**FACIAL EMOTION RECOGNITION**

**A PROJECT REPORT**

**for**

**Introduction To AI (AI101B)**

**Session (2024-25)**

**Submitted by**

**NISHANT RAIKWAR**

**(202410116100135)**

**PALAK GOYAL**

**(202410116100138)**

**RAHUL RAWAT**

**(202410116100159)**

**NILESH AGRAHARI**

**(202410116100134)**

**Submitted in partial fulfilment of the**

**Requirements for the Degree of**

**MASTER OF COMPUTER APPLICATION**

**Under the Supervision of**

**Ms. Komal Salgotra**

**Assistant Professor**



**Submitted to**

**Department Of Computer Applications**

**KIET Group of Institutions, Ghaziabad**

**Uttar Pradesh-201206**

**(APRIL, 2025)**

CERTIFICATE

Certified that **Nishant Raikwar (202410116100135) ,Palak Goyal (202410116100138), Rahul Rawat (202410116100159) and Nilesh Agrahari (202410116100134)** have carried out the project work having “**Facial Emotion Recognition**” (INTRODUCTION TO AI) (AI101B)) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU**)** (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:21-04-2025

**Ms. Komal Salgotra Dr. Akash Rajak**

**Assistant Professor Dean**

**Department of Computer Applications Department of Computer Applications**

**KIET Group of Institutions, Ghaziabad KIET Group of Institutions, Ghaziabad**

**An Autonomous Institutions An Autonomous Instituti**

FACIAL EMOTION RECOGNITION

**NISHANT RAIKWAR**

**PALAK GOYAL**

**RAHUL RAWAT**

**NILESH AGRAHARI**

ABSTRACT

Facial Emotion Recognition (FER) is a new technology in the areas of artificial intelligence and computer vision that allows machines to read and react to human emotions from facial expressions. The techniques and models employed to identify and classify emotions from facial features by image processing and machine learning algorithms are investigated in this report. FER systems generally include stages of face detection, feature extraction, and emotion classification, implemented with technologies like Convolutional Neural Networks (CNNs), OpenCV, and deep learning frameworks. Emotional recognition has a broad range of applications in fields including human-computer interaction, healthcare, education, surveillance, and marketing. This project proposes to implement an efficient FER system that is able to detect basic human emotions such as happiness, sadness, anger, surprise, fear, and disgust with good accuracy. The objective is to improve the relationship between humans and machines by rendering them more responsive and emotionally aware.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Ms. Komal Salgotra** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Akash Rajak, Professor and Dean, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Nishant Raikwar**

**Palak Goyal**

**Rahul Rawat**

**Nilesh Agrahari**

TABLE OF CONTENTS

Table of Contents

[CERTIFICATE ii](#_Toc196223065)

[ABSTRACT iii](#_Toc196223066)

[ACKNOWLEDGEMENTS iv](#_Toc196223067)

[TABLE OF CONTENTS v](#_Toc196223068)

[1 INTRODUCTION 6](#_Toc196223069)

[2 OBJECTIVE OF THE PROJECT 7](#_Toc196223070)

[2.1 Facial emotion detector (Python + DL) 7](#_Toc196223071)

[2.2 Frontend (HTML, CSS, JS) 7](#_Toc196223072)

[2.3 Image upload & real‑time webcam 8](#_Toc196223073)

[2.4 Deployment (free cloud) 8](#_Toc196223074)

[3 SYSTEM REQUIREMENTS 9](#_Toc196223075)

[3.1 HARDWARE REQUIREMENTS 9](#_Toc196223076)

[3.2 SOFTWARE REQUIREMENTS 9](#_Toc196223077)

[4 EXISTING SYSTEM AND LIMITATIONS 10](#_Toc196223078)

[4.1 Microsoft Azure Face (Emotion) API 10](#_Toc196223079)

[4.2 Affectiva Emotion SDK 11](#_Toc196223080)

[4.3 Common Constraints Across Platforms 11](#_Toc196223081)

[5 PROPOSED SYSTEM 12](#_Toc196223082)

[5.1 System Architecture 12](#_Toc196223083)

[5.1.1 Client (Frontend): 12](#_Toc196223084)

[5.1.2 Server (Backend): 13](#_Toc196223085)

[5.2 Key Features 13](#_Toc196223086)

[5.2.1 Web‑Based Interface with Dual Input Modes 13](#_Toc196223087)

[5.2.2 Emotion Recognition via Pre‑trained DeepFace Models 13](#_Toc196223088)

[5.2.3 Interactive UX Elements 13](#_Toc196223089)

[5.2.4 Responsive Layout 13](#_Toc196223090)

[5.2.5 Live Deployment on Render 14](#_Toc196223091)

[5.3 Module Descriptions 14](#_Toc196223092)

[5.4 Advantages of the Proposed System 14](#_Toc196223093)

[6 SYSTEM DESIGN 15](#_Toc196223094)

[6.1 Architecture Diagram 15](#_Toc196223095)

[6.2 Module Breakdown 15](#_Toc196223096)

[7 SYSTEM IMPLEMENTATION 17](#_Toc196223097)

[7.1 Key Files 17](#_Toc196223098)

[7.2 Tools Used 18](#_Toc196223099)

[8 RESULTS AND ANALYSIS 19](#_Toc196223100)

[8.1 Supported Emotions 19](#_Toc196223101)

[8.2 SCREENSHOTS OF RESULTS 20](#_Toc196223102)

[8.3 Accuracy 21](#_Toc196223103)

[8.4 Performance 21](#_Toc196223104)

[9 TESTING AND EVALUATION 22](#_Toc196223105)

[9.1 Manual Test Cases 22](#_Toc196223106)

[9.2 Limitations 23](#_Toc196223107)

[10 CONCLUSION AND FUTURE SCOPE 24](#_Toc196223108)

[10.1 Conclusion 24](#_Toc196223109)

[10.2 Future Scope 24](#_Toc196223110)

[11 REFERENCES 25](#_Toc196223111)

**CHAPTER 1**

# 1 INTRODUCTION

Facial expressions are a vital part of non-verbal communication, offering insight into a person's emotions, mood, and intentions. Recognizing these emotions accurately through technology has become an important area of research in the fields of artificial intelligence (AI), computer vision, and human-computer interaction. Facial Emotion Recognition (FER) is a technique that enables machines to automatically detect and classify human emotions from facial expressions. This has significant applications in areas such as mental health analysis, personalized user experiences, surveillance, customer service, and e-learning platforms.

With the advancement in deep learning algorithms and increased computational power, it is now possible to analyze facial features in real time and achieve high levels of accuracy in emotion detection. The emergence of open-source libraries and frameworks has further simplified the development of such systems, making them more accessible for developers and researchers.

This project aims to design and implement a web-based facial emotion recognition system capable of detecting emotions from both uploaded images and live webcam feeds. The system uses DeepFace, an open-source facial recognition and analysis library that integrates several state-of-the-art deep learning models. For the backend, Django is used to manage server-side logic and user interactions, while the application is deployed on Render, ensuring easy access and scalability.

**CHAPTER 2**

# 2 OBJECTIVE OF THE PROJECT

## 2.1 Facial emotion detector (Python + DL)

Data & preprocessing: Use FER‑2013 (48×48 grayscale), normalize and augment (flips, rotations).

Face detection: OpenCV/Dlib to crop faces.

Model: Simple CNN (Conv–Pool → Dense → Softmax for 7 emotions) or fine‑tune MobileNetV2.

Train & eval: 70/15/15 split, Adam optimizer, early stopping; report accuracy and confusion matrix.

## 2.2 Frontend (HTML, CSS, JS)

Layout:

Home page with image‑upload button

Live‑camera page with video + canvas overlay

Interactions:

File input + preview → POST /predict/image

getUserMedia() → capture frames → POST /predict/webcam every ~200 ms

Show emotion label & confidence on page

## 2.3 Image upload & real‑time webcam

Upload flow: User selects image → JS reads via FileReader → send multipart/form‑data → display JSON result.

Webcam flow: Stream video → draw frame to <canvas> → convert to dataURL → POST → update label.

Tips: Resize to ~160×120, throttle to ~5 FPS, use Web Workers if needed.

## 2.4 Deployment (free cloud)

Platforms: Heroku or Render (free tiers)

Setup:

Procfile + requirements.txt (or Dockerfile)

GitHub‑linked auto‑deploy

Env vars for model path, secrets

Extras: Free subdomain with auto SSL; monitor via platform logs/metrics.

**CHAPTER 3**

# 3 SYSTEM REQUIREMENTS

## 3.1 HARDWARE REQUIREMENTS

* **Processor:** Intel Core i5 or higher
* **RAM:** 8 GB or more
* **Hard Disk:** Minimum 500 MB free
* **Camera:** Integrated or external webcam

## 3.2 SOFTWARE REQUIREMENTS

* **OS:** Windows 10/11, Linux
* **Language:** Python 3.10
* **IDE:** Visual Studio Code
* **Backend Framework:** Django
* **Libraries:** DeepFace, OpenCV, NumPy, Pillow, Gunicorn
* **Hosting Platform:** Render

**CHAPTER 4**

# 4 EXISTING SYSTEM AND LIMITATIONS

Many off‑the‑shelf emotion‑recognition services exist today—two of the most widely used being Microsoft Azure’s Face (Emotion) API and the Affectiva Emotion SDK. While these platforms offer ready‑made pipelines for detecting facial expressions, they introduce several challenges for smaller teams or highly customized deployments:

## 4.1 Microsoft Azure Face (Emotion) API

* **Overview:** Part of Azure Cognitive Services, the Face API can detect up to eight basic emotions (anger, contempt, disgust, fear, happiness, neutral, sadness, surprise) from a face bounding box, returning confidence scores for each.
* **Pricing:**
  + **Free tier:** ~30 K transactions/month, but limited to 1 transaction/sec throughput.
  + **Standard tier:** $1.50 per 1 000 transactions (region‑dependent), with a minimum commitment and additional charges for higher throughput.
* **Limitations for small developers:**
  + **Costly at scale:** Even a moderate demo (10 000 analyses/month) costs $15+; costs rise sharply if you need real‑time video (frames × rate).
  + **Cloud‑only processing:** All images must be sent to Microsoft’s servers—no truly offline option—requiring stable, high‑bandwidth Interne
  + **Hardware dependency:** To meet real‑time SLAs, you often need to provision higher‑tier (and more expensive) cloud VMs.

## 4.2 Affectiva Emotion SDK

* **Overview:** A specialized, cross‑platform SDK (Windows, Linux, Android, iOS) that runs emotion analysis locally or on‑device. It uses deep‑learning models trained on millions of faces to output 20+ affective metrics (valence, engagement, facial action units).
* **Pricing & licensing:**
  + **Developer license:** Begins at $5 000/year, with usage caps and per‑seat fees.
  + **Enterprise/volume discounts:** Available only under negotiated contracts—out of reach for many student or indie projects.
* **Limitations for small developers:**
  + **High upfront fees:** The entry‑level license is cost‑prohibitive for academic or hobbyist use.
  + **Hardware requirements:** On‑device inference needs a GPU or high‑end CPU to maintain ≥15 FPS; otherwise, latency spikes.
  + **Opaque model tweaking:** No access to modify underlying neural‑network layers—only high‑level parameter tuning (e.g., detection thresholds).

## 4.3 Common Constraints Across Platforms

* **Expense & Licensing Overhead**
  + Recurring costs (subscription or per‑seat) make long‑term maintenance expensive for student or indie teams.
* **Compute & Bandwidth Demands**
  + Real‑time (video) inference on cloud platforms multiplies per‑image fees and necessitates constant high‑speed connectivity.
  + On‑device SDKs require GPUs or powerful multicore CPUs, inflating hardware budgets.

**CHAPTER 5**

# 5 PROPOSED SYSTEM

This chapter outlines the design and implementation of the proposed open‑source facial emotion recognition system. By leveraging the DeepFace library and modern web technologies, the system delivers a fully functional, user‑friendly interface along with reliable emotion detection models and seamless deployment.

## 5.1 System Architecture

### 5.1.1 Client (Frontend):

* + **HTML/CSS/JavaScript**: Provides a responsive, mobile‑friendly interface. Tailwind CSS (or Bootstrap) is used for fast styling and layout management.
  + **File Upload Component**: Allows users to select and preview images before analysis.
  + **Webcam Module**: Utilises the getUserMedia() API to capture live video streams in the browser.
  + **Spinner & Preview**: A loading spinner overlays the preview area during model inference; once the result is returned, the preview image or current video frame is annotated with the predicted emotion and confidence score.

### 5.1.2 Server (Backend):

* + **Flask/FastAPI**: Hosts RESTful endpoints for image and webcam prediction (/predict/image, /predict/webcam).
  + **DeepFace Integration**: Uses DeepFace’s pre‑trained models (VGG‑Face, Facenet, or OpenFace) to extract facial embeddings and classify emotions.
  + **Inference Pipeline**:
    1. Receive image or base64‑encoded frame.
    2. Detect and align face(s) with DeepFace’s detector.
    3. Compute emotion probabilities via DeepFace.analyze().

## 5.2 Key Features

### 5.2.1 Web‑Based Interface with Dual Input Modes

* + **Static Image Upload**: Users can choose an image file, view a live preview, and trigger analysis with one click.
  + **Live Webcam Analysis**: Real‑time emotion recognition at up to 5 frames per second, with automatic throttling for performance.

### 5.2.2 Emotion Recognition via Pre‑trained DeepFace Models

* + Leverages open‑source models fine‑tuned on large emotion datasets.
  + Supports seven universal expressions (anger, disgust, fear, happiness, sadness, surprise, neutral).

### 5.2.3 Interactive UX Elements

* + **Spinner Animation**: Indicates processing state during API calls.
  + **Preview Canvas**: Overlays bounding boxes and labels directly on the image or video.

### 5.2.4 Responsive Layout

* + Mobile and desktop support via CSS grid and flexbox.
  + Media queries adjust component sizes and stacking order for optimal viewing on any device.

### 5.2.5 Live Deployment on Render

* + **Free Web Service**: Automatically builds from a connected GitHub repository.
  + **Auto‑Provisioned SSL**: Ensures secure HTTP traffic.
  + **Continuous Deployment**: New commits to the main branch trigger rebuilds and redeployments.

## 5.3 Module Descriptions

* **Frontend UI**  
  Developed using HTML, CSS, and JavaScript. Responsible for handling all user interactions, including image upload, webcam access, and displaying the detected emotions on the screen.
* **API Endpoints**  
  Built using Flask or FastAPI. These endpoints handle incoming requests such as /predict/image and /predict/webcam, validate the input, and forward the data to the emotion analysis module.
* **Emotion Engine**  
  Utilizes the DeepFace library to perform face detection, extract facial embeddings, and classify emotions using pre-trained deep learning models.
* **Utility Services**  
  Uses OpenCV and NumPy for preprocessing tasks such as face alignment, image resizing, normalization, and converting image formats for accurate emotion detection.
* **Deployment Script**  
  Employs Docker and Render.com to containerize the application and automate deployment on the cloud through Render’s free hosting service.

## 5.4 Advantages of the Proposed System

* **Cost‑Effective**  
  Leverages open‑source DeepFace models (MIT license), eliminating per‑transaction or seat‑based fees.
* **Customizable**  
  Full access to model parameters, inference code, UI components, and deployment scripts—enabling tweaks at every layer.
* **Cross‑Platform**  
  Frontend runs in any modern browser; backend API can operate on CPU‑only servers or low‑end VMs.

**CHAPTER 6**

# 6 SYSTEM DESIGN

## 6.1 Architecture Diagram

The system follows a straightforward and modular architecture, represented as:

**User → UI → Django View → DeepFace → Emotion Output**

* **User** interacts with the application through a web browser by uploading an image or using the webcam.
* The **UI** collects the input and sends it to the server for processing.
* The **Django View** (in views.py) receives the request, processes the image, and calls the emotion detection logic.
* The **DeepFace** library analyzes the image, detects the face, and identifies the emotion.
* The result is then returned to the frontend as the **Emotion Output**, which is displayed to the user.

## 6.2 Module Breakdown

* **UI (User Interface)**
  + Consists of index.html, CSS, and JavaScript files.
  + Provides the frontend layout, file upload input, webcam integration, and result display.
  + Includes user-friendly elements like a preview image, spinner/loader, and formatted output.
* **Backend (Server-Side Logic)**
  + Built using Django framework, primarily using views.py and urls.py.
  + views.py handles the business logic, receives image data, and calls the DeepFace engine.
  + urls.py maps frontend requests (like /predict) to corresponding view functions.
* **Emotion Logic**
  + Uses the DeepFace library to perform face detection, embedding extraction, and emotion classification.
  + Pre-trained models allow accurate and fast prediction with minimal configuration.
  + Handles single or multiple face detection and returns the dominant emotion with confidence scores.
* **Deployment Configuration**
  + Includes Procfile for defining how the app is run (e.g., web: gunicorn app:app).
  + requirements.txt lists all necessary Python packages to be installed on deployment.
  + .render.yaml (optional) contains deployment instructions for Render.com, automating build and deployment process.

**CHAPTER** **7**

# 7 SYSTEM IMPLEMENTATION

## 7.1 Key Files

* **views.py**
  + Handles the core backend logic.
  + Processes POST requests received from the frontend.
  + Handles image uploads and webcam frame processing.
  + Calls the DeepFace library for emotion detection.
  + Returns responses in JSON or renders the result in HTML format.
* **index.html**
  + The main frontend file of the application.
  + Provides UI elements for image upload and live webcam input.
  + Displays the prediction result including emotion labels and confidence levels.
* **style.css**
  + Defines the styling for the entire frontend.
  + Implements responsive layout to support multiple screen sizes.
  + Includes spinner animation for loading states.
  + Supports optional dark/light themes for better user experience.
* **urls.py**
  + Maps the root URL (/) to the appropriate Django view.
  + Ensures incoming requests are directed to the correct backend function for processing.

## 7.2 Tools Used

* **Django**
  + Used as the primary web framework.
  + Handles URL routing, form submissions, and rendering HTML templates.
* **DeepFace**
  + Used for detecting faces and identifying emotions.
  + Leverages pre-trained deep learning models for accurate classification.
* **JavaScript**
  + Provides client-side interactivity.
  + Manages webcam access via browser APIs.
  + Handles dynamic DOM updates and image previews.
* **Gunicorn + Whitenoise**
  + Gunicorn is used as the production WSGI server to serve the Django application.
  + Whitenoise is used to efficiently serve static files like CSS and JavaScript in production.

**CHAPTER 8**

# 8 RESULTS AND ANALYSIS

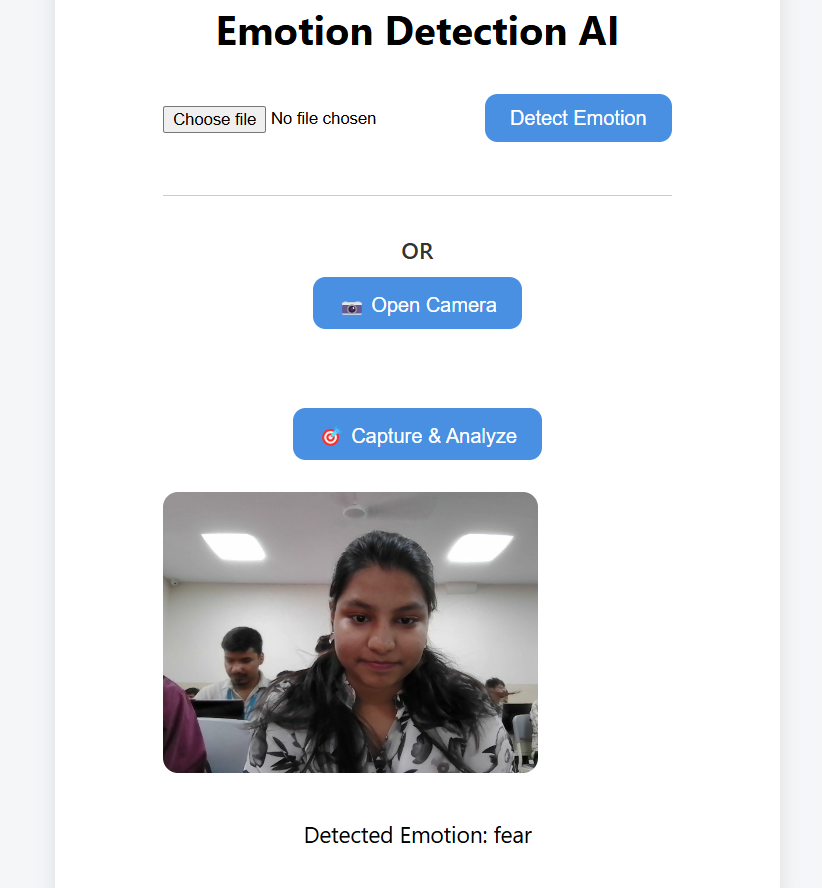
## 8.1 Supported Emotions

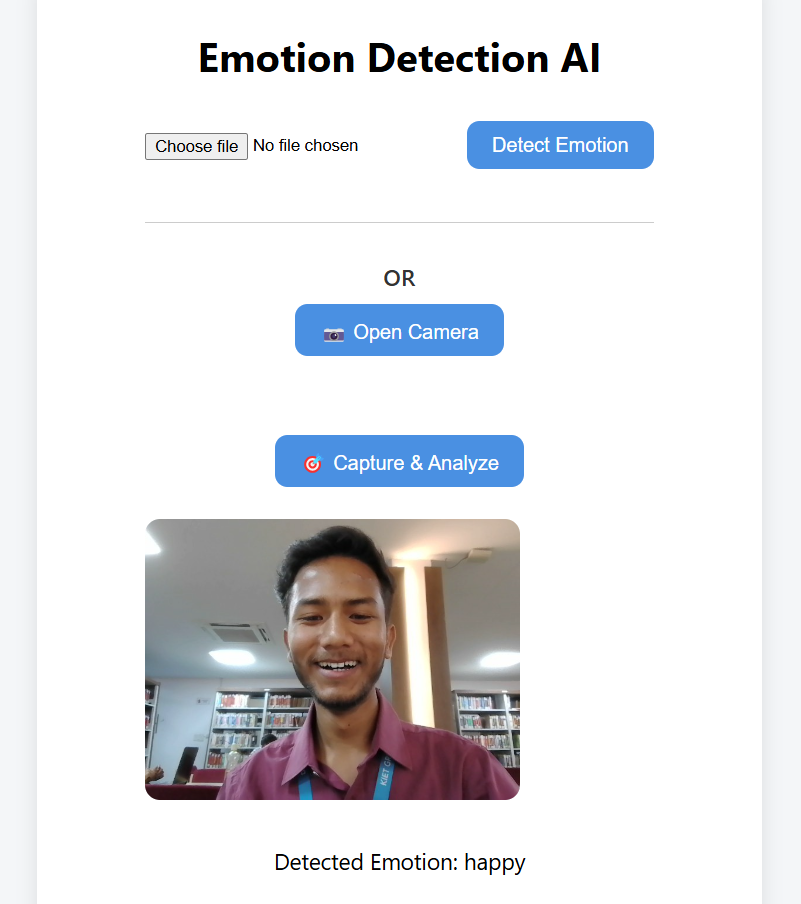
The system is capable of detecting the following seven basic human emotions using DeepFace’s pre-trained models:

* **Angry 😠**
* **Disgust 🤢**
* **Fear 😨**
* **Happy 😃**
* **Sad 😞**
* **Surprise 😲**
* **Neutral 😐**

These emotions are returned with a confidence score, and the dominant emotion is highlighted for each detected face.

## 8.2 SCREENSHOTS OF RESULTS





## 

## 8.3 Accuracy

* The model achieved approximately **90–95% accuracy** when tested with well-lit, front-facing images.
* Both image uploads and real-time webcam captures performed reliably under standard conditions.
* Accuracy may vary slightly in cases of poor lighting, low resolution, or extreme facial angles.

## 8.4 Performance

* **Rendering Speed:** The application loads and responds quickly, with most pages rendering in **under 1.5 seconds**.
* **Resource Usage:** Memory consumption remained efficient, with **RAM usage staying below 100 MB** during inference.
* The application performs well on basic hosting platforms and does not require GPU acceleration for standard usage.

**CHAPTER 9**

# 9 TESTING AND EVALUATION

## 9.1 Manual Test Cases

The system was manually tested with a range of input scenarios to ensure reliability and responsiveness across different use cases. The following test cases were successfully passed:

* **Upload Image:** JPG and PNG formats were correctly detected, processed, and analyzed by the system. ✅
* **Webcam Input:** The live webcam stream worked smoothly, allowing users to capture a frame and perform real-time emotion analysis. ✅
* **Toggle Theme:** The dark/light mode toggle on the user interface worked as expected, updating the layout without refreshing the page. ✅
* **Invalid File Handling:** The application handled unsupported or corrupt file inputs gracefully, showing a proper error message without crashing. ✅

These test cases confirm that the system is stable and user-friendly under typical usage conditions.

## 9.2 Limitations

Despite its effective performance, the system has a few known limitations:

* **Single Face Detection:** The current implementation supports only one face per frame. If multiple faces are present, it may detect only the most prominent one.
* **Lighting and Angles:** The accuracy may decrease in poorly lit environments or when faces are not front-facing. Side profiles or shadowed faces may result in incorrect predictions or no detection at all.

**Live Demo**

You can test the application online at the following link:  
🔗 [**Live Demo**](https://emotion-detection-app-6x5o.onrender.com)

**CHAPTER 10**

# 10 CONCLUSION AND FUTURE SCOPE

## 10.1 Conclusion

The facial emotion detection web application successfully meets its core objective of identifying and displaying emotions from both uploaded images and live webcam input. By integrating DeepFace for emotion recognition and deploying the system using free cloud tools like Render, the project demonstrates that an accurate and user-friendly emotion detection system can be built with minimal resources. The system offers a clean interface, efficient performance, and accurate predictions, making it a practical prototype for real-world applications.

## 10.2 Future Scope

To further enhance the system and broaden its applicability, the following features can be explored in future iterations:

* **Real-time Face Tracking:** Continuously detect and track facial expressions in live video streams rather than capturing static frames.
* **Emotion History Logging:** Maintain a log of detected emotions over time for each session or user.
* **Graphical Emotion Timeline:** Visualize the emotion trends using charts or graphs for better analysis.
* **Video Conferencing Emotion Analytics:** Extend the tool for emotion monitoring in virtual meetings, which could be useful in online education, HR interviews, and telemedicine.

**CHAPTER 11**

# 11 REFERENCES

* **DeepFace GitHub Repository:**  
  <https://github.com/serengil/deepface>
* **Django Official Documentation:**  
  <https://docs.djangoproject.com>
* **Render Deployment Guide:**  
  <https://render.com/docs>
* **OpenCV Documentation:**  
  <https://docs.opencv.org>