Software Requirements Specification (SRS)

**Project Title:** Deepfake Detection System using MesoNet

**Version:** 1.0

**Date:** April 12, 2025

**Prepared by:** The DeepFake Detection Team

# Table of Contents

* 1. Introduction
* 1.1 Purpose
* 1.2 Scope
* 1.3 Definitions, Acronyms, and Abbreviations
* 1.4 References
* 2. Overall Description
* 2.1 Product Perspective
* 2.2 Product Features
* 2.3 User Classes and Characteristics
* 2.4 Operating Environment
* 2.5 Design and Implementation Constraints
* 2.6 Assumptions and Dependencies
* 3. Specific Requirements
* 3.1 Functional Requirements
* 3.2 Non-Functional Requirements
* 4. External Interface Requirements
* 4.1 User Interfaces
* 4.2 Software Interfaces
* 4.3 Communications Interfaces
* 5. System Models
* 5.1 Use Case Diagrams
* 5.2 Data Flow Diagrams
* 5.3 ER Diagrams
* 6. Appendices
* 6.1 Glossary
* 6.2 Acknowledgements

# 1. Introduction

**1.1 Purpose**

The purpose of this Software Requirements Specification (SRS) document is to outline the functional and non-functional requirements of the Deepfake Detection System using the MesoNet architecture. This project aims to provide a web-based solution for real-time detection of facial image manipulation through deep learning-based methods. The SRS serves as a foundation for software development and will be referenced throughout the **system design, implementation, testing, and maintenance phases.**

**1.2 Scope**

The Deepfake Detection System focuses on static image analysis and uses a convolutional neural network (CNN) called MesoNet to classify images as real or fake. The application supports image upload, face detection, and classification using MesoNet. It also offers awareness and educational modules, including a gallery and quiz.

**1.3 Definitions, Acronyms, and Abbreviations**

Deepfake: Digitally manipulated media using AI

MesoNet: CNN architecture for deepfake detection

CNN: Convolutional Neural Network

UI: User Interface

API: Application Programming Interface

HTTPS: Secure web protocol

Flask: Python web framework

PostgreSQL: Relational database system

**1.4 References**

- Afchar et al., 2018. MesoNet: a Compact Facial Video Forgery Detection Network

- TensorFlow API Docs

- Flask Documentation

- OpenCV Documentation

- Python 3.11 Documentation

# 2. Overall Description

**2.1 Product Perspective**

The Deepfake Detection System is a standalone web application using Flask for the backend and TensorFlow for model inference. It is built for deployment on cloud platforms and supports interaction via web browsers.

**2.2 Product Features**

- Image upload interface

- Face detection preprocessing using OpenCV

- Deepfake classification with confidence score via MesoNet

- Result display with session tracking

- Educational gallery and awareness quiz

**2.3 User Classes and Characteristics**

- General Users: Basic familiarity with web apps

- Students: Intermediate ML/AI understanding

- Educators: Knowledgeable in AI concepts

- Developers: Able to integrate APIs

**2.4 Operating Environment**

- Browser: Chrome, Firefox, Safari

- Backend: Python 3.11+, Flask, Gunicorn

- ML Libraries: TensorFlow/Keras, OpenCV

- Database: PostgreSQL 14+

**2.5 Design and Implementation Constraints**

- Face-only images

- Max file size: 5MB

- MesoNet pre-trained model used

- Responsive UI design required

**2.6 Assumptions and Dependencies**

- Pre-trained model available

- Internet connectivity assumed

- Libraries compatible with server OS

# 3. Specific Requirements

**3.1 Functional Requirements**

FR1.1 - Secure image upload

FR1.2 - Format and size validation

FR2.1 - Face detection using OpenCV

FR3.1 - MesoNet classification

FR3.2 - Result display with confidence

FR4.1 - Result storage in temporary session

FR4.2 - Educational module with quiz

FR4.3 - Gallery page with examples

**3.2 Non-Functional Requirements**

- Performance: <5 sec classification

- Accuracy: >=85%

- Security: HTTPS, session-based data handling

- Usability: Responsive interface

- Reliability: Graceful error handling

# 4. External Interface Requirements

**4.1 User Interfaces**

- Homepage: Intro and purpose

- Upload: Image selection form

- Results: Display classification

- Gallery: Real vs Fake image samples

- Quiz: Awareness assessment

- About: Team and project info

**4.2 Software Interfaces**

- PostgreSQL: Temporary session data

- OpenCV: Face preprocessing

- TensorFlow/Keras: Model inference

- Flask: Routing and session handling

**4.3 Communications Interfaces**

- HTTP/HTTPS: Client-server communication

- REST API: Internal endpoints

# 5. System Models

**5.1 Use Case Diagrams**

- Users upload → System preprocesses → Model classifies → Result displayed

**5.2 Data Flow Diagrams (DFD)**

- L0: Upload → Preprocess → Classify → Display

- L1: Upload form → Face crop → Model predict → Show result

**5.3 ER Diagrams**

- Tables: Users, Sessions, QuizResults

- One user → many sessions, many quiz attempts

# 6. Appendices

**6.1 Glossary**

Forgery: Deceptively modified image or video

Session: Temporary user interaction tracking

Inference: Prediction from a trained model

**6.2 Acknowledgements**

Thanks to Afchar et al. for MesoNet.

Thanks to open-source contributors of TensorFlow, OpenCV, Flask.