
Software Requirements Specification

for

TraceFake: AI-Based Image Authenticity Checker

Prepared by: Hamza Shafiq

<version 1.0>

Table of Contents

Table of Contents	ii
Revision History	ii
1. Introduction	1
1.1 Overview	1
1.2 Purpose	1
1.3 Intended Audience and Reading Suggestions	1
1.4 Project Scope	2
1.5 Objectives	2
1.6 Document Conventions	3
1.7 References	3
2. System Overview	4
2.1 Product Perspective	4
2.2 Product Functions	4
2.3 User Classes and Characteristics	4
2.4 Operating Environment	5
2.5 Design and Implementation Constraints	5
2.6 Assumptions and Dependencies	5
3. Functional Requirements	6
3.1 FR1: Image Upload	6
3.2 FR2: Image Classification	6
3.3 FR3: EXIF Metadata Extraction & Display	7
3.4 FR4: Error Level Analysis (ELA)	7
3.5 FR5: Display Result with Confidence Score & Forensic Evidence	8
3.6 FR6: Reverse Image Search Integration	8
4. Non Functional Requirements	9
4.1 NFR1: Performance	9
4.2 NFR2: Accuracy	9
4.3 NFR3: User Interface Responsiveness	9
4.4 NFR4: Model & Application Size	9
4.5 NFR5: Offline Capability	9
4.6 NFR6: Reliability	10
4.7 NFR7: Portability	10
4.8 NFR8: Maintainability & Readability	10
4.9 NFR9: Security	10
4.10 NFR10: Scalability	10
5. System Models	11
5.1 Use Case Diagram	11
5.1.1 Actors	
5.1.2 Use Cases	
5.2 Use Case Diagram Description	12
6. Usage Scenario	13
6.1 Scenario 1 - Typical End User Flow	13
6.2 Scenario 2 - Administrator Retrains the Model	13
6.3 Scenario 3 - Real Life - Journalist Fact Checking	13

Appendix A: Glossary	5
Appendix B: Analysis Models	6
Appendix C: Issues List	6

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Overview

TraceFake is a web app that tells if a digital image is real or made by a generative model like Midjourney, Stable Diffusion, or DALL·E. It doesn't just rely on deep learning; it also uses classic digital forensics, so you get both a verdict and some actual evidence you can see. You can upload standard image files like JPEG or PNG, nothing fancy. The app checks them automatically and then shows the results in a simple, easy-to-use interface built with Streamlit.

1.2 Purpose of the Document

This document lays out exactly what the TraceFake project needs to do. What the system should offer, what it shouldn't, and what it connects to outside itself. Think of it as the go-to guide for the whole CS619 Final Year Project (Fall 2025).

1.3 Intended Audience and Reading Suggestions

The intended audience for this document includes:

	Audience	Purpose of Using TraceFake
	General public concerned about visual misinformation.	Quickly check if viral images are real or AI generated.
	Students and Academic Researchers.	Study deepfake detection techniques and datasets.
	Digital forensic experts.	Use a supportive tool during investigations.
	Journalist & media verification teams.	Verify images before publishing tools.
	Fact checkers & misinformation investigators.	Rapid authenticity checking for social media claims.
	Social media users.	Personal verification of photos shared online.

Reading Suggestion:

- 1- Evaluators and supervisors are recommended to read the document in its entirety.
- 2- End-users and future developers may focus on Sections 1 (Project Overview, Scope, Objectives), 3 (Functional Requirements), and 6 (System Features) for a quick understanding of capabilities and usage.

1.4 Project Scope

Here's what the system does:

- 1- Users can upload one image at a time to check if it's authentic. The system tells you if the image is "Real" or "AI-Generated," using a Convolutional Neural Network (CNN) or a tweaked pre-trained model. If there's EXIF metadata, the system pulls it out and shows it.
- 2- The tool runs Error Level Analysis (ELA) on the photo and shows you the results, pointing out any weird compression patterns.
- 3- You get one clear report at the end: the prediction (real or AI), confidence score, a quick EXIF summary, and the ELA image.
- 4- A Bonus feature could also be added. You might also be able to run a reverse image search to check where else the picture shows up online.

Here's what we have not included in the system:

- 1- No real-time video or audio deepfake detection
- 2- No telling you which specific model generated the image
- 3- No batch processing of multiple images at once

1.5 Objectives

- 1- Create a tool that can identify AI-generated photos and explain its findings.
- 2- Combining deep learning with traditional forensic procedures improves the reliability of the results.
- 3- Use Streamlit to design a user-friendly web interface that looks nice on a variety of devices.
- 4- Achieve at least 90% accuracy on typical datasets comparing actual and synthetic photos.
- 5- Deliver a finished, ready-to-run app, complete with documentation and a demo video, by the end of Spring 2026.

1.6 Document Conventions (Definitions, Acronyms and Abbreviations)

Term	Definition
AI-Generated	<i>Images created by generative adversarial networks (GANs) or diffusion models</i>
CNN	<i>Convolutional Neural Network</i>
EXIF	<i>Exchangeable Image File Format – metadata embedded in photographs</i>
ELA	<i>Error Level Analysis – forensic technique to detect JPEG compression differences</i>
Streamlit	<i>Open-source Python library for creating web applications with minimal code</i>
Confidence Score	<i>Probability percentage output by the classifier indicating certainty</i>
Pretrained Model	<i>Deep learning model already trained on large datasets (e.g., ImageNet)</i>

1.7 References

- VU CS619 Project Guidelines – Fall 2025
- “CNN-based Detection of Generic Deepfake Images” – various research papers (2022–2025)
- Streamlit Official Documentation – <https://docs.streamlit.io>
- OpenCV Documentation – <https://docs.opencv.org>
- ExifRead Python Library – <https://pypi.org/project/ExifRead/>

2. System Overview

2.1 Product Perspective

TraceFake is a free, open-source web app built for checking if an image is real or fake. It is a much needed tool because it's getting harder and harder to spot AI-generated images these days. Anyone can make a convincing fake in seconds. Most commercial tools either cost money or keep their code hidden. With TraceFake, we wanted something different: a lightweight, transparent tool that anyone can use, whether you're a researcher, journalist, fact-checker, or just someone curious about what's real online.

2.2 Product Functions

The system shall provide following major functions.

- 1- You can upload a single image and instantly see what it looks like.
- 2- The system runs a deep learning model to figure out if the image is real or AI-generated, and it shows you a confidence score.
- 3- It also pulls out and explains any EXIF metadata hidden inside the file.
- 4- Error Level Analysis (ELA) is available too. You get to see that side by side with your original image.
- 5- All the results come together in one place, mixing the AI's verdict with the forensic details.
- 6- If you want, there's even a reverse image search to help check where the picture came from.

2.3 User Classes and Characteristics

User Class	Characteristics	Frequency of Use
General Public	Limited technical knowledge, uses via web browser.	Occasional
Journalists & Fact-checkers	Moderate technical skills, needs quick and interpretable results.	Frequent
Students & Researchers	High technical knowledge, may study source code and model behaviour as well.	Frequent
Digital Forensic Experts	Professional users will use TraceFake as a preliminary triage tool for verification purposes.	Regular

2.4 Operating Environment

Component	Specification
<i>Hardware</i>	<i>A laptop or a simple desktop PC above 6th gen.</i>
<i>CPU</i>	<i>Any modern multi-core processor.</i>
<i>Storage</i>	<i>Less than 1 GB.</i>
<i>Operating System</i>	<i>Windows 10/11, macOS 11+, or any Linux distribution (Ubuntu 20.04+)</i>
<i>Python Version</i>	<i>Python 3.12</i>
<i>Primary Libraries</i>	<i>TensorFlow/Keras or PyTorch, OpenCV-Python, Streamlit, exifread, Pillow</i>
<i>Web Browser (Client Side)</i>	<i>Google Chrome, Mozilla Firefox, Microsoft Edge, or Safari</i>
<i>Network</i>	<i>Capable of running fully offline. Internet connection required only for optional reverse image search.</i>
<i>Execution Mode</i>	<i>Local: <code>streamlit run app.py</code> OR Free cloud deployment using Streamlit Community Cloud or Render.</i>

2.5 Design and Implementation Constraints

- 1- *Implementation Language:* Python only
- 2- *Deep learning framework:* TensorFlow or PyTorch
- 3- *User interface:* Streamlit

2.6 Assumptions and Dependencies

Users need an active internet connection for reverse image search to work.

The pretrained or fine-tuned model hits at least 90% accuracy on the held-out test set.

Most real photos still have some EXIF data. If the metadata's missing entirely, that's a red flag (AI generated).

This system isn't meant to be used as legal forensic proof.

3. Functional Requirements

3.1 FR1 – Image Upload

3.1.1 Description

The system shall allow the user to upload a single image at a time for authenticity analysis.

3.1.2 Priority

It is MANDATORY to upload the image for analysis.

3.1.3 Pre-conditions

The application is running and the user interface is displayed to the user.

3.1.4 Input:

The uploaded image is in JPEG or PNG format. The size of the uploaded image should not exceed 10 MB.

3.1.5 Process

The user selects the image. Another option is that user use the drag and drop feature using Streamlit file uploader.

3.1.6 Post-conditions

The uploaded image is temporarily stored in memory and also displayed on screen for confirmation.

3.2 FR2 – Image classification

3.2.1 Description

The system shall classify the image as REAL or AI GENERATED using a trained CNN model.

3.2.2 Priority

Mandatory.

3.2.3 Pre-conditions

The image has been successfully uploaded and preprocessed.

3.2.4 Input:

Preprocessed image tensor (224 × 224 or 256 × 256 pixels).

3.2.5 Process

The image is uploaded to the deep learning model which outputs two probabilities (AI generated or Real).

3.3 FR3 – EXIF Metadata Extraction & Display

3.3.1 Description

The system shall extract and display the EXIF metadata from the uploaded image.

3.3.2 Priority

Mandatory.

3.3.3 Pre-conditions

Image has been uploaded.

3.3.4 Input:

Original image file.

3.3.5 Process

The Pillow library to read EXIF tags (Make, Model, DateTime, Software, GPS, etc.).

3.3.6 Output

Key-value list of available metadata.

If no EXIF data exists, display “No EXIF meta data found.”.

3.3.6 Post-conditions

Metadata information is ready to be shown alongside the final result.

3.4 FR4 – Error Level Analysis (ELA) Generation

3.4.1 Description

The system shall perform Error Level Analysis and generate a visual ELA image to highlight potential tampering or uniform compression.

3.4.2 Priority

Highly Recommended.

3.4.3 Pre-conditions

The Image is in JPEG format or can be converted into JPEG.

3.4.5 Process

- 1- Re-save image at 90-95% quality.
- 2- Compute absolute difference with original.
- 3- Amplify and display the difference map.

3.4.6 Output

Side-by-side view: Original image + ELA image.

3.5 FR5 – Display Result with Confidence Score & Forensic Evidence

3.5.1 Description

The system shall present a clear, consolidated result page containing all the outcomes of the analysis.

3.5.2 Priority

Mandatory.

3.5.3 Pre-conditions

FR2, FR3 and FR4 have completed successfully.

3.5.4 Output:

- 1- Large verdict text: “REAL” (green) or “AI-GENERATED” (red).
- 2- Confidence percentage and progress bar.
- 3- EXIF metadata table or list.
- 4- Original vs ELA image comparison.
- 5- Summary explanation in plain language.

3.5.5 Post-conditions

Users can view, screenshot or download the complete report.

3.6 FR6 – Reverse Image Search Integration.

3.6.1 Description

The system may perform a reverse image search using Bing API.

3.6.2 Priority

Optional.

3.6.3 Pre-conditions

Internet connection available and API key configured.

3.6.4 Process

Upload image to reverse search API and retrieve top results.

3.6.5 Output

Thumbnails and links to matching web pages (if any).

3.6.6 Post-conditions

Additional evidence displayed below the main result (only if enabled).

4. Non-Functional Requirements

4.1 NFR1 – Performance.

4.1.1 Description

Complete analysis must finish within 8 seconds on a standard laptop.

4.1.2 Priority

Mandatory.

4.2 NFR2 – Accuracy.

4.2.1 Description

AI classification accuracy shall be greater than 90%.

4.2.2 Priority

Mandatory.

4.3 NFR3 – User Interface Responsiveness.

4.3.1 Description

Streamlit interface shall load and respond within 2 seconds on normal internet or locally.

4.3.2 Priority

Mandatory.

4.4 NFR4 – Model & Application Size.

4.4.1 Description

Total package shall not exceed 1 GB for easy distribution.

4.4.2 Priority

Mandatory.

4.5 NFR5 – Offline Capability.

4.5.1 Description

Core detection (CNN + EXIF + ELA) must work completely offline after first run.

4.5.2 Priority

Mandatory.

4.6 NFR6 – Reliability.

4.6.1 Description

The web app shall not crash on valid JPEG/PNG inputs. It must also handle corrupt files gracefully with clear error messages.

4.6.2 Priority

Mandatory.

4.7 NFR7 – Portability.

4.7.1 Description

The system shall run on Windows, macOS and Linux with a single command (`streamlit run app.py`).

4.7.2 Priority

Mandatory.

4.8 NFR8 – Maintainability & Readability

4.8.1 Description

Code must be properly commented so that it's easy to maintain and understand.

4.8.2 Priority

Highly Recommended.

4.9 NFR9 – Security

4.9.1 Description

No user data or images are stored permanently on disk or server after the session ends.

4.9.2 Priority

Mandatory.

4.10 NFR10 – Scalability.

4.10.1 Description

Architecture should allow easy upgrade to newer models or additional forensic modules without rewriting the UI.

4.10.2 Priority

Recommended.

5. System Models

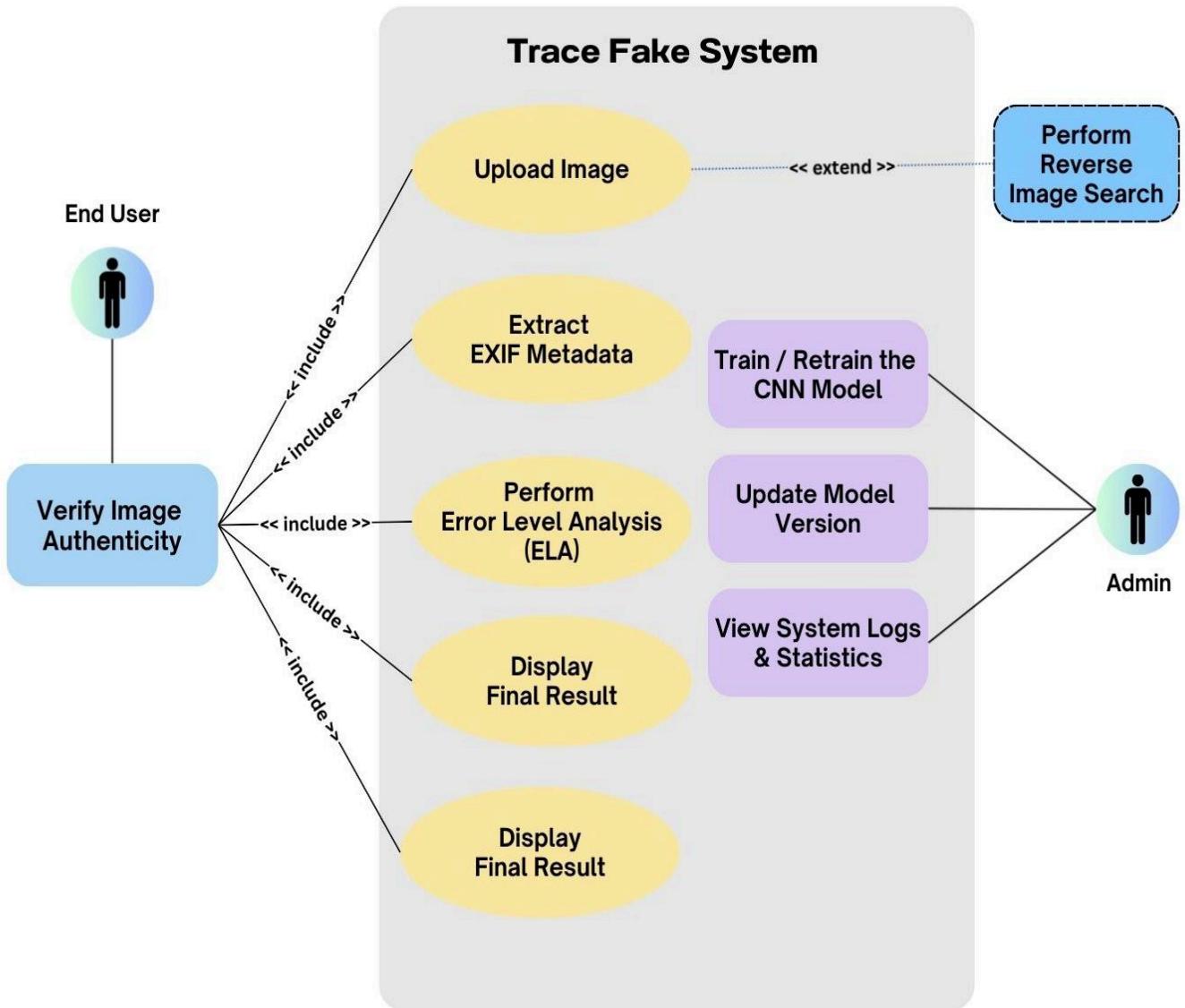
5.1 Use case diagram

5.1.1 Actors

*End User: Any user who wants to check if the image is real or AI generated
 Admin: An authorized person who is responsible for maintaining & updating.*

5.1.2 Use Cases

*End User use cases:
 Upload Image, Verify Authenticity, Perform Reverse Image Search
 Admin:
 Train/Retrain System, Update Model, View System Logs & Stats.*



5.2 Use Case Diagram Description

Use Case Name	Actor	Description
Verify Image Authenticity	End User	<i>The user requests the system to check whether the uploaded image is real or AI-generated.</i>
Upload Image	End User	<i>The user selects and uploads a single image file.</i>
Perform Reverse Image Search (Optional)	End User	<i>The user have the option if they want to perform the reverse image search to find the source of the image.</i>
Classify Image using CNN	System	<i>The system runs the pre-trained CNN model to classify the image as Real or AI-Generated.</i>
Extract EXIF Metadata	System	<i>The system extracts and analyses EXIF data. The info may include camera, date, software, GPS, etc.</i>
Perform Error Level Analysis (ELA)	System	<i>The system performs ELA to detect compression inconsistencies. It will highlight any manipulation.</i>
Display Final Result with Evidence	System	<i>The system shows the final verdict, confidence score, EXIF highlights, and ELA report.</i>
Train / Retrain CNN Model	Admin	<i>The admin uploads new dataset and retrains the CNN model to improve accuracy when needed.</i>
Update Model Version	Admin	<i>The admin switches the live system to the newly trained model version.</i>
View System Logs & Statistics	Admin	<i>The admin views usage statistics, error logs, and system performance metrics.</i>

6. Usage Scenario

6.1 Scenario 1 – Typical End-User Flow

On the TraceFake website, an anonymous user chooses "Choose File," picks a JPEG or PNG file from their device, and then clicks "Verify Authenticity." The system then shows "Real" or "AI-Generated" in five to eight seconds, along with the ELA report, EXIF highlights, and confidence score. After that, the user can select "Reverse Image Search" to view the results in a new tab only if they want.

6.2 Scenario 2 – Administrator Retrains the Model

User Story: A few weeks later, the administrator discovers that the accuracy of the system has somewhat decreased on recently released AI generators (such as more recent iterations of Midjourney or Stable Diffusion). Since the initial model was trained using older data, this is to be expected.

Procedure:

- The admin panel is accessed by the administrator.
- Admin uploads a new, balanced dataset that includes the most recent AI-generated images along with thousands of genuine shots.
- To track training progress, click "Train / Retrain CNN Model."
- The system exhibits enhanced validation accuracy when training is complete.
- To activate the freshly trained model, the administrator selects "Update Model Version."
- All end users immediately benefit from the enhanced detection after that; no more action is required..

6.3 Scenario 3 (Real Life) - Journalist Fact Checking

A reporter notices an odd photo of OpenAI co-founder Ilya Sutskever circulating on social media. It simply doesn't look right. Ilya is a well-known figure in deep learning, so before publishing the story, the journalist wants to confirm that the image is authentic or identify it as phony.

- 1- The reporter then launches TraceFake in the web browser.
- 2- No hassle, no login. simply drags the questionable image into the upload window.
- 3- TraceFake starts working.
- 4- It starts by using its CNN-based classifier, which has been trained to distinguish between authentic and artificial intelligence-generated images.
- 5- Next, it examines the EXIF metadata, which includes information about the camera model, date, GPS, and even the software that was used to take the picture.
- 6- Additionally, TraceFake performs Error Level Analysis, which searches for strange compression artifacts that typically appear in artificial intelligence (AI)-generated or altered images.

7- Results appear a few seconds later:

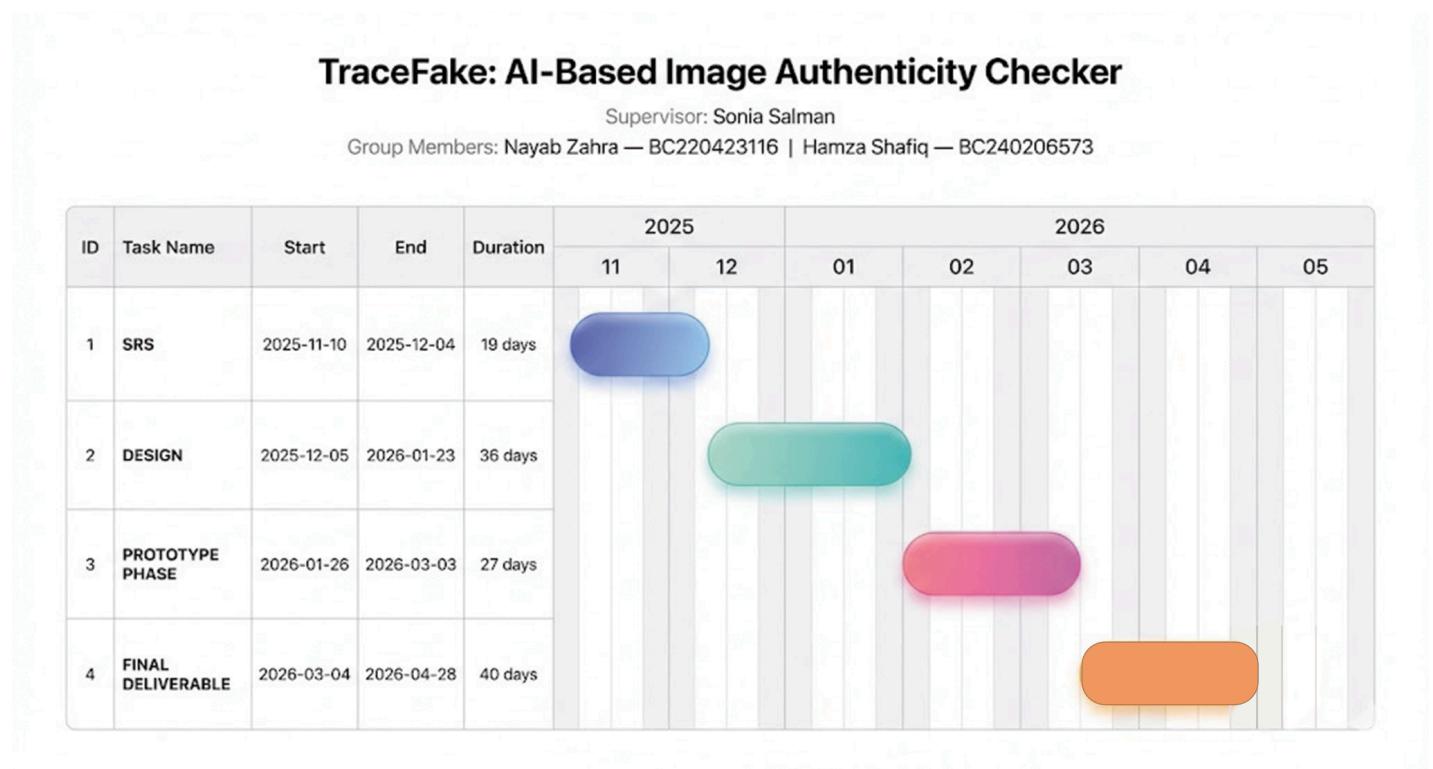
- Prediction: 92% likely fake*
- Metadata: A classic indication of AI generation: no camera information found*
- Forensics Unnatural compression block illuminates the ELA heat map.*

8- The journalist uses "Reverse Image Search" out of curiosity. TraceFake opens TinEye and Google in new tabs instantly. As it happens, Midjourney was used to create the original image only two days ago.

The journalist publishes a fact-check article outlining the proof that the image is AI-generated after obtaining the source and TraceFake's evidence.

7.0 Work Plan

7.1 Gantt Chart



7.2 Summary (Gantt Chart)

Week	Activity	Deliverable
1–2	SRS preparation & submission.	Approved SRS
3–4	System design	SDD draft
5–7	System diagrams	All diagrams
8–9	Backend development + CNN model training	Prototype
10–11	Frontend + reverse search integration	Complete web application
10–11	Testing + bug fixing	Test cases & reports
12	Final documentation & presentation prep	Final Deliverable, SDD, Code, Video

Appendix A: Glossary

CNN (Convolutional Neural Network)

Deep learning model used to classify images as Real or A I-Generated

EXIF Metadata

Embedded information in image files (camera model, date, GPS, software, etc.)

Error Level Analysis (ELA)

Forensic technique that visualises compression differences to detect manipulation.

Reverse Image Search

Searching the web for identical or similar copies of the uploaded image

<include>

UML relationship: one use case automatically includes another

<extend>

UML relationship: one use case optionally extends another

Appendix B: References

- Virtual University of Pakistan. (2024). CS619 – Final Year Project Guidelines.
- IEEE Std 830-1998. Recommended Practice for Software Requirements Specifications.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.