BCSE209L - Machine Learning

Multimodal System for Deepfake Detection

22BDS0002 Ashwin V

22BDS0342 M Thirunarayanan

22BDS0364 Anand Vignesh

Under the Supervision of

AARTHY S.L

Professor Grade 1

School of Computer Science and Engineering (SCOPE)

B.Tech.

in

Computer Science and Engineering

Data Science

School of Computer Science and Engineering

Fall Semester 2025-26

TABLE OF CONTENTS

Sl.No	Contents	Page No.
1.	**Literature Review**	3
2.	**Objectives**	5
3.	**Requirement Analysis**	5
	3.1 Hardware Requirements	5
	3.2 Software Requirements	5
4.	**PROJECT MANAGEMENT**	6
	4.1 Gantt Chart	6
	4.2 Work Breakdown Structure	7
5.	**Software Requirements Specification (SRS)**	8
	5.1 Purpose	8
	5.2 Scope	8
	5.3 Product Functions	8
	5.4 User Classes and Characteristics	8
	5.5 Operating Environment	8
	5.6 Design and Implementation Constraints	9
	5.7 External Interface Requirements	9
	5.8 System Features	9
	5.9 Non-Functional Requirements	9
	5.10 Assumptions and Dependencies	9
6.	**Workflow model**	10
7.	**Module design and Implementation**	11
	7.1 Notebook overview	11
	7.2 Output Screenshots	12
	7.3 Links	15
8.	**References**	15

1. LITERATURE REVIEW

No	Title	Authors	Dataset	Metrics	Methodology	Gaps
1	SIDA: Social Media Image Deepfake Detection, Localization & Explanation with Large Multimodal Model	Zhenglin Huang; Jinwei Hu; Xiangtai Li; Yiwei He; Xingyu Zhao; Bei Peng; Baoyuan Wu; Xiaowei Huang; Guangliang Cheng	Social-media focused benchmark of 300,000 images	Accuracy, F1 Score	Jointly performs image detection, localization (mask prediction) and textual explanation. Built on large vision-language model	Dependence on a single generator (FLUX) might introduce data skew
2	Human Performance in Deepfake Detection: A Systematic Review	Klaire Somoray; Dan J. Miller; Mary Holmes	DFDC, Celeb-DFv2, PEFS, TMC, FaceForensics++	F1 Score, Precision, AUC	Search of IEEE, ProQuest, PubMed, Web of Science, and Scopus; two-phase screening, data extraction and quality assessment	Lacks standardization in measuring human detection performance
3	Preserving Fairness Generalization in Deepfake Detection	Li Lin; Xinan He; Yan Ju; Xin Wang; Feng Ding; Shu Hu	FaceForensics++, DFD	Fairness scores and standard detection performance	Expose demographic and domain-agnostic forgery features; fair learning	Dependency on datasets of forged videos generated by multiple manipulation methods
4	Real-Time Deepfake Detection in the Real-World	Bar Cavia; Eliahu Horwitz; Tal Reiss; Yedid Hoshen	ForenSynths, UFD and WildRF (Introduced in this paper)	Accuracy, mean Average Precision (mAP)	LaDeDa (Locally Aware Deepfake Detection Algorithm): patch-based classifier splitting an image into small patches; Tiny-LaDeDa: distilled model	Poor generalization to real-world data; JPEG compression bias
5	Unveiling the Impact of Image Transformations on Deepfake Detection: An Experimental Analysis	Federico Cocchi; Lorenzo Baraldi; Samuele Poppi; Marcella Cornia; Lorenzo Baraldi; Rita Cucchiara	COCOFake dataset (extension of COCO)	Accuracy	Uses pretrained models CLIP, DINO, DINOv2 for feature extraction; 2 classifiers (Linear probe, k-NN); 12 image transformations	Vulnerability of detectors to simple post-processing techniques; Linear-probe overfitting hypothesis

No	Title	Authors	Dataset	Metrics	Methodology	Gaps
6	A GAN-Based Model of Deepfake Detection in Social Media	Preeti; Manoj Kumar; Hitesh Kumar Sharma	CelebA	Accuracy of discriminator, Inception Score (IS), Fréchet Inception Distance (FID)	Proposes DCGAN for detection; updating the discriminator and generator alternately using mini-batches of real and fake images	Handling small datasets; GAN constraints such as mode collapse
7	DeepFake Detection for Human Face Images and Videos: A Survey	Asad Malik; Minoru Kuribayashi; Sani M. Abdullahi; Ahmad Neyaz Khan	CelebA, UADFV, DF-TIMIT, FaceForensics, FaceForensics++, DFD, DFDC	Classification Accuracy, ROC/AUC	Comprehensive review and summarization of existing DeepFake creation tools and detection techniques	Adversarial attacks; Unknown images
8	Delving into Sequential Patches for Deepfake Detection	Jiazhi Guan; Hang Zhou; Zhibin Hong; Errui Ding; Jingdong Wang; Chengbin Quan; Youjian Zhao	FaceForensics++ (FF++), DFFD, DFD, DFDC, Celeb-DF	Binary Classification Accuracy, ROC and AUC	Local- & Temporal-aware Transformer-based Deepfake Detection (LTTD); models sequences of local spatial patches across successive frames	Real-world deployment unknown; struggles with limited or unlabeled data
9	EfficientNets for DeepFake Detection: Comparison of Pretrained Models	Artem Pokroy; Alexey Egorov	DFDC (DeepFake Detection Challenge) dataset	Accuracy, AUC-ROC	Extract face images from video frames (using MTCNN) after lowering the video frame rate; use a pre-trained EfficientNet model	Large models performed worse than smaller ones; larger networks learn more complex patterns harder to transfer
10	DeepfakeNet, an Efficient Deepfake Detection Method	Dafeng Gong; Yogan Jaya Kumar; Ong Sing Goh; Zi Ye; Wanie Chi	FaceForensics++ (3,200 videos selected), Kaggle DFDC (21,767 videos selected), TIMIT	Accuracy, Area Under the Curve (AUC)	New CNN architecture named DeepfakeNet; uses a “split-transform-merge” strategy	Solving the precise detection of faces with different video qualities

2. OBJECTIVES

This project aims to build a system that can detect deepfakes and explain how they were created, while also showing the positive side of the technology through an easy-to-use website. To reach this goal, we plan to:

- Create a deepfake detection model that can check images and videos and say if they have been manipulated.
- Add a classification step to tell which method was used to make the deepfake, such as face swap, lip sync, audio cloning or motion transfer.
- Build an interactive website designed like a cyber-forensics office where users can upload media, see the analysis and learn from it.
- Give visitors clear information about what deepfakes are, why they matter, and also show examples of how similar techniques can be used for good purposes.
- Connect the machine learning models with the website front-end so that the system feels smooth, responsive and visually consistent.
- Keep proper documentation and evaluation to help future updates or deployment.

Together, these objectives ensure that our project not only works technically but also helps people understand the technology and its impact.

---

3. REQUIREMENT ANALYSIS

3.1 Hardware Requirements

- Processor: Intel i5 or equivalent (i7 recommended for development/testing).
- RAM: Minimum 8 GB (16 GB recommended for training and large files).
- GPU: NVIDIA CUDA-enabled GPU (only for model training or heavy inference).
- Storage: 10 GB free space for models, datasets, and logs.

3.2 Software Requirements

- Operating System: Windows 10/11, Linux, or macOS.
- Browser: Chrome/Edge/Firefox (latest versions).
- Programming Language: Python 3.8+
- Libraries: PyTorch, OpenCV, ONNX Runtime (for model deployment).
- Front-End Stack: HTML5, CSS3, JavaScript.
- Development Tools: VS Code, Jupyter Notebook, Git.

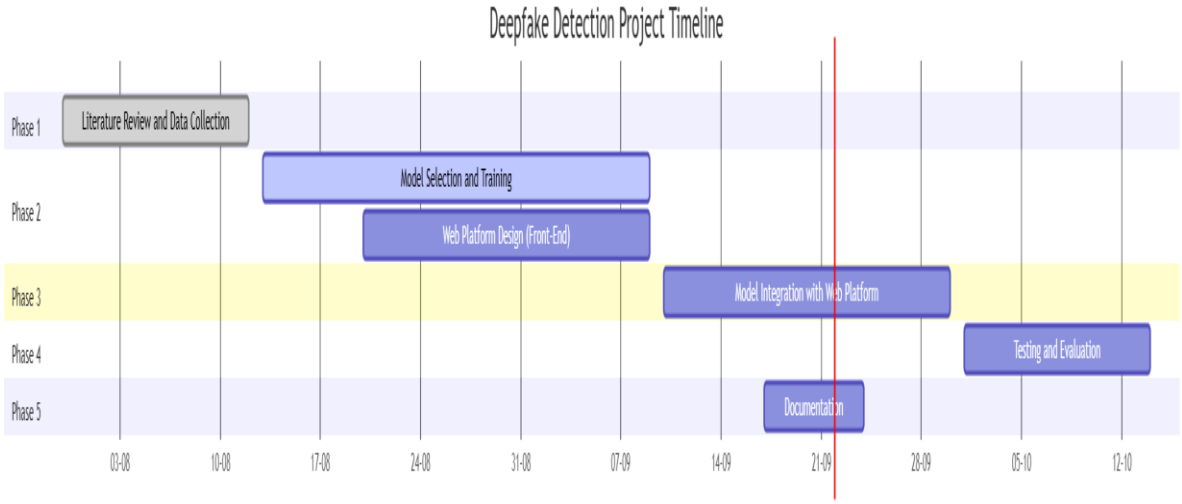
---

4. PROJECT MANAGEMENT

4.1 Gantt Chart

We planned the project in advance so that every stage could be completed on time. Work started on 30 July 2025 and finished on 24 September 2025. Tasks were arranged so that some of them could run in parallel. For example, front-end development began while the AI model was still being trained, which saved time and let us test integration earlier.

Task	Duration (Weeks)	Start Date	End Date
Literature Review and Data Collection	2	30-07-2025	12-08-2025
Model Selection and Training	4	13-08-2025	09-09-2025
Web Platform Design (Front-End)	3	20-08-2025	09-09-2025
Model Integration with Web Platform	3	10-09-2025	30-09-2025
Testing and Evaluation	2	01-10-2025	14-10-2025
Documentation and Report Preparation	1	17-09-2025	24-09-2025



4.2 Work Breakdown Structure

To manage the work, we broke the project into five main phases:

4.2.1. Project Initiation

We defined the objectives, reviewed existing deepfake detection methods and collected user requirements.

4.2.2. System Design

We specified the features, selected the machine learning models and designed the system architecture.

4.2.3. Development

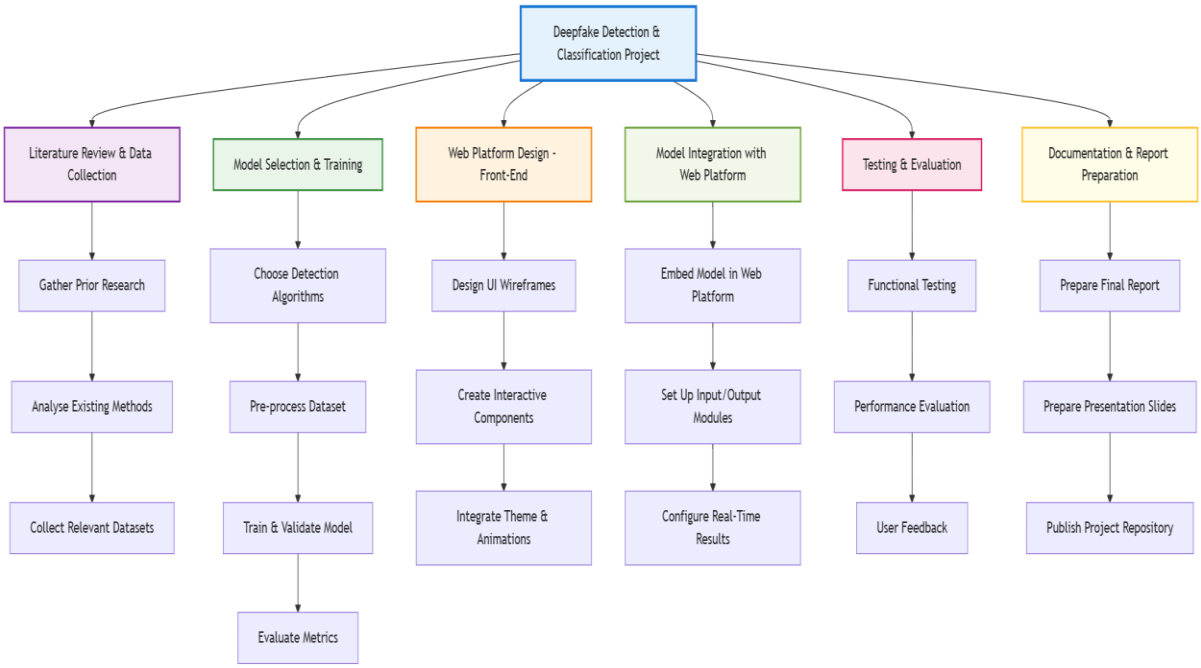
We built the detection model, created the web platform and designed the user interface.

4.2.4. Testing and Evaluation

We verified the functions, checked performance under different conditions and collected feedback for improvements.

4.2.5. Deployment and Documentation

We prepared a small pilot release, wrote user instructions and completed the final report.



5. SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

5.1 Purpose

The purpose of this project is to create a web-based system that can detect deepfakes in images and videos. Alongside detection, the system also aims to educate users about ethical and constructive applications of such technologies. By combining machine learning with a simple, responsive interface, the platform offers both a technical tool and an educational resource.

5.2 Scope

The system allows a user to upload an image or video, automatically processes the content, and returns a clear verdict on whether it is authentic or manipulated. After presenting the results, the platform also displays information about positive uses of deepfake-related technology. The first version focuses on desktop browsers but has been designed so it can be adapted for mobile later.

5.3 Product Functions

The main functions of the system are:

- Upload and preview of images or videos.
- Deepfake detection using a trained model.
- Side-by-side display of the original and analysed content.
- Educational section listing positive applications of similar technologies.
- Display of project team information and acknowledgements.

These functions together provide a complete workflow from input to analysis to user education.

5.4 User Classes and Characteristics

The platform is aimed at several groups:

- **General users** who want to test media and learn about deepfakes.
- **Students and researchers** interested in detection methods and positive applications.
- **Content moderators or journalists** needing quick checks for manipulated media.
- **Developers** seeking examples of integrating detection models into web applications.

5.5 Operating Environment

The system runs as a browser-based application with a front-end built on HTML, CSS and JavaScript and a Python back-end for running the detection model. It works on common desktop browsers such as Chrome, Edge, Firefox and Safari. A mobile-friendly version can be added later.

5.6 Design and Implementation Constraints

Detection should complete quickly — under ten seconds for images and under thirty seconds for short videos. Files are processed locally by default and not stored after analysis to protect privacy. Heavy tasks may be offloaded to cloud resources if required. Models and code are modular so they can be updated without rewriting the entire system.

5.7 External Interface Requirements

The user interacts with a simple upload area, results display and educational section. Input is via standard devices (mouse, keyboard, touchscreen). On the software side, the system uses frameworks such as PyTorch and ONNX Runtime for the model and standard web APIs for file handling and communication.

5.8 System Features

Key features include:

- Simple media upload.
- Deepfake detection.
- Split-screen display of original versus analysed content.
- Educational content about positive applications.
- Team page with project details.

5.9 Non-Functional Requirements

The system must be clear and intuitive for first-time users, perform analysis within the stated time limits, and allow easy addition of new detection techniques and educational content. Uploaded files must be handled securely and not stored permanently. The code should be modular for maintainability and robust enough to handle different file types and sizes.

5.10 Assumptions and Dependencies

The system assumes a stable internet connection if cloud inference is used, availability of a trained detection model, and browser support for the required JavaScript and Web APIs. Training the model requires access to labelled datasets.

---

6. WORKFLOW MODEL

The system follows a simple, step-by-step process from the time a user uploads a file to when the results are shown. This clear flow makes it easy to maintain and helps users understand what is happening at each stage.

Step 1 – Upload and Pre-processing

Users open the website and upload an image or video. The system checks the file type and size, extracts frames from videos if needed, and normalises the data so that it is ready for analysis.

Step 2 – Deepfake Detection

The prepared content is sent to the detection model. This model analyses the media and decides whether it is authentic or manipulated.

Step 3 – Results Display

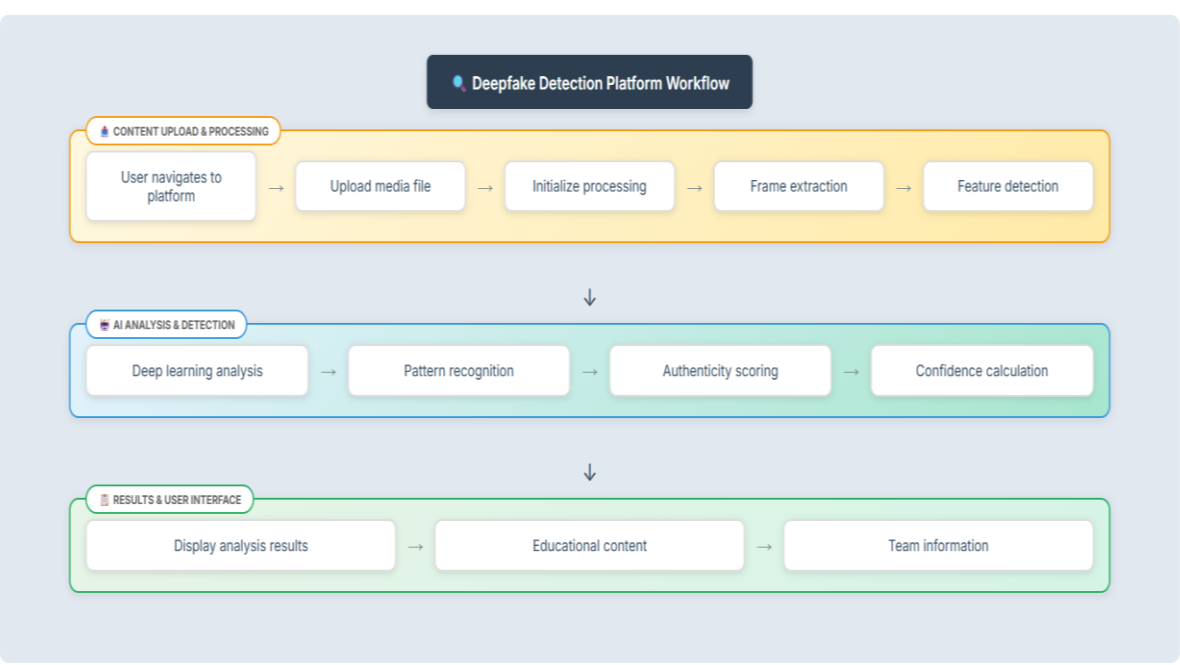
The original and processed versions are shown side by side with a clear indication of the outcome and, if applicable, the detected technique.

Step 4 – Educational Information

After viewing their results, users can read about how similar technology can also be used for positive purposes, giving a balanced view of deepfakes.

Step 5 – Team and Credits

The final section of the platform shows the team members, project title and course information. This workflow keeps the technical processes in the background while presenting a simple, understandable experience to the user and producing reliable, interpretable outputs.

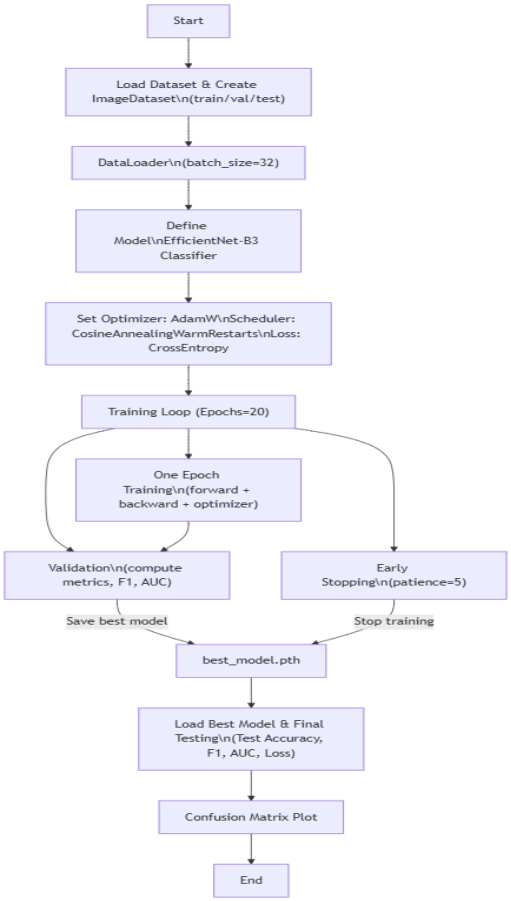


7. Module Design and Implementation

7.1 Notebook Overview

The core image classification model for cataract-related accessibility research was developed and tested in a Jupyter Notebook environment. The notebook contains the following major steps:

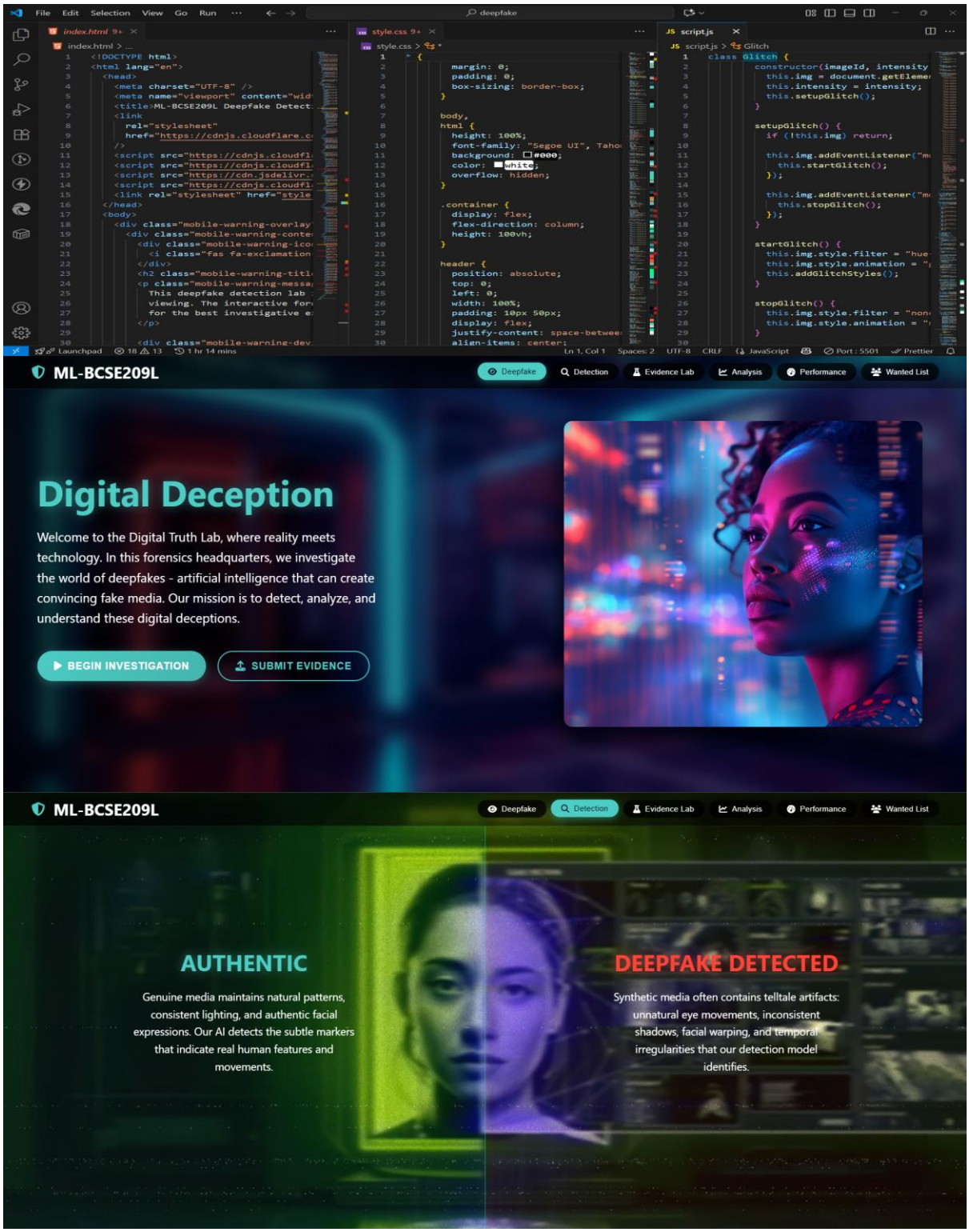
- **Dataset Preparation:** Loading train, validation, and test splits using a custom ImageDataset class.
- **Data Loading:** Batched input handling with DataLoader (batch_size=32) for efficient GPU usage.
- **Model Setup:** Initializing an EfficientNet-B3 classifier with pretrained weights.
- **Optimization:** Using AdamW optimizer, CosineAnnealingWarmRestarts scheduler, and CrossEntropy loss.
- **Training Loop:** Running for 20 epochs with validation, tracking F1-score and AUC.
- **Early Stopping & Saving:** Best weights saved as best_model.pth with a patience of 5 epochs.
- **Evaluation:** Reloading the best model, testing on unseen data, and computing Accuracy, F1, AUC, and Loss.
- **Visualization:** Generating a confusion matrix to analyze classification errors.

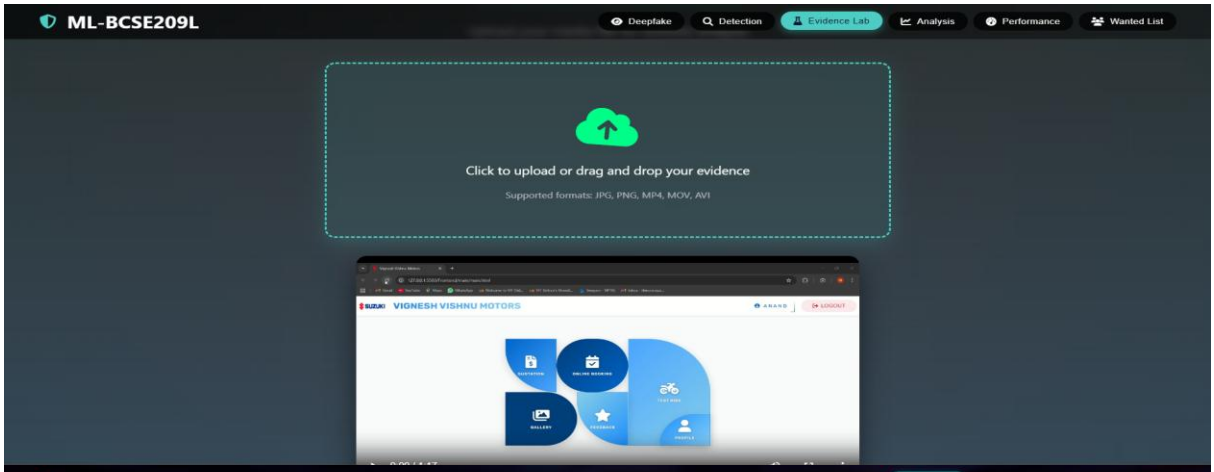


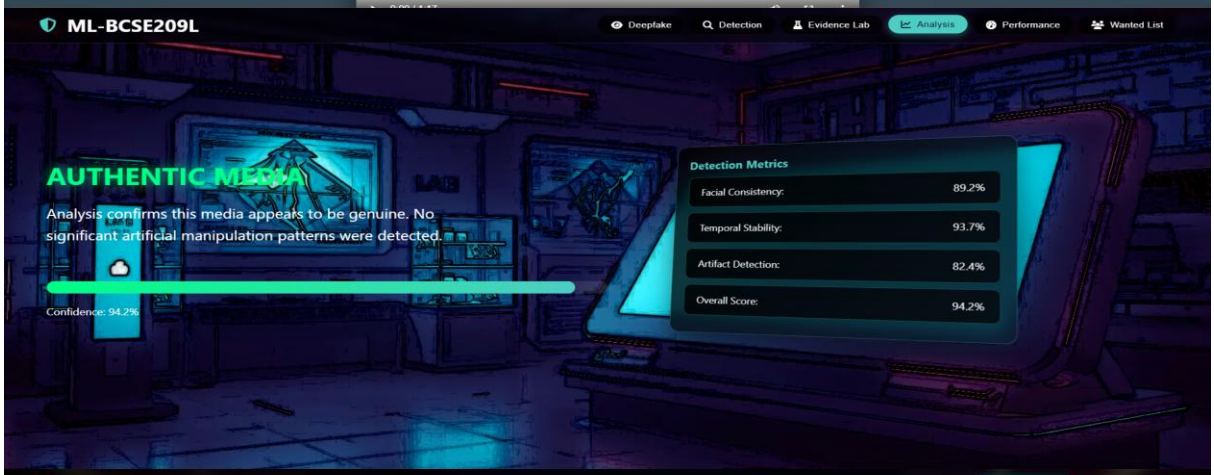
7.2 Output Screenshots

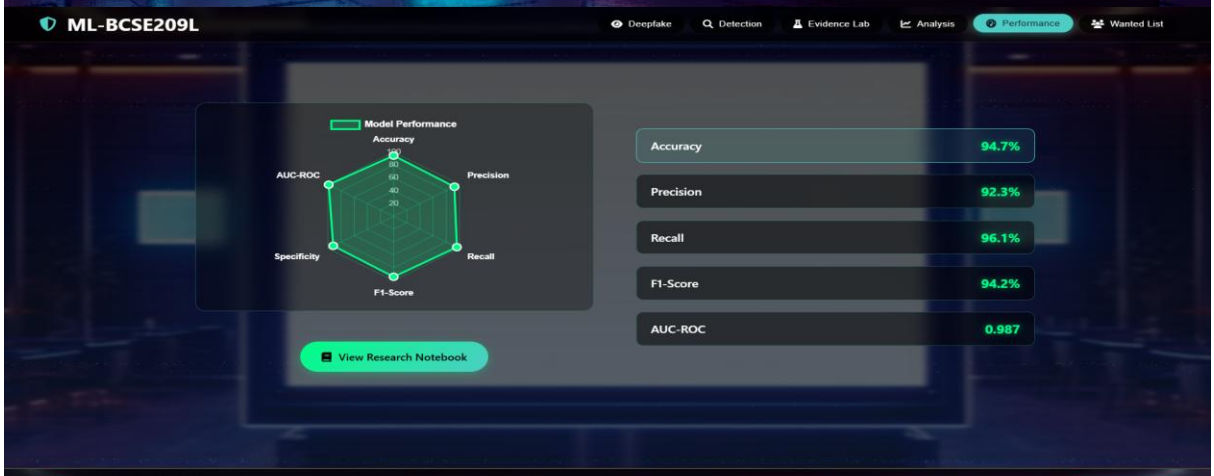
Representative screenshots of the working system will be inserted here to illustrate:

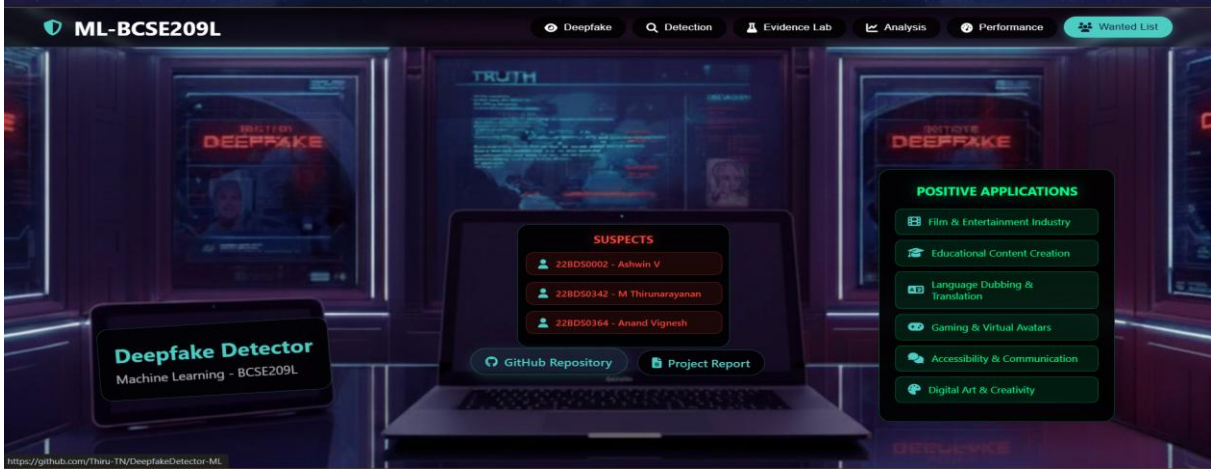
- The upload interface for images/videos.
- Sample detection results (original vs. detected output).
- Classification results showing manipulation type.
- Educational “good use” section showing positive applications.



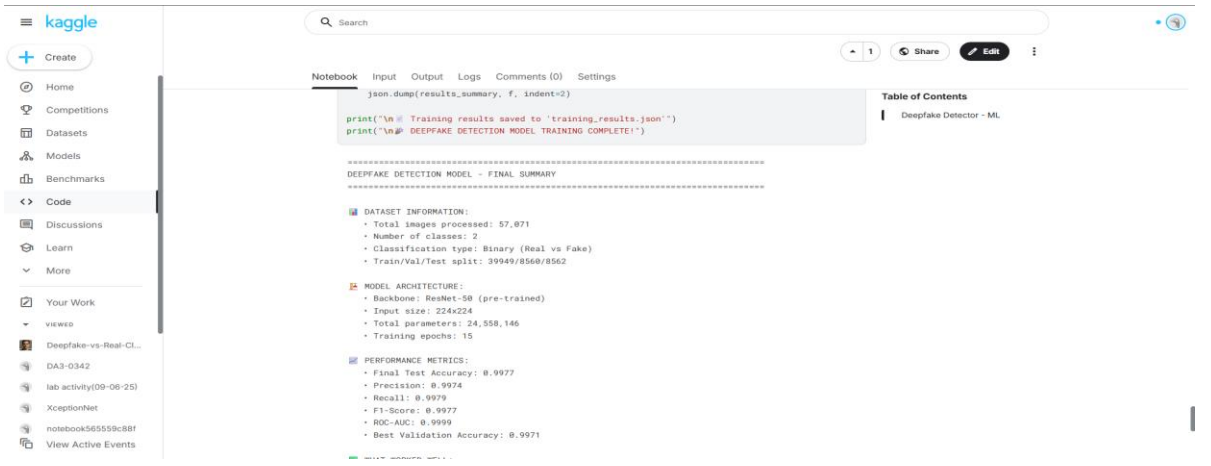

ML-BCSE209L
Deepfake
Detection
Evidence Lab
Analysis
Performance
Wanted List
Click to upload or drag and drop your evidence
Supported formats: JPG, PNG, MP4, MOV, AVI
VIGNESH VISHNU MOTORS

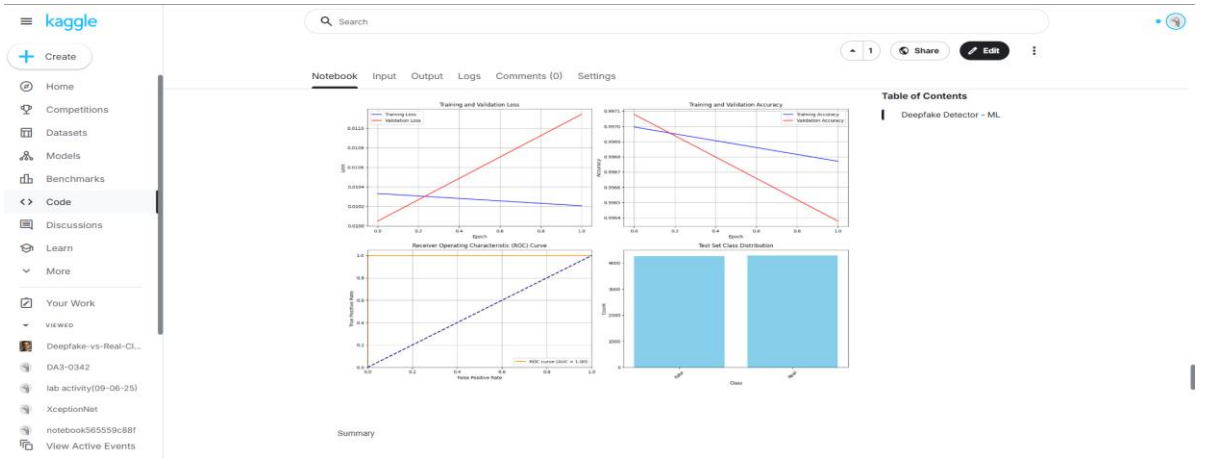

ML-BCSE209L
Deepfake
Detection
Evidence Lab
Analysis
Performance
Wanted List
AUTHENTIC MEDIA
Analysis confirms this media appears to be genuine. No significant artificial manipulation patterns were detected.
Confidence: 94.2%
Detection Metrics
Facial Consistency: 89.2%
Temporal Stability: 93.7%
Artifact Detection: 82.4%
Overall Score: 94.2%


ML-BCSE209L
Deepfake
Detection
Evidence Lab
Analysis
Performance
Wanted List
Model Performance
Accuracy
Precision
Recall
F1-Score
Specificity
AUC-ROC
View Research Notebook
Accuracy 94.7%
Precision 92.3%
Recall 96.1%
F1-Score 94.2%
AUC-ROC 0.987


ML-BCSE209L
Deepfake
Detection
Evidence Lab
Analysis
Performance
Wanted List
TRUTH
SUSPECTS
22BDS0002 - Ashwin V
22BDS0342 - M Thirunarayanan
22BDS0364 - Anand Vignesh
GitHub Repository
Project Report
POSITIVE APPLICATIONS
Film & Entertainment Industry
Educational Content Creation
Language Dubbing & Translation
Gaming & Virtual Avatars
Accessibility & Communication
Digital Art & Creativity
Deepfake Detector
Machine Learning - BCSE209L
https://github.com/Thiru-TN/DeepfakeDetector-ML

```
------------------------------------------------------------

Epoch 1/20
Training: 100%|██████████| 1431/1431 [08:00<00:00,  2.98it/s, Loss=0.4829]
Validation: 100%|██████████| 409/409 [00:38<00:00, 10.73it/s]

Classification Report:
              precision    recall  f1-score   support

        Real       0.91      0.82      0.86      3114
        Fake       0.94      0.97      0.96      9961

    accuracy                           0.94     13075
   macro avg       0.93      0.89      0.91     13075
weighted avg       0.94      0.94      0.94     13075


Train Metrics:
  Loss: 0.2626 | Acc: 0.8911 | F1: 0.9316
  Prec: 0.8931 | Rec: 0.9736 | AUC: 0.9294

Val Metrics:
  Loss: 0.1590 | Acc: 0.9365 | F1: 0.9590
  Prec: 0.9442 | Rec: 0.9743 | AUC: 0.9752
  --> Best model saved!
```





7.3 Links

The following links provide direct access to project resources:

- **GitHub Repository:** https://github.com/Thiru-TN/DeepfakeDetector-ML
- **Website :** https://thiru-tn.github.io/DeepfakeDetector-ML/
- **Presentation Slides:** Google Slides Link
- **Report Draft:** Google Docs Link
- **Model Notebook (Kaggle):** Deepfake Trial Notebook
- **Video:** Demo of Model

---

8. REFERENCES

[1] Z. Huang, J. Hu, X. Li, Y. He, X. Zhao, B. Peng, B. Wu, X. Huang and G. Cheng, "SIDA: Social Media Image Deepfake Detection, Localization and Explanation with Large Multimodal Model," (Dataset: Social-media benchmark of 300,000 images).

[2] K. Somoray, D. J. Miller and M. Holmes, "Human Performance in Deepfake Detection: A Systematic Review," using DFDC, Celeb-DFv2, PEFS, TMC and FaceForensics++ datasets.

[3] L. Lin, X. He, Y. Ju, X. Wang, F. Ding and S. Hu, "Preserving Fairness Generalization in Deepfake Detection," evaluated on FaceForensics++ and DFD datasets.

[4] B. Cavia, E. Horwitz, T. Reiss and Y. Hoshen, "Real-Time Deepfake Detection in the Real-World," introducing the ForenSynths, UFD and WildRF datasets and the LaDeDa algorithm.

[5] F. Cocchi, L. Baraldi, S. Poppi, M. Cornia, L. Baraldi and R. Cucchiara, "Unveiling the Impact of Image Transformations on Deepfake Detection: An Experimental Analysis," using COCOFake dataset.

[6] Preeti, M. Kumar and H. K. Sharma, "A GAN-Based Model of Deepfake Detection in Social Media," based on CelebA dataset.

[7] A. Malik, M. Kuribayashi, S. M. Abdullahi and A. N. Khan, "DeepFake Detection for Human Face Images and Videos: A Survey," covering CelebA, UADFV, DF-TIMIT, FaceForensics, FaceForensics++, DFD and DFDC datasets.

[8] J. Guan, H. Zhou, Z. Hong, E. Ding, J. Wang, C. Quan and Y. Zhao, "Delving into Sequential Patches for Deepfake Detection," on FaceForensics++, DFFD, DFD, DFDC and Celeb-DF datasets.

[9] A. Pokroy and A. Egorov, "EfficientNets for DeepFake Detection: Comparison of Pretrained Models," based on DFDC dataset.

[10] D. Gong, Y. J. Kumar, O. S. Goh, Z. Ye and W. Chi, "DeepfakeNet, an Efficient Deepfake Detection Method," tested on FaceForensics++, Kaggle DFDC and TIMIT datasets.
