Face Detection in a Video

A project report submitted in the fulfillment of the requirement for the award of the degree of Master of Computer Applications (MCA)

Submitted by

Sankrita Patel
Roll No: 22223078
MCA- VI Semester
(Session: 2024-2025)

Submitted to

Dr. Priyanka Tripathi

Head

Department of Computer Applications
National Institute of Technology Raipur

Dr. Naeem Ahmad

Supervisor

Department of Computer Applications
National Institute of Technology Raipur

Department of Computer Applications
National Institute of Technology, Raipur

Department of Computer Applications (संगणक उपयोजन विभाग)
National Institute of Technology Raipur (राष्ट्रीय प्रौद्योगिकी संस्थान रायपुर)
(An Institute of National Importance, Govt. of India)
G.E. Road, Raipur, Chhattisgarh, India, Pin – 492010 www.nitrr.ac.in

DECLARATION

I, **Sankrita Patel, 22223078**, hereby declare that the work done in the project entitled **Face Detection in a Video** is done on my own.

I confirm that:

- The work contained in this report is original and has been done by me under the guidance of **Dr. Naeem Ahmad, Assistant Professor**, Department of Computer Applications, National Institute of Technology Raipur.
- The work has not been submitted to any other institute for any other degree or diploma;
- I have followed the guidelines provided by the institute in preparing the project report;
- I have conformed to ethical norms and guidelines while writing the project report.
- Whenever I have used materials such as data, models, figures, and text from other sources, I have given them due credit by citing them in the report and providing their details in the references.

Place: Raipur

Date:

Sankrita Patel

Roll No: 22223078

MCA-VI Semester

Department of Computer Applications (संगणक उपयोजन विभाग)
National Institute of Technology Raipur (राष्ट्रीय प्रौद्योगिकी संस्थान रायपुर)
(An Institute of National Importance, Govt. of India)
G.E. Road, Raipur, Chhattisgarh, India, Pin – 492010
www.nitrr.ac.in

Certificate from Organization

भारत सरकार
इलेक्ट्रॉनिक और सूचना प्रौधोगिकी मंत्रालय
राष्ट्रीय सूचना-विज्ञान केंद्र (एन० आई० सी०),
जिला इकाई – रायपुर
कक्ष क्रमांक – 15, कलेक्ट्रेट परिसर, रायपुर (छ०ग०)

क्रमांक /6/प्रशिक्षण/2025 रायपुर दिनांक 21/06/2025

इंटर्नशिप सर्टिफिकेट

यह प्रमाणित किया जाता है कि राष्ट्रीय प्रौद्योगिकी संस्थान, रायपुर, छत्तीसगढ़ में अध्यनरत एमसीए की छात्रा **संकृता पटेल**, जिनका रोल नंबर 22223078 है, ने राष्ट्रीय सूचना विज्ञान केंद्र रायपुर में संयुक्त निदेशक (आईटी) और जिला सूचना विज्ञान अधिकारी श्री पी.सी. वर्मा की देखरेख में 01 जनवरी, 2025 से 20 जून, 2025 तक अपनी इंटर्नशिप सफलतापूर्वक पूरी कर ली है।

अपनी इंटर्नशिप के दौरान, उन्होंने **"Face Detection in a Video"** नामक परियोजना में सक्रिय रूप से योगदान दिया और अनुकरणीय समर्पण और क्षमता का प्रदर्शन किया। सुश्री संकृता पटेल ने ईमानदारी और परिश्रम का परिचय देते हुए दिए गए समय सीमा के भीतर अपने सौंपे गए कार्यों को पूरा किया। पूरे प्रोजेक्ट के दौरान उनका आचरण बेहद सराहनीय रहा।

हम संकृता पटेल को उनके भविष्य के प्रयासों के लिए अपनी शुभकामनाएं देते हैं।

21/06/2025
जिला सूचना-विज्ञान अधिकारी
District Informatics Officer
भारत सरकार / Govt. Of India
राष्ट्रीय सूचना-विज्ञान केन्द्र जिला केन्द्र
National Informatics Centre District Centre
रायपुर (छ.ग.) / Raipur (C.G.)

Department of Computer Applications (संगणक उपयोजन विभाग)
National Institute of Technology Raipur (राष्ट्रीय प्रौद्योगिकी संस्थान रायपुर)
(An Institute of National Importance, Govt. of India)
G.E. Road, Raipur, Chhattisgarh, India, Pin – 492010
www.nitrr.ac.in

CERTIFICATE FROM THE SUPERVISOR

This is to certify that the project entitled **Face Detection in a Video** has been carried out by **Sankrita Patel, 22223078**, MCA 6th Semester, under my guidance.

The matter embodied in this project has not been submitted for the award of any other degree or diploma to the best of my knowledge.

Place: Raipur

Date:

_____

(Supervisor signature and seal)

Department of Computer Applications (संगणक उपयोजन विभाग)
National Institute of Technology Raipur (राष्ट्रीय प्रौद्योगिकी संस्थान रायपुर)
(An Institute of National Importance, Govt. of India)
G.E. Road, Raipur, Chhattisgarh, India, Pin – 492010
www.nitrr.ac.in

BONAFIDE CERTIFICATE BY THE HEAD OF THE DEPARTMENT

This is to certify that Mr. **Sankrita Patel,** a student of the Department of Computer Applications, National Institute of Technology, Raipur, Roll No. **22223078,** has carried out the project training at **National Informatics Centre (NIC)** as partial fulfillment of the requirement for the award of the degree of Master of Computer Applications.

H/She has worked in the project entitled **Face Detection in a Video.** Their performance and conduct have been found to be good.

This certificate issued by the undersigned does not cover my responsibility regarding the statements made and work carried out by the concerned student. The current dissertation is hereby being forwarded for evaluation for the purpose for which it has been submitted.

Place: Raipur

Date:

_____

(Head signature and seal)

Department of Computer Applications (संगणक उपयोजन विभाग)
National Institute of Technology Raipur (राष्ट्रीय प्रौद्योगिकी संस्थान रायपुर)
(An Institute of National Importance, Govt. of India)
G.E. Road, Raipur, Chhattisgarh, India, Pin – 492010
www.nitrr.ac.in

CERTIFICATE OF APPROVAL

The forgoing project entitled **Face Detection in a Video** is hereby approved as a creditable work for the partial fulfillment of the requirement for the award of the degree of Master of Computer Applications and has been presented in a satisfactory manner.

This certificate issued by the undersigned does not cover my responsibility regarding the statements made and work carried out by the concerned students. The current dissertation is hereby being forwarded for evaluation for the purpose for which it has been submitted.

Place: Raipur

Date:

_____
(Name and Signature Internal Examiner)

_____
(Name and Signature External Examiner)

Acknowledgement

I would like to express my sincere gratitude to everyone who supported and guided me throughout the development of the **Face Detection in a Video** project.

I am especially thankful to **Dr. Naeem Ahmad** for his valuable mentorship, insightful feedback, and constant encouragement at every stage of this project. His technical expertise and guidance played a crucial role in shaping the final outcome.

I am grateful to the **Dr. P. C. Verma, National Informatics Centre (NIC), Raipur**, for providing me the opportunity to undertake this project as part of my internship. The experience and exposure gained during this period have been immensely enriching and instrumental in the successful completion of this work.

I also extend my appreciation to the faculty and staff of the **National Institute of Technology, Raipur** for providing the necessary infrastructure and a conducive environment for learning and development.

My heartfelt thanks go to my friends and classmates for their continuous support, helpful suggestions, and collaboration during the course of this project. I would also like to acknowledge the developers and contributors of open-source technologies such as **OpenCV**, **Django**, and **React**, which formed the foundation of this application.

Finally, I am deeply grateful to my family for their unwavering support, patience, and motivation, which helped me stay focused and committed throughout this journey.

Table of Content

List of Figures

Face Detection in a Video

Abstract:

This project presents the development of a face detection system that enables users to upload video files and receive output videos with automatically annotated faces. The system integrates a React.js-based frontend with a Django-powered backend, utilizing OpenCV's Haar Cascade classifier for real-time frame-by-frame face detection. Upon receiving a video file, the backend processes each frame to identify facial features and draws bounding boxes around detected faces. The processed frames are then compiled into a new video that is returned to the user. The application emphasizes modular architecture, API-driven communication, and user-friendly design inspired by modern UI frameworks. Performance evaluation demonstrates satisfactory processing times and accurate detection for standard video inputs. The report discusses the implementation details, testing outcomes, system limitations, and potential future enhancements.

Keywords— *Face Detection, Video Processing, Haar Cascade, OpenCV, React.js, Django, Computer Vision, API Integration, Machine Learning*

1. Introduction

1.1. Introduction to the Problem

In the digital age, facial detection has become a key component in security systems, biometric devices, social media applications, and many AI-driven platforms. The ability to detect faces accurately in videos has numerous applications ranging from surveillance to user experience personalization. Despite its prevalence, implementing efficient and accurate face detection in real-world applications remains a challenging task due to variations in lighting, occlusion, facial expressions, and movement.

1.2. Motivation

With the proliferation of video content and the increasing need for automated analysis, a system that can accurately detect faces in user-uploaded videos is highly desirable. Such a system could be used for monitoring, tagging individuals, or anonymizing faces for privacy protection. The motivation behind this project stems from these practical needs and the desire to explore the integration of modern web technologies with machine learning algorithms.

1.3. Objectives of the Face Detection Project

The objectives of **Face Detection in a Video** Project are:

- To build a web-based application where users can upload videos.
- To apply face detection algorithms to each frame of the video.
- To return the processed video with rectangles marking detected faces.
- To utilize React.js for the frontend and Django for the backend.

1.4. Application of Face Detection

The applications of **Face Detection in a Video** project are:

- Security and Surveillance
- Attendance Systems
- Social Media Tagging
- Human-Computer Interaction
- Emotion and Expression Analysis

2. Project Overview

2.1. Problem Definition

Face detection is a computer technology used to identify and locate human faces in digital images or videos. It is a critical step in many facial analysis systems, including facial recognition, emotion detection, and biometric authentication. The technology uses machine learning or deep learning techniques to detect the presence of a face, regardless of its orientation, lighting, or scale.

2.2. Scope of the Project

This project focuses on detecting human faces in video files uploaded by users through a web interface. The detected faces are highlighted with rectangles in the output video. The application is built using a React.js frontend and a Django backend, and uses computer vision libraries such as OpenCV. The project does not include real-time face detection or face recognition capabilities.

2.3. Contribution

The key contributions of this project are:

- A user-friendly web application for video upload and processing.
- Backend logic for frame-by-frame face detection and video reconstruction.
- Integration of modern web technologies with AI-based image processing.
- A modular architecture that allows easy extension for future work.

2.4. Expected Outcomes

The expected Outcomes of this project are:

- A fully functional web interface where users can upload videos.
- Detection of human faces in all frames of the video.
- Output video file with bounding boxes around detected faces.
- Logs and metadata about detection such as number of faces per frame.

2.5. Challenges

The challenges of this project are:

- Processing large video files within a limited time frame.
- Handling variations in face orientation, lighting, and occlusions.
- Ensuring synchronization between frame processing and video reassembly.
- Maintaining a responsive and efficient user interface

3. System Models

3.1. System Architecture

The system is composed of three major components:

- Frontend (React.js): Allows users to upload videos and displays the processed output.
- Backend (Django): Handles requests, video frame processing, face detection, and reconstruction.
- Face Detection Engine (OpenCV + Python): Performs the actual face detection using Haar Cascades.

3.2. Software Stack

The essential software technology that is used in this project are:

- Frontend: React.js, JavaScript, CSS, Axios
- Backend: Django, Django REST Framework
- Face Detection: Python, OpenCV
- Video Processing: OpenCV, FFmpeg
- Deployment: Local server (can be extended to cloud platforms like AWS or Heroku)

3.3. Libraries & Tools

The essential Libraries & Tools that is used in this project are:

- OpenCV: For image and video processing.
- FFmpeg-python: For handling video encoding/decoding.
- Axios: For HTTP requests between frontend and backend.
- Bootstrap/Tailwind CSS: For UI styling (inspired by the UI at uxpilot.ai).

3.4. Hardware Requirements

The essential hardware requirements are:

- Processor: Intel i5 or higher
- RAM: Minimum 8 GB
- GPU: Optional (for accelerated processing)
- Storage: Minimum 5 GB free space
- OS: Windows/Linux/macOS

3.5. Platform Details

The system is designed to run on standard web browsers for the frontend. The backend is built on Django and runs on a local development server but is compatible with deployment on any WSGI-compliant web server. Video files are temporarily stored for processing and removed after results are returned to maintain privacy.

3.6. Software Development Methodology

To ensure systematic development and timely delivery, the **Agile Software Development** methodology is adopted:

3.6.1. Iterative Development

Features are developed in short sprints (1–2 weeks), allowing for continuous feedback and improvements.

3.6.2. Daily Standups & Task Boards

Helped track progress and remove blockers efficiently.

3.6.3. Collaboration

Frequent interactions between frontend, backend, and testing teams ensured alignment with project goals.

3.6.4. Flexibility

Agile allowed easy incorporation of changes based on feedback from faculty and users.

3.6.5. Testing & Integration

Continuous testing and integration ensured system stability throughout the development cycle.

4. Methodology

4.1. Face Detection Algorithm

This project uses the **Haar Cascade Classifier** provided by OpenCV for face detection. The Haar Cascade classifier is a machine learning-based approach where a cascade function is trained with a large number of positive and negative images. It scans an image at multiple scales and locations to detect features like edges, lines, and textures commonly associated with human faces.

4.2. Workflow

The following steps outline the face detection process:

1. **Video Upload**: Users upload videos via the frontend interface built in React.js.
2. **Frame Extraction**: The Django backend extracts individual frames from the uploaded video using OpenCV.
3. **Face Detection**: Each frame is scanned for faces using the Haar cascade classifier.
4. **Rectangle Drawing**: Bounding boxes are drawn around each detected face.
5. **Frame Storage**: The processed frames are stored temporarily.
6. **Video Reconstruction**: Frames are reassembled into a processed video with face rectangles.
7. **Output Delivery**: The final video is returned to the frontend for user download.

4.3. Proposed Pipeline

The complete face detection pipeline is structured as follows:

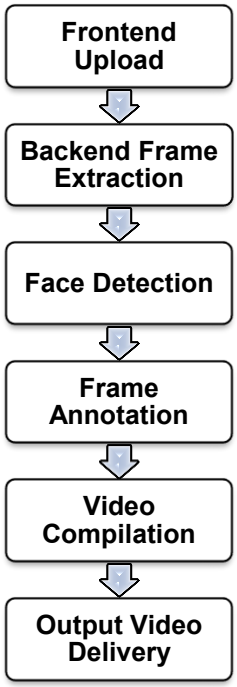


Figure 1 Pipeline of Face Detection Algorithm

This modular approach improves maintainability and allows for future scalability, such as adding support for real-time detection or face recognition.

5. Implementation

5.1. Flow Chart

5.1.1. Flowchart for User Authentication

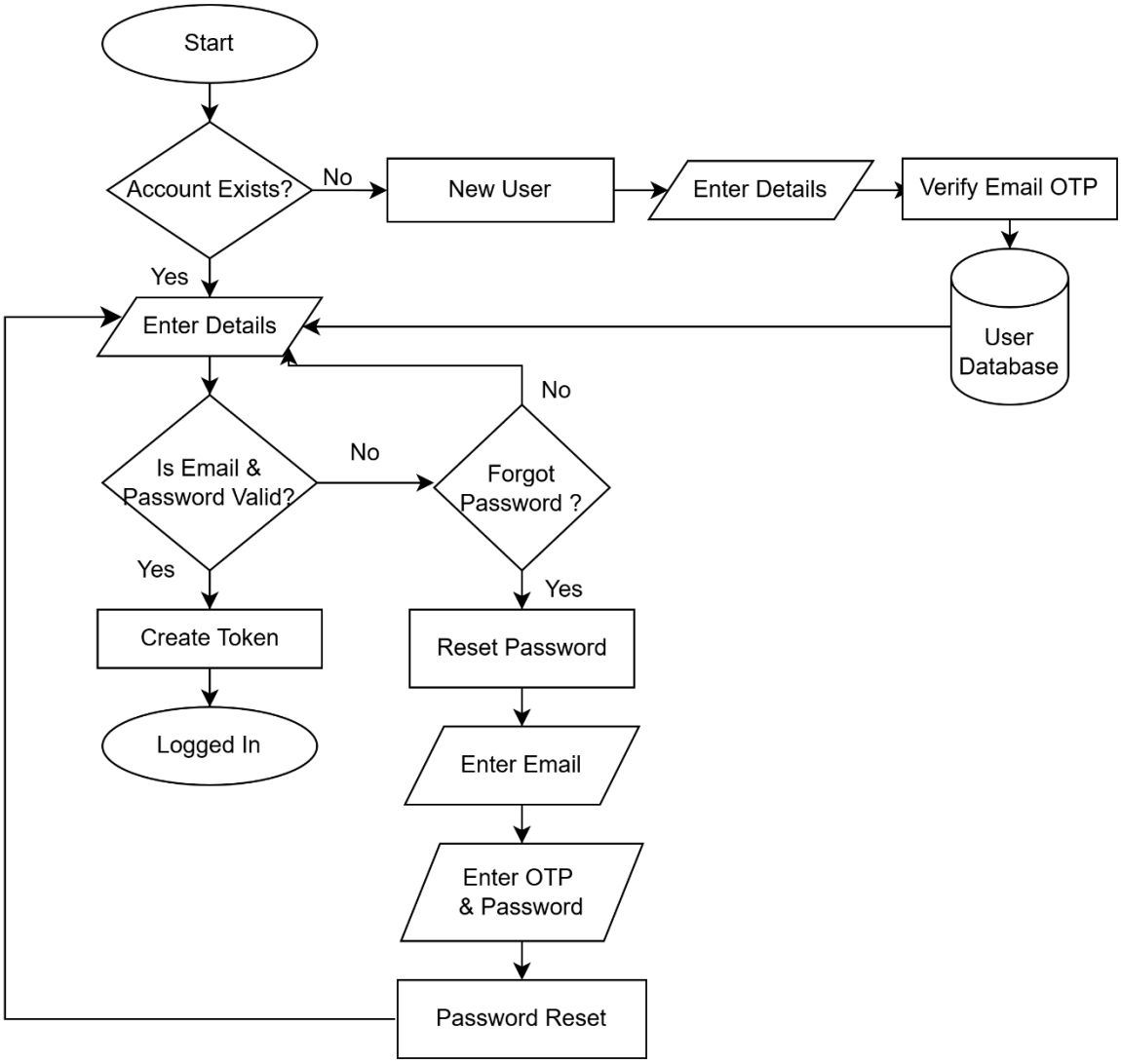


Figure 2 Flow-Chart of User Authentication

5.1.2. Home Page

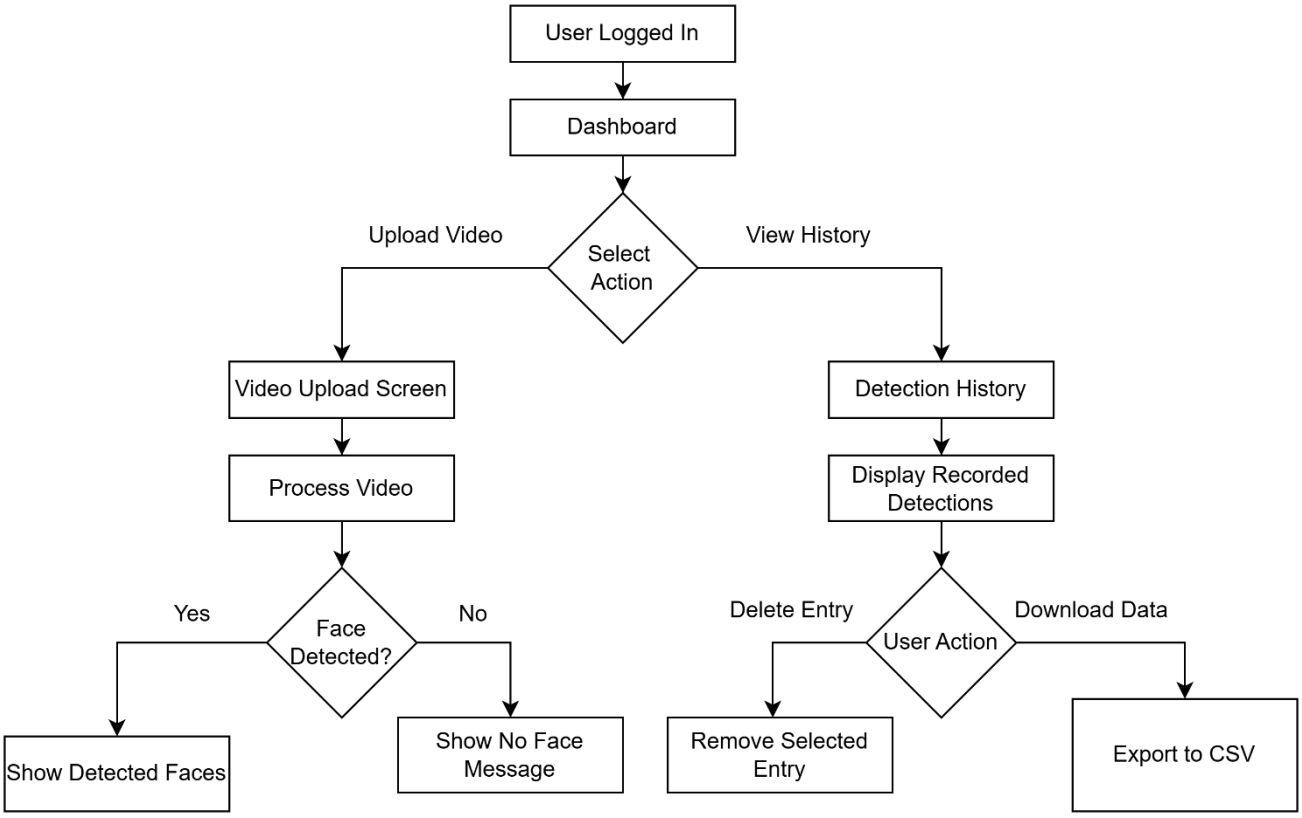
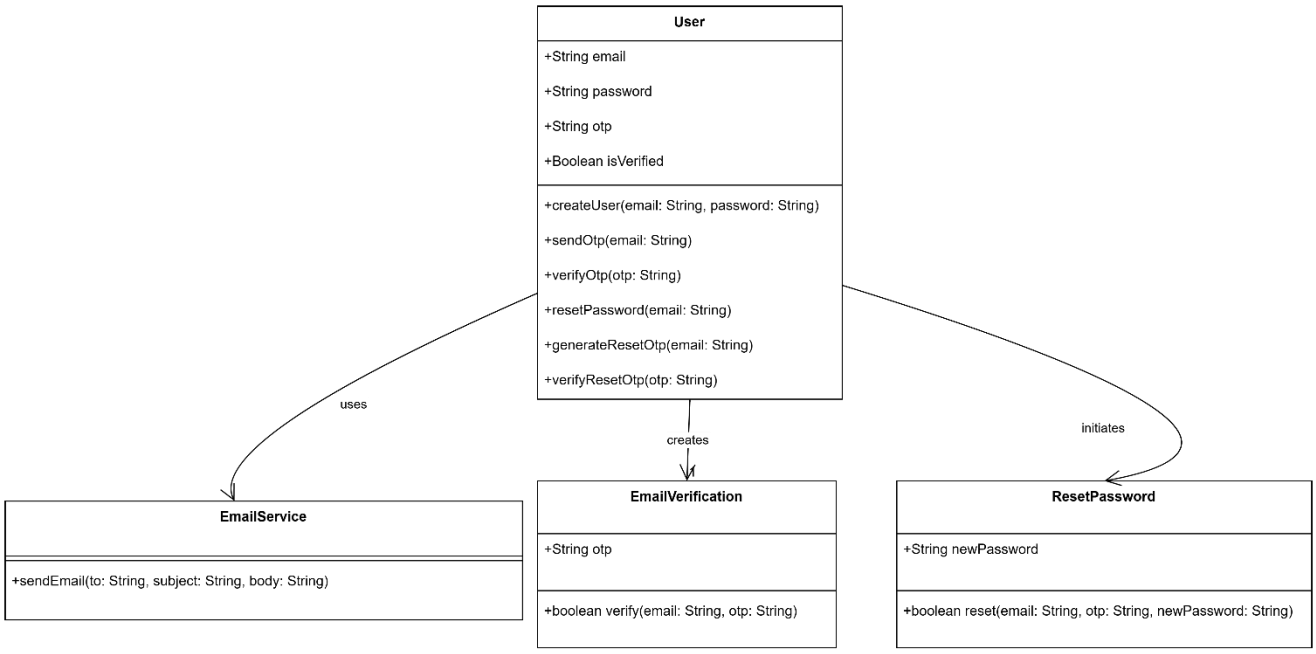


Figure 3 Flow-Chart of Home Page

5.2. UML Diagram

5.2.1. UML Diagram for User Authentication



5.3. Code

5.3.1. Frontend

The frontend was developed using React.js to provide a dynamic, responsive user interface. The UI allows users to upload a video file, monitor the upload status, and download the processed video with face annotations.

5.3.2. Backend

The backend is built with Django and Django REST Framework, providing API endpoints to receive the video, process it, and return the output.

5.3.3. Video Processing Logic

Step 1: Frame Extraction

$$vidcap = cv2.VideoCapture(input\_path)$$

$$success, image = vidcap.read()$$

```
while success:
  frame_list.append(image)
  success, image = vidcap.read()
```

Step 2: Face Detection

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.1, 4)
for (x, y, w, h) in faces:
  cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
```

Step 3: Reconstruct Video

```
out = cv2.VideoWriter(output_path, cv2.VideoWriter_fourcc(
            *'mp4v'), fps, (width, height))
for frame in processed_frames:
  out.write(frame)
out.release()
```

5.3.4. Integration of Frontend and Backend

Integration is achieved using Axios and Django CORS headers.

Axios (React)

```
const response = await axios.post(
    'http://localhost:8000/api/face-detection/',
    formData,
    {
     headers: {
      'Content-Type': 'multipart/form-data',
     },
     responseType: 'blob',
    }
   );
```

CORS Handling (Django)

```
pip install django-cors-headers
INSTALLED_APPS += ['corsheaders']
MIDDLEWARE = ['corsheaders.middleware.CorsMiddleware'] + MIDDLEWARE
CORS_ALLOW_ALL_ORIGINS = True
```

5.3.5. Error Handling and Validation

Frontend Validations

- File type and size
- Empty input

Backend Validations

- MIME type verification
- Exception handling

Django Error Handling Example

```
try:
  # video processing logic
except Exception as e:
  return Response({'error': str(e)}, status = 500)
```

6. Results and Discussion

6.1. Outcomes

6.1.1. Accuracy of Detection

The face detection model used in this project is based on Haar Cascade Classifiers, which provide a decent level of accuracy in detecting frontal faces. However, the accuracy depends on the lighting, video quality, face orientation, and presence of occlusions.

Testing Results:

- High accuracy for clear, well-lit faces.
- Reduced detection rate for side-profile or partially visible faces.
- False positives were minimal but occurred occasionally in background elements.

6.1.2. Performance Evaluation

The application was tested on various videos of different lengths and resolutions. The frame processing rate ranged between 10-25 FPS depending on the system resources.

Performance Observations:

- Short videos (<30s): processed in under 10 seconds.
- Long videos (1-2 minutes): processed in 20–45 seconds.
- Optimizations using threading and efficient memory usage improved performance.

6.1.3. Limitations

- Limited detection of non-frontal faces.
- Performance may degrade on low-end hardware.
- Detection accuracy is affected by poor lighting and blurry frames.
- No support for real-time video feed processing (yet).

6.2. Experimental Results

6.2.1. User Registration Page

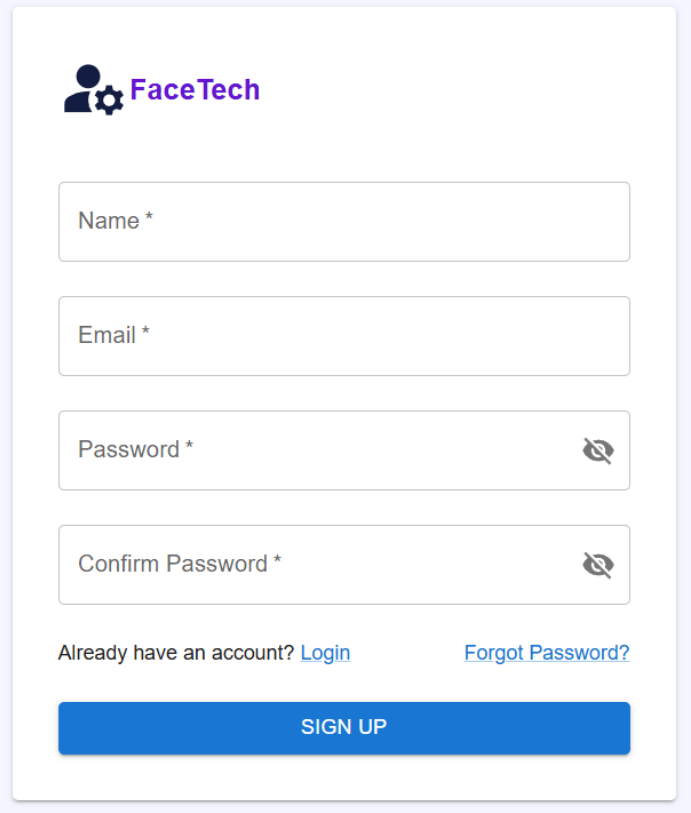


Figure 4 User Registration Page

6.2.2. Login Page

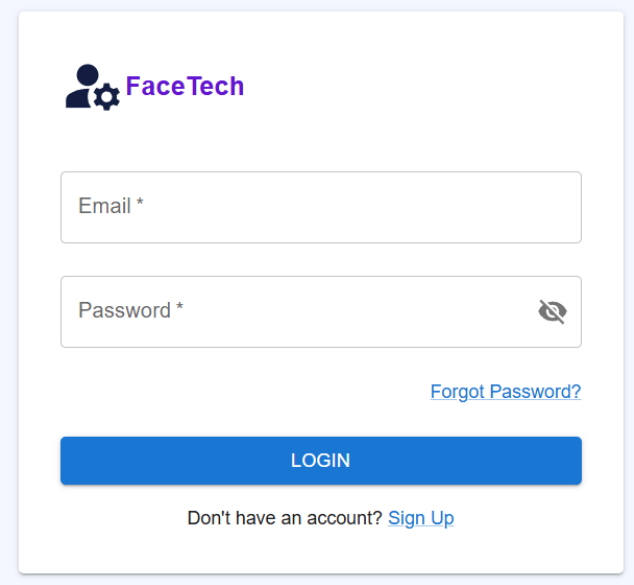


Figure 5 User Login Page

6.2.3. Forgot Password

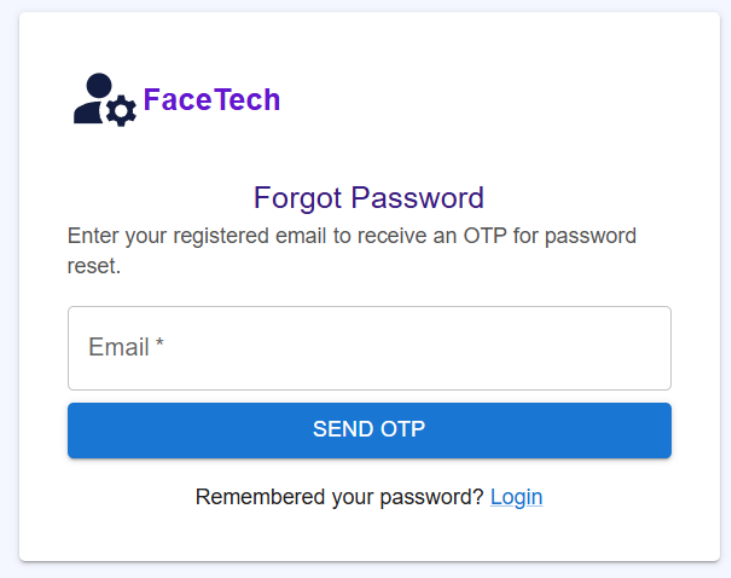


Figure 6 User Forgot Password

6.2.4. Reset Password

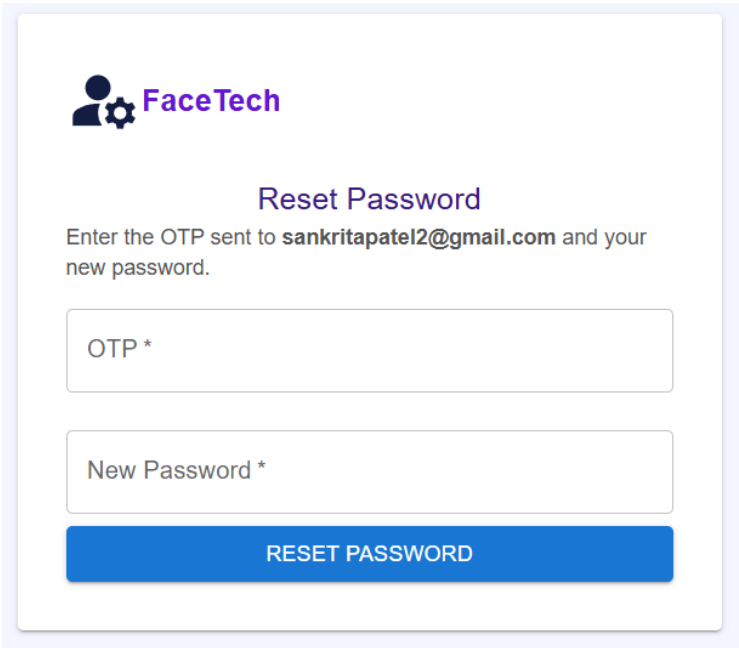


Figure 7 User Reset Password

6.2.5. Upload Video

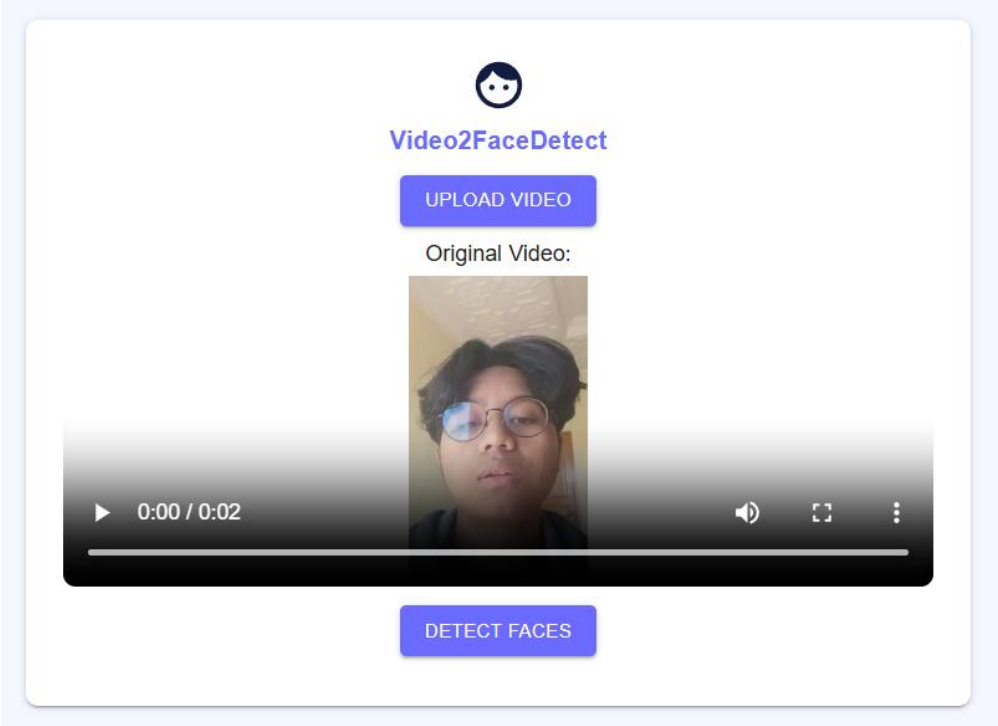


Figure 8 Upload Video Screen

6.2.6. Processed Video

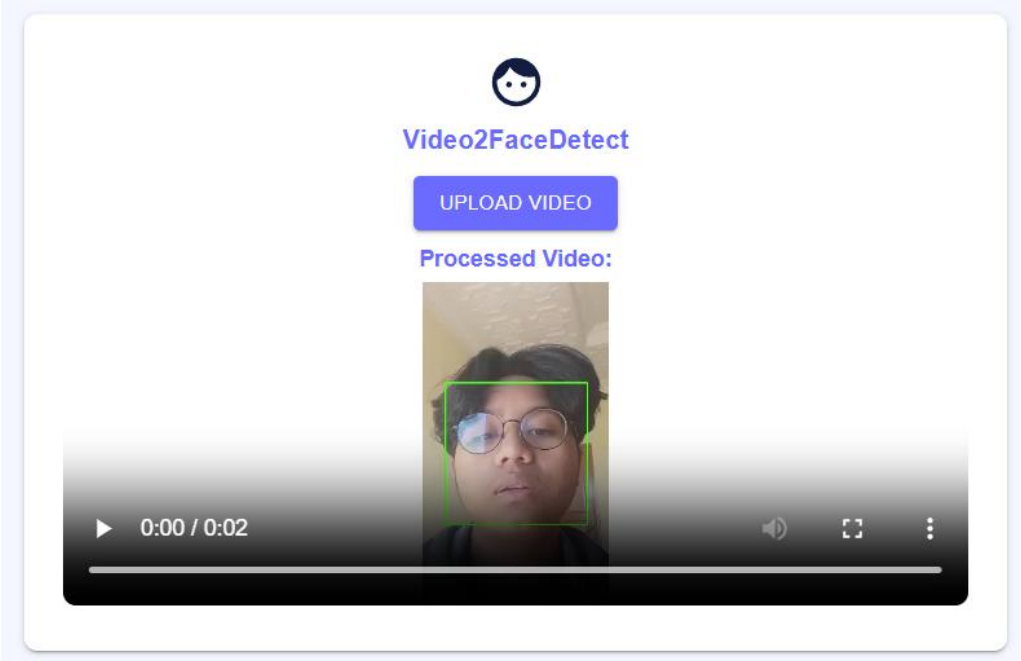


Figure 9 Face Detection Result Page

7. Conclusion and Future Work

7.1. Conclusion

The Face Detection System developed as part of this project demonstrates the capability of integrating computer vision with modern web technologies. The system successfully accepts user-uploaded videos, detects faces frame by frame using the Haar Cascade algorithm, and produces a video output with rectangles around each detected face.

The use of **React.js** on the frontend provided a clean, responsive user interface, while **Django** ensured robust backend API handling and video processing logic. Despite some limitations in detection accuracy for non-frontal or low-quality frames, the system performs well under standard conditions.

In terms of performance, the application achieves satisfactory processing times for short and medium-length videos, making it suitable for offline batch processing scenarios. The comparative analysis with other tools also provides direction for future improvements in accuracy and performance.

7.2. Future Enhancements

- Integrate real-time webcam-based face detection to support live monitoring and streaming applications.
- Use more advanced models like MTCNN, SSD, or YOLO to improve detection accuracy and robustness under varied conditions.
- Provide user login and history tracking, enabling users to access previously processed videos.
- Improve UI/UX based on iterative testing and user feedback, inspired by modern designs such as the one referenced from UXPilot.
- Deploy the application using scalable cloud platforms like AWS or GCP for broader accessibility and reliability.

8. References

- Bradski, G. (2000). *The OpenCV Library*. Dr. Dobb's Journal of Software Tools.
- Django Software Foundation. (2024). *Django Documentation*. Available at: https://docs.djangoproject.com
- Facebook Inc. (2024). *React – A JavaScript Library for Building User Interfaces*. Available at: https://reactjs.org
- GitHub Repository – Haar Cascades for Face Detection. Available at: https://github.com/opencv/opencv/tree/master/data/haarcascades
- UXPilot.ai. (2025). *UI Design Inspirations*. Available at: https://uxpilot.ai/a/ui-design?page=cVPAcPefA09ARbKLoOcc
- Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.
- Dlib C++ Library. (2024). *High Quality Face Recognition with Deep Metric Learning*. Available at: http://dlib.net/