

---

# Design Document

for

# TraceFake: AI-Based Image Authenticity Checker

Prepared By: Hamza Shafiq

<version 2.1>

## Table of Contents

<b>1- Introduction to Design Document</b>	-----	<b>Page. 2</b>
<b>2- Entity Relation Diagram</b>	-----	<b>Page. 4</b>
<b>3- Sequence Diagrams</b>	-----	<b>Page. 6</b>
3.1 Sequence Diagram for End Users		
3.2 Sequence Diagram for Admin		
<b>4- Architecture Design Diagram</b>	-----	<b>Page. 9</b>
<b>5- Class Diagram</b>	-----	<b>Page. 11</b>
<b>6- Dataset Designs</b>	-----	<b>Page. 13</b>
6.1 Which datasets are being used?		
6.2 List of datasets and their sources		
6.3 Reason for opting these datasets		
<b>7- Interface Diagrams</b>	-----	<b>Page. 16</b>
<b>8- <u>Test Cases</u></b>	-----	<b>Page. 21</b>

## Revision History

Name	Date	Reason For Changes	Version
	11/01/26	Entity Relation Diagram Updated	1.1
	13/01/26	Reverse Image Search added in Class Diagram	1.2
	15/01/26	Sequence Diagram from Admin has also been added along with the End User	1.3
	16/01/26	Orientation of pages with Sequence Diagram changed to landscape from portrait	2.0
	17/01/26	Non-functional requirements added in test cases. 6 core FRs also divided into two tests; one for success and one for failure	2.1

# 1- Introduction to Design Document

*This design document presents a detailed and comprehensive design for TraceFake: AI Based Authenticity Checker. It's a standalone software tool. Various components are crucial for the development of this project. This document serves as a blueprint. A blueprint for the developers and tester ensuring that the project is well understood. It also ensures that the project is meticulously planned. The primary purpose is to provide a blueprint of system architecture, data structures, interactions and implement decisions before moving towards coding and testing phase.*

*The design document serves the following key objectives:*

- 1. To translate the functional and non-functional requirements stated in the SRS document into a technical structure.*
- 2. To define the high level and detailed design elements. By doing so implementation process can be carried out in a systematic way.*
- 3. To facilitate the communication with the intended audience by clearly documenting how the system will work.*
- 4. To support future maintenance, extension or replication.*
- 5. To ensure traceability between design and final deliverable.*

*This document focuses on logical, architectural and data design of TraceFake. It does not contain any detailed source code.*

## 1.1 Scope of Design Document:

*The design covers the core functionality of TraceFake as defined in the project proposal:*

- 1. Uploading an image (local file or URL).*
- 2. Classifying the image as real or AI-generated using a CNN-based model.*
- 3. Extracting and analyzing EXIF metadata.*
- 4. Performing optional Error Level Analysis (ELA) for tampering detection.*
- 5. Displaying results with confidence scores and supporting forensic cues.*
- 6. Admin panel for viewing logs, statistics, and updating the CNN model version.*

## 1.2 Structure of Design Document:

*The document is organized into the following main sections:*

### 1.2.1 System Architecture

*High-level layered/block diagram showing the overall structure (Frontend, Application Layer, Admin & Storage).*

### 1.2.2 Use Case View

*UML Use Case Diagram illustrating main actors (End User, Admin) and use cases.*

### 1.2.3 Data Model / Entity-Relationship Diagram (ERD)

*Database schema design for storing analysis records, extracted metadata, model versions, and system logs (using SQLite or similar lightweight database).*

#### **1.2.4 Sequence Diagrams**

*Dynamic interaction flows:*

- *Verify Image Authenticity ( End User workflow)*
- *Update or Retrain CNN Model (Admin workflow)*

#### **1.2.5 Class Diagram**

*Static structure showing main classes, attributes, operations, and relationships (including analysis components, result aggregation, and model management).*

#### **1.2.6 Dataset Design**

*Description of training and validation datasets used for the CNN model, including sources, sizes, rationale for selection, and balance between real and AI-generated images.*

#### **1.2.7 Test Cases**

*Description of functional and non-functional test cases.*

*Entity Relation Diagram*

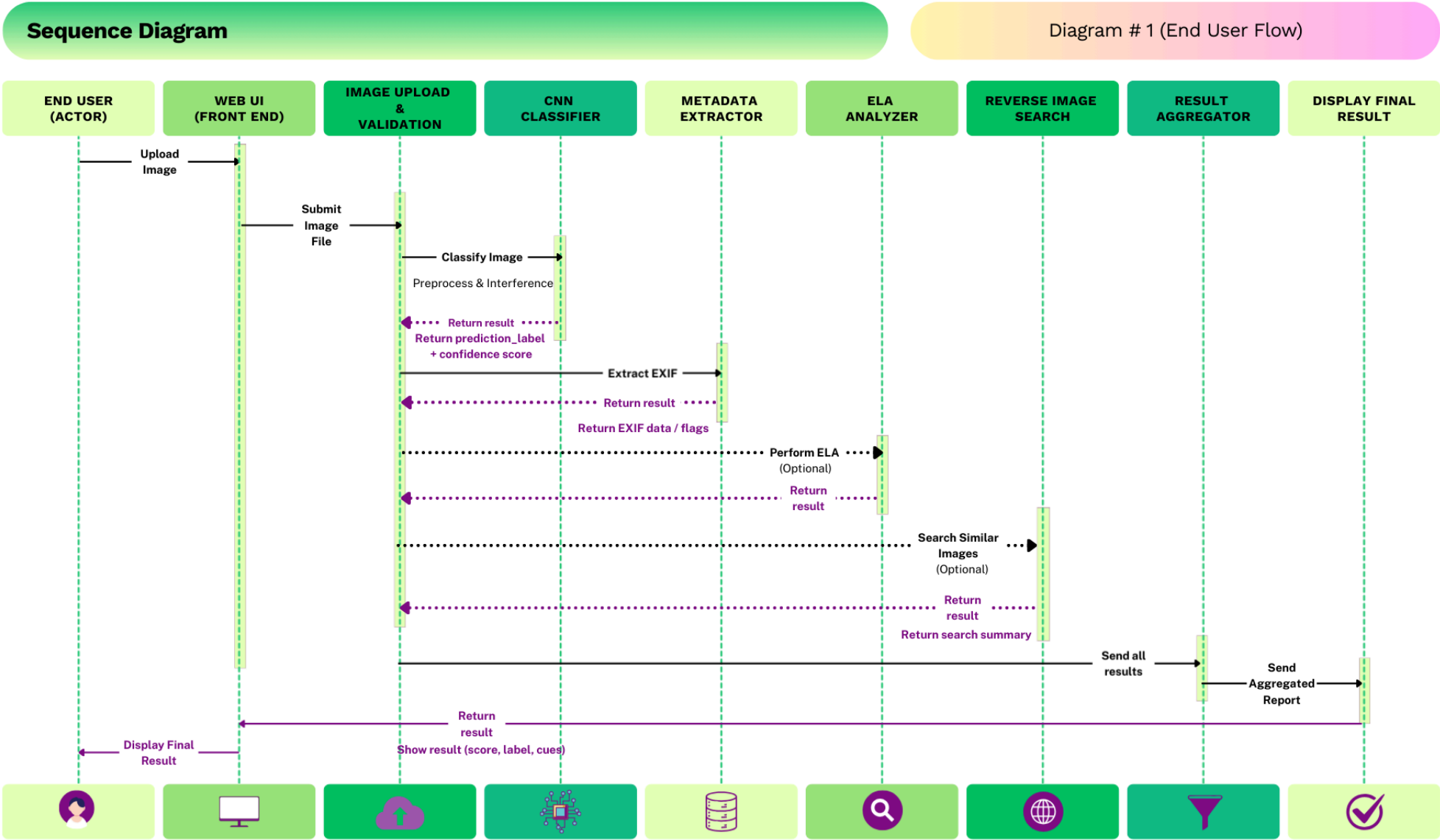
**ENTITY  
RELATION  
DIAGRAM  
(ERD)**

# Entity Relationship Diagram



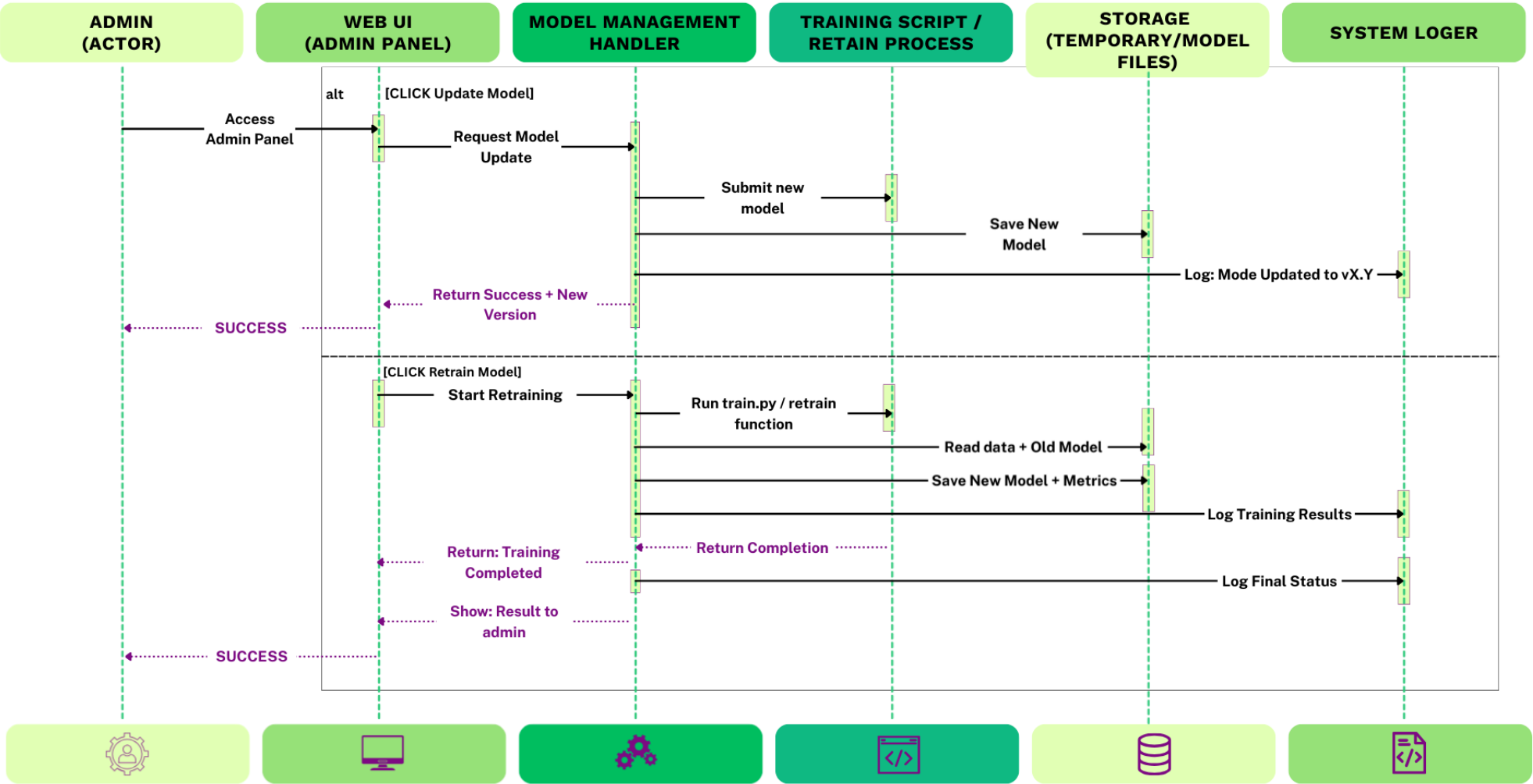
# SEQUENCE DIAGRAMS

THIS SECTION CONSISTS OF TWO DIAGRAMS.  
THE FIRST ONE IS THE MAIN DIAGRAM SHOWING END USER  
FLOW WHILE THE SECOND DIAGRAM SHOWS ADMIN ACCESS



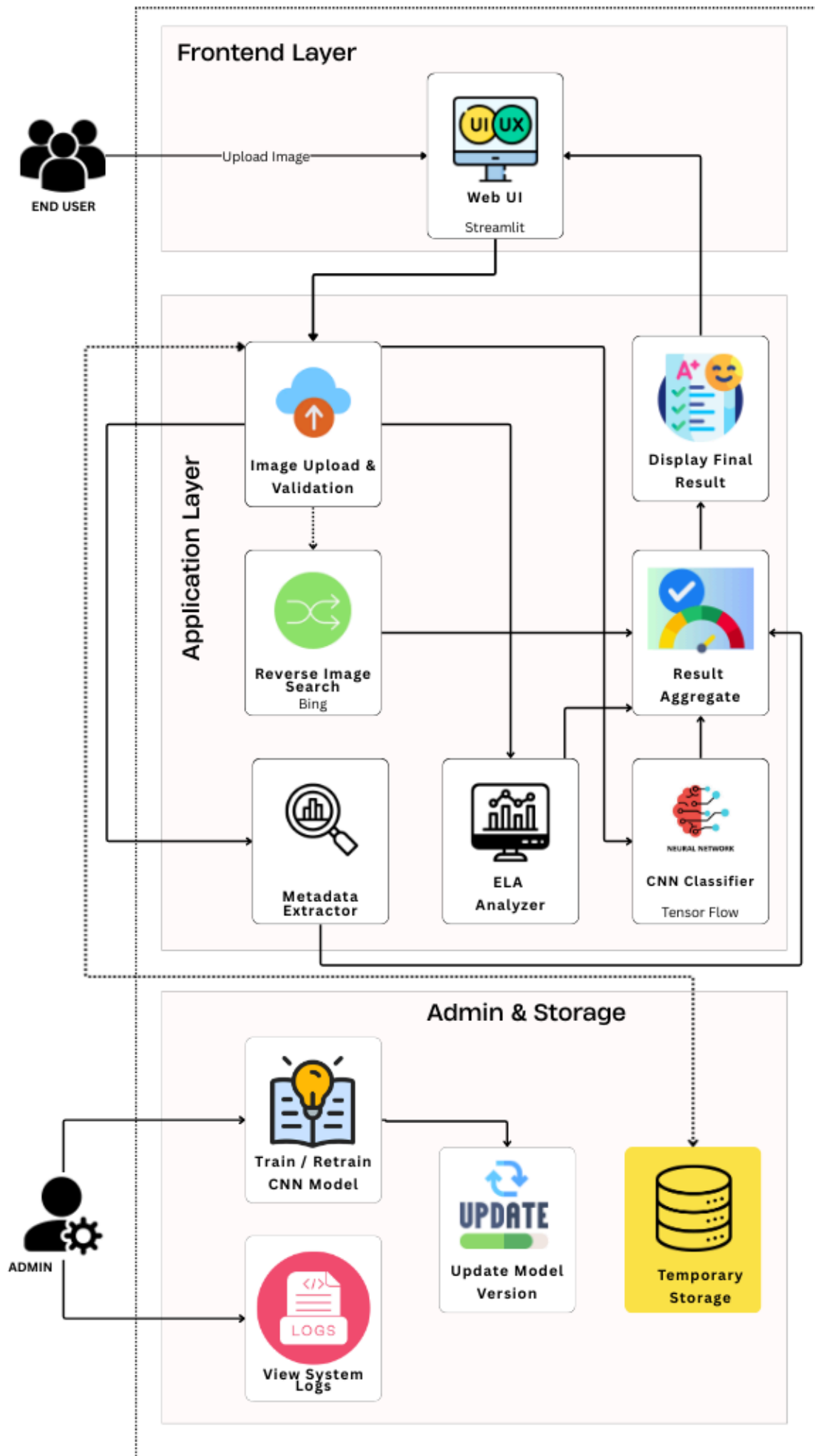
Sequence Diagram

Diagram # 2- Admin: Update or Retrain CNN Model



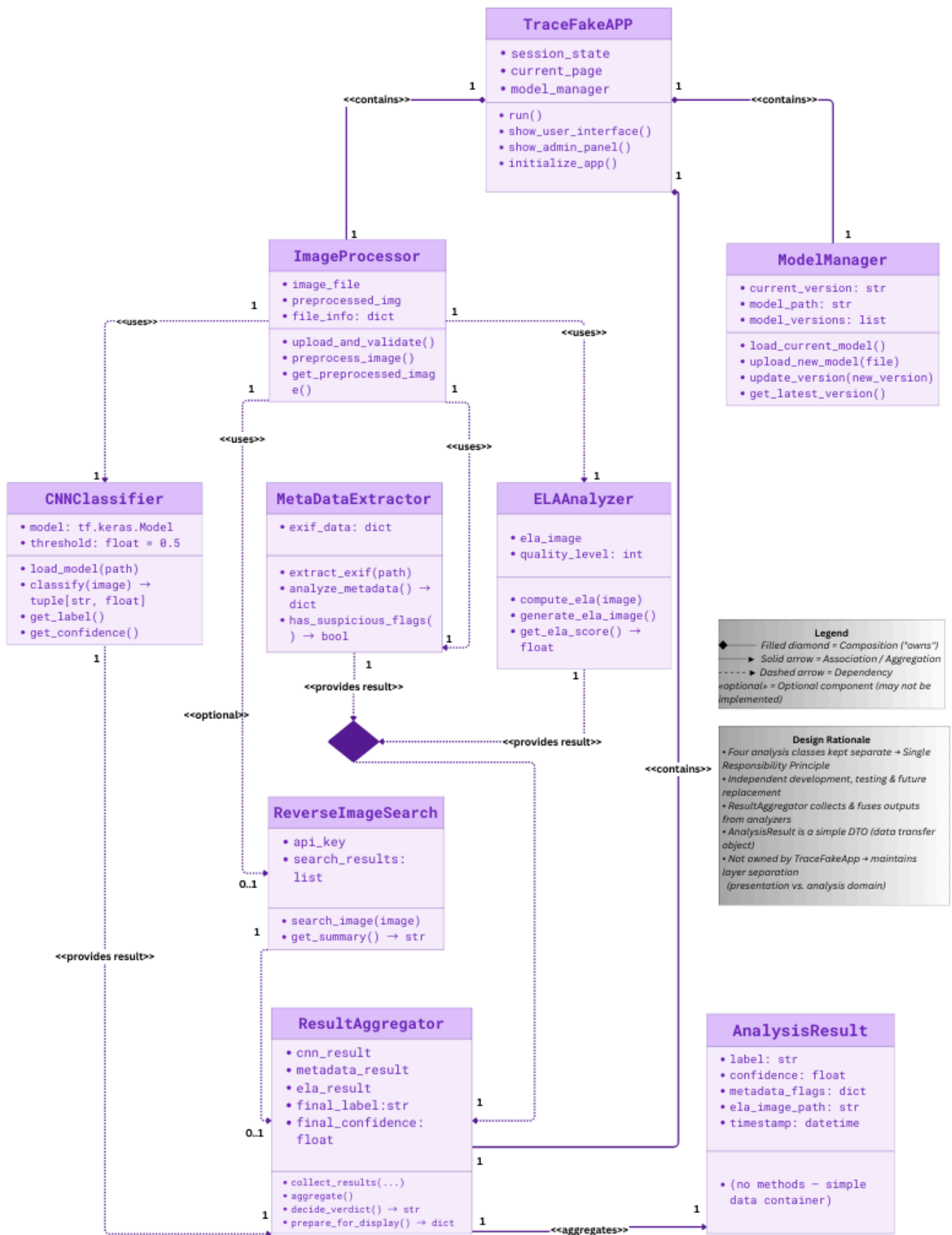
# ARCHITECTURE DESIGN DIAGRAM

## Architecture Diagram



# CLASS DIAGRAM

## Class Diagram



# **DATASET DESIGN**

## Introduction:

The TraceFake system relies upon CNN-based models. A CNN-based model is used to classify images as “Real” or “AI Generated”. So the model requires training. This training is done on balanced datasets. The term balanced means that these will consist of both real and fake images to learn discriminative features. Discriminative features include pixel patterns, noise distribution, and generation artifacts. This section outlines the dataset selection process, including the ones already chosen for the project along with a few alternatives and rationale as well. Datasets are sourced from publicly available repositories. Publicly available repositories are chosen to ensure reproducibility and compliance with ethical guidelines.

Training is performed offline using Google Colab with the pretrained model weights (.h5 or SavedModel format) uploaded to the system via admin panel. No real-time training occurs in the deployed system.

## Visual Overview of Datasets:

Dataset Name	Type	Size (Approximate)	Source	Description	Usage in Project
CelebA (Real) +	Real +	CelebA: 200K images;	CelebA: University of Maryland;	CelebA provides high-quality real celebrity face images.	<b>This is our chosen primary dataset for the system for following reasons.</b>
StyleGAN-g generated (Fake)	Fake	StyleGAN: 100K generated	StyleGAN: GitHub pretrained models	StyleGAN-generated images simulate fake faces using GAN architecture.	Balanced split (50% real, 50% fake) for training CNN model.  Used for baseline classification.
FFHQ (Real)	Real	70K images	NVIDIA	High-resolution face images from Flickr, focusing on diversity. Diversity may include age etc.	Alternative for real images: Can replace or augment CelebA for better diversity in training.
ProGAN-generated (Fake)	Fake	The size is not fixed. It's variable.	GitHub pretrained models	Images generated are mimicking real faces.	Alternative for fake images..

The visual overview answers to all the possible questions that may come in ones mind but still some of the most common ones are answered below for the better understanding.

### **1. Which Dataset Do We Use or Decide to Use?**

The primary dataset chosen for training the CNN model is a combination of **CelebA** (for real images) and **StyleGAN**-generated images (for fake images). This is a hybrid approach. This hybrid approach creates a balanced training set. Balanced training set means almost equal number of real and fake images (e.g 50,000 real + 50,000 fake images for initial experiments). The model is fine-tuned on this data to detect AI-generated artifacts.

### **2. List Down a Few Datasets with Their Sources**

A few of the datasets are already mentioned in the visual overview. It consists of the one chosen for the project and a few alternative options, just incase we need them. A few datasets along with their sources are mentioned below.

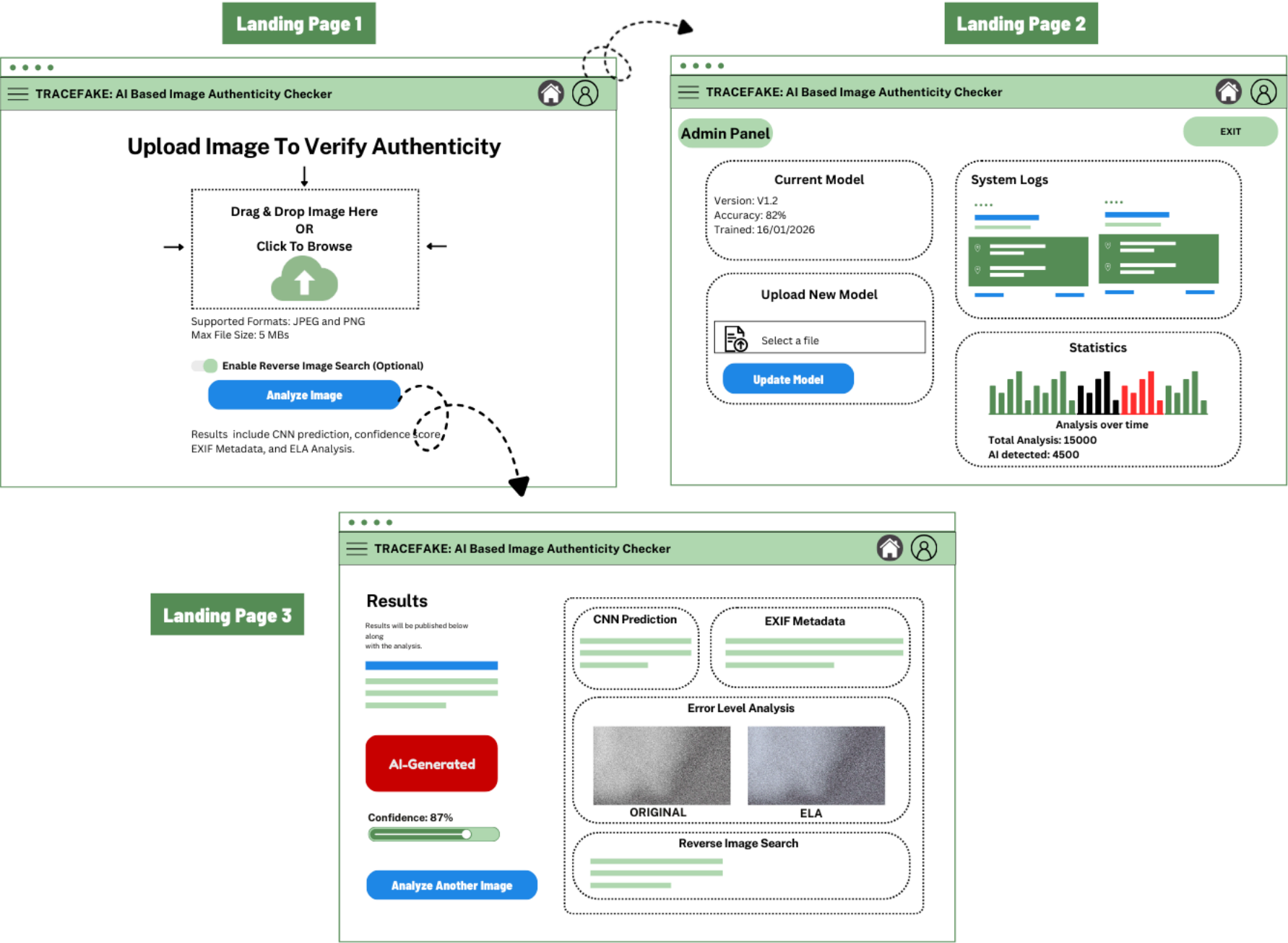
1. CelebA: Source - University of Maryland  
([mmlab.ie.cuhk.edu.hk](http://mmlab.ie.cuhk.edu.hk))
2. FFHQ: Source - NVIDIA  
([github.com/NVLabs/ffhq-dataset](https://github.com/NVLabs/ffhq-dataset))
3. StyleGAN-generated: Source - NVIDIA's pretrained StyleGAN models on GitHub  
([github.com/NVLabs/stylegan2](https://github.com/NVLabs/stylegan2))
4. ProGAN-generated: Source - Progressive GAN pretrained models on GitHub  
([github.com/tkarras/progressive\\_growing\\_of\\_gans](https://github.com/tkarras/progressive_growing_of_gans))

### **3. What's the Reason for Choosing Those Datasets?**



The datasets were selected based on the following criteria, ensuring alignment with project goals, feasibility, and ethical considerations:

- Relevance to Domain
- Balance and Size
- Quality and Diversity
- Availability and Ethics
- Feasibility for Final-Year Project

# INTERFACE DIAGRAMS



TRACEFAKE: AI Based Image Authenticity Checker




## Upload Image To Verify Authenticity

↓

→

Drag & Drop Image Here  
OR  
Click To Browse



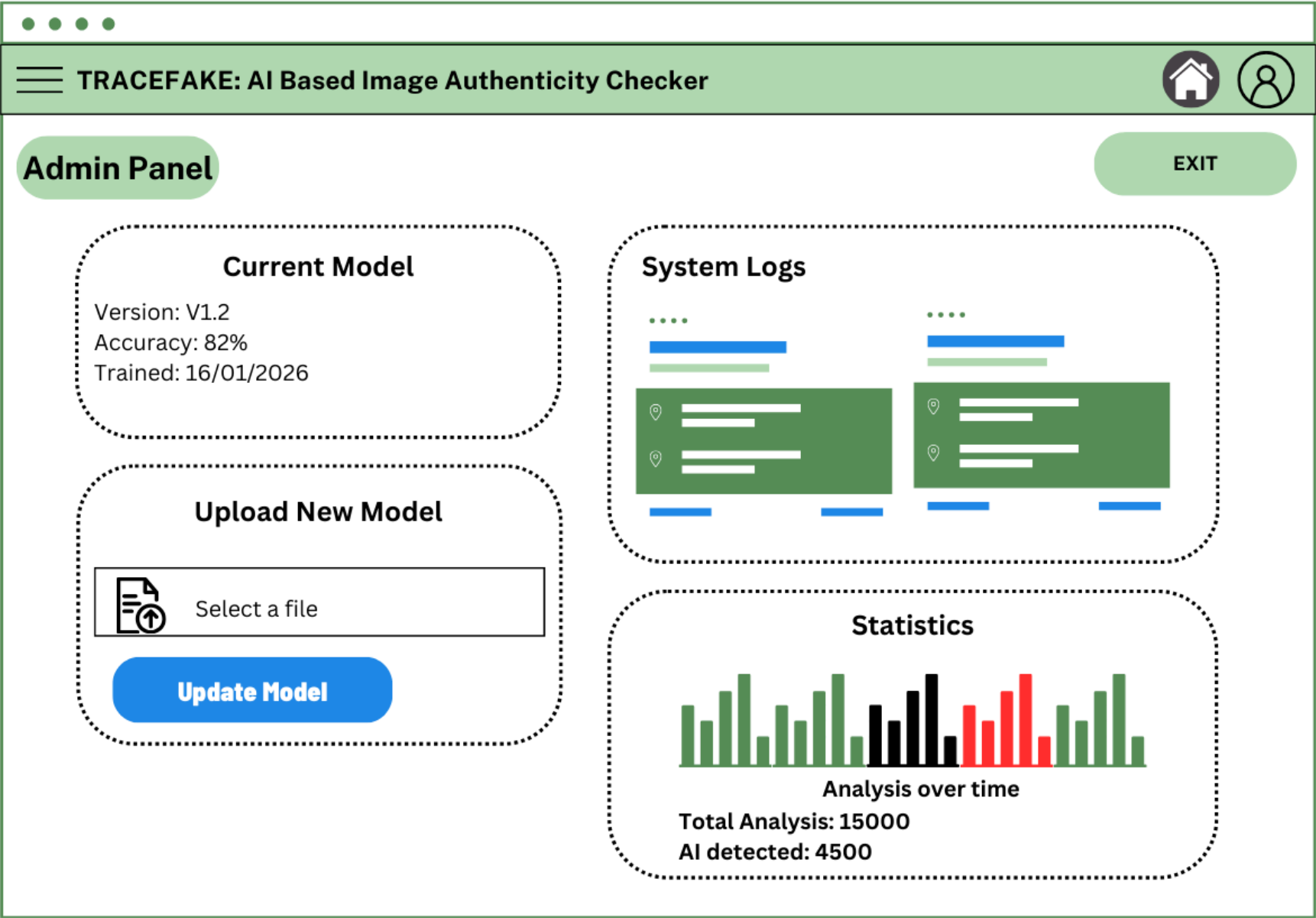
←

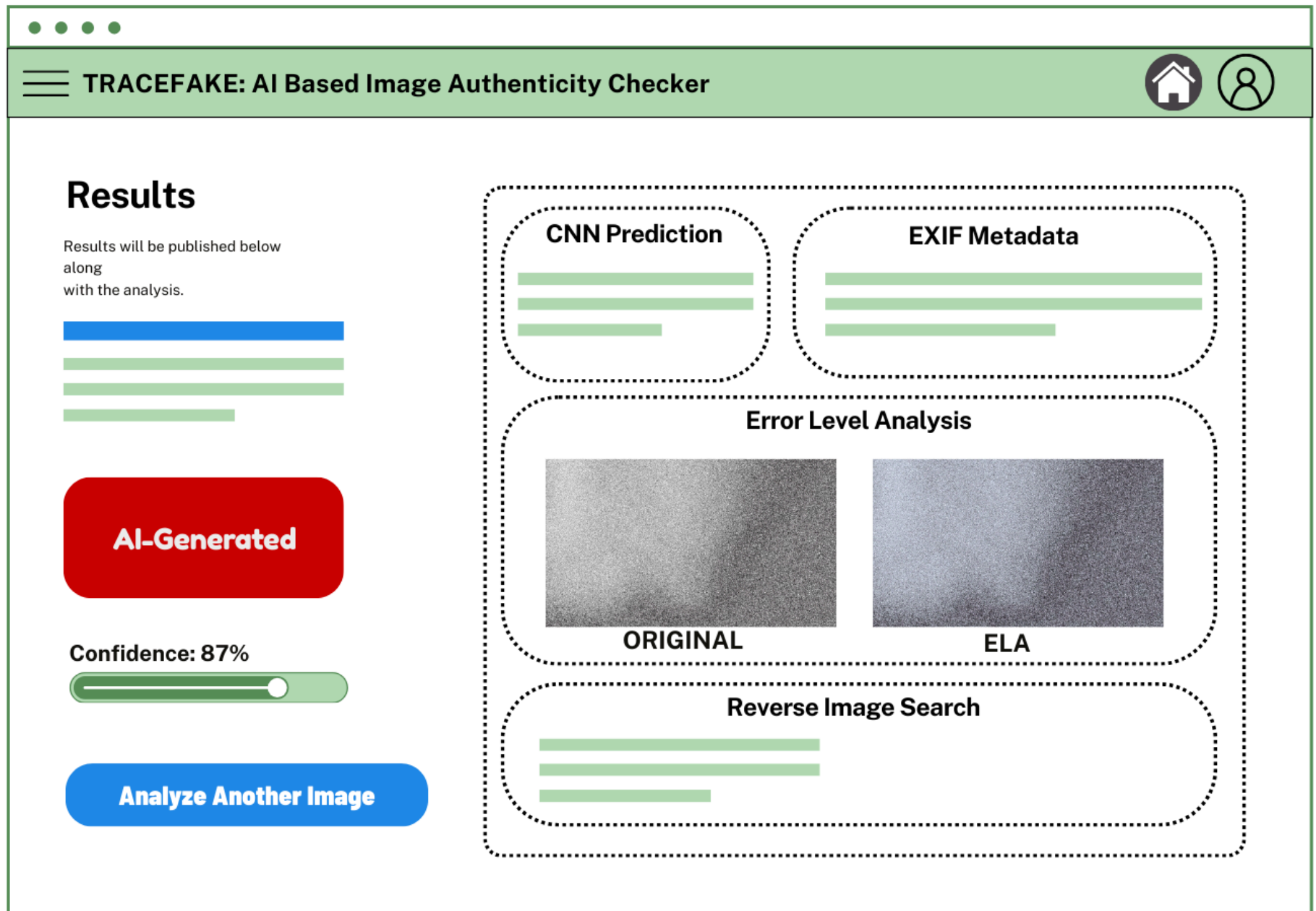
Supported Formats: JPEG and PNG  
Max File Size: 5 MBs

☒ Enable Reverse Image Search (Optional)

Analyze Image

Results include CNN prediction, confidence score  
EXIF Metadata, and ELA Analysis.





# TEST CASES

## Purpose:

*This section defines test cases during the design phase to verify that the system satisfies all functional requirements (FR1–FR6) and non-functional requirements (NFR1–NFR10). The cases cover normal, edge, negative, and performance scenarios. They will be executed during the implementation and testing phase. This will be carried out using a combination of automated (pytest, unit/integration tests) and manual testing (UI, usability, visual inspection).*

## Test Case Table:

Test ID	Req	Category	Description	Preconditions	Input / Steps	Expected Output	Status
TC-01	FR1	Functional	Successful image upload.  (supported formats)	- System is running. - User on main page	1. Select JPEG/PNG file < 5 MB  2. Click "Upload" / "Analyze"	Image successfully uploaded, preview shown, processing starts.	TBD
TC-02	FR1	Edge/Negative	Upload of invalid file type or size	System is running	1. Select .txt file or image > 10 MB  2. Click "Upload"	Clear error message: "Unsupported file type" or "File size exceeds limit". No processing starts	TBD

TC-03	FR2	Functional	Correct classification of real image	Valid model loaded	1. Upload known real image 2. Click "Analyze"	Label = "Real", confidence $\geq 0.80$ (or project-defined threshold)	TBD
TC-04	FR2	Functional	Correct classification of AI-generated image	Valid model loaded	1. Upload known fake image (e.g. StyleGAN sample) 2. Click "Analyze"	Label = "AI-generated", confidence $\geq 0.80$	TBD
TC-05	FR3	Functional	EXIF metadata extraction and display	Image with EXIF data uploaded	1. Upload JPEG with camera/software tags 2. Complete analysis	Metadata table displayed (make, model, software, date, etc.); suspicious flags highlighted if edited	TBD
TC-06	FR3	Edge	Image without EXIF data	Image stripped of EXIF uploaded	1. Upload image with no EXIF 2. Complete analysis	Message: "No EXIF metadata found" or "Possible stripped metadata – potential manipulation"	TBD

TC-07	FR4	Functional	ELA image generation and display	Valid image uploaded	1. Upload image 2. Complete analysis	ELA visualization image generated and displayed; compression artifacts visible if manipulated	TBD
TC-08	FR4	Edge	ELA on uncompressed / lossless image	Upload PNG (lossless)	1. Upload PNG 2. Complete analysis	ELA shows uniform pattern (expected for lossless); no false tampering indication	TBD
TC-09	FR5	Functional	Complete result display with confidence & evidence	Any valid image analyzed	1. Complete full analysis	Clear result page showing: <ul style="list-style-type: none"> <li>• Final label (Real / AI-generated)</li> <li>• Confidence score (0–100%)</li> <li>• Metadata summary</li> <li>• ELA image</li> <li>• Optional reverse search summary</li> </ul>	TBD
TC-10	FR5 + NFR3	Usability	Result page readability & responsiveness	Mobile/desktop browser	1. Analyze image 2. View result on phone & desktop	Layout responsive, text readable, images not distorted, loading < 2 s after analysis	TBD

TC-11	FR6	Functional	Reverse image search triggered and results shown	API key configured, checkbox selected	1. Upload image 2. Enable reverse search 3. Analyze	Summary of matches (e.g. number of similar images, source links or "No matches found")	TBD
TC-12	FR6	Negative	Reverse image search when API unavailable	API key invalid / network down	1. Disable network or use wrong key  2. Enable reverse search & analyze	Graceful fallback: "Reverse search unavailable" message; core analysis (CNN+ELA+metadata) continues	TBD
TC-13	NFR1	Performance	End-to-end analysis time	Standard laptop / Colab CPU	1. Upload 1–2 MB image  2. Measure time from upload to result	Total processing time $\leq$ 12 seconds (target 5–8 s on modern hardware)	TBD
TC-14	NFR2	Accuracy	Model accuracy on validation set	Trained model, validation split prepared	1. Run model on 500–1000 unseen images (balanced real/fake)  2. Compute metrics	Accuracy $\geq$ 85%, Precision $\geq$ 0.82, Recall $\geq$ 0.80, F1-score $\geq$ 0.81 (project target)	TBD
TC-15	NFR4 (Size)	Resource	Application & model size check	Deployed version	1. Check file sizes	Total app size (including model) $\leq$ 500 MB Model file $\leq$ 300 MB	TBD

TC-16	NFR5 Offline	Functional / Non-functional	System works without internet (except optional reverse search)	No internet connection	1. Disconnect internet 2. Upload & analyze image	Core features (upload, CNN, ELA, metadata, result) work normally Reverse search shows "unavailable"	TBD
TC-17	NFR6	Reliability	Repeated analysis with no crash	System running	1. Analyze 20 different images in sequence	No crashes, no memory leaks, consistent results	TBD
TC-18	NFR7	Portability	Run on different OS / environments	Windows / macOS / Linux	1. Install & run on different OS/browser	Application starts and functions correctly on tested platforms	TBD
TC-19	NFR8	Maintainability	Code readability & structure check	Source code available	1. Review code structure, comments, modularization	Code is modular, well-commented, functions/classes < 100 lines where possible, meaningful names	TBD
TC-20	NFR9	Security	No sensitive data exposure & basic input validation	System running	1. Attempt path traversal in upload 2. Inspect network responses	No file system access outside temp folder No API keys or model weights exposed in frontend	TBD