for real-time prediction
├── train.py            # Model training & continuous learning script
├── requirements.txt    # Dependencies
├── utils.py            # Helper functions (preprocessing, data loading)
```

4.2 Technologies Used

- **Deep Learning Framework**: TensorFlow/Keras
- **Frontend**: Streamlit (Python-based UI)
- **Data Processing**: OpenCV, NumPy, Pandas

5. User Interaction

5.1 Steps to Use the System

1. **Upload an X-ray image** via the web interface.
2. **Get instant results** (COVID-positive or Normal) with a confidence score.
3. **Add new images** to the dataset for continuous learning.
4. The model automatically updates with **new data**.

6. Deployment Plan

6.1 Local Deployment

- Run the Streamlit app using:
- streamlit run app.py
- Train the model with:
- python train.py

7. Future Enhancements

- **User Feedback Learning**: Allow users to correct wrong predictions.
- **Transfer Learning**: Improve accuracy using **pretrained models like ResNet, VGG16**.

8. Conclusion

This AI-based **COVID-19 detection system** provides a simple yet powerful way to classify chest X-ray images and continuously improve over time. The system is lightweight, easy to deploy, and capable of self-learning as new images are added.