Academic Project: Robotic Interview Process System

ABSTRACT

The interview process in large organizations with numerous candidates is a time-consuming and labor-intensive task. Often, HR professionals must manage a large pool of shortlisted candidates, making it challenging to conduct one-on-one interviews efficiently. With the rise of digitalization and artificial intelligence (AI), process automation in recruitment is becoming increasingly relevant.

This project focuses on automating the recruitment process using AI-based facial recognition and machine learning techniques to analyze candidate confidence levels and emotional states during interviews. The system utilizes HAAR cascade algorithm for face recognition, Recurrent Neural Network (RNN) for classifying interview answers, and NLP-based text processing for analyzing responses.

Machine learning models are trained with datasets containing confident and fearful expressions, enabling AI to assess candidates' emotional states. The project integrates video acquisition, emotion detection, voice recognition, and NLP-based keyword mapping to ensure an accurate evaluation of candidates' performance. The system is built using Python with Tkinter for the frontend and MySQL as the backend.

This research highlights how AI-driven recruitment automation can streamline hiring processes, reduce HR workload, and improve the accuracy of candidate assessment.

MODULE DESCRIPTION

1. VIDEO ACQUISITION

- Video capturing via CCTV-based surveillance is implemented.
- Video frames are extracted for image-based analysis, reducing complexity.

2. FACE DETECTION

- Uses Haar Cascade algorithm, known for its high detection rate and fast processing speed.
- Steps include feature selection, evaluation, learning, and cascading classifiers.
- Haar features are extracted to identify facial structures and expressions.

3. FEATURE EXTRACTION

- Identifies key facial points required for emotion analysis.
- Extracts pixel-based features to detect emotions during interviews.

4. EMOTION DETECTION

- Employs deep learning, particularly Convolutional Neural Networks (CNN), for accurate emotion recognition.
- Analyzes candidates' facial expressions to determine confidence and stress levels.

5. TRAIN INTERVIEW QUERY AND KEYWORD

- AI is trained with company-specific questions and trends.
- Uses NLP to process and understand candidate responses.

6. VOICE RECOGNITION

- Converts spoken answers to text for further processing.

7. NLP PREPROCESSING

- Includes segmentation, tokenization, lemmatization, stop-word removal, and keyword mapping.
- Ensures efficient analysis of candidate responses.

8. OVERALL PERFORMANCE ANALYSIS

- Combines facial expressions, voice analysis, and textual responses to generate an overall assessment.
- Provides detailed feedback to improve candidates' confidence.

FRONTEND (Python - Tkinter Framework)

Introduction

Python is a high-level programming language known for its simplicity and flexibility. Tkinter is used to build a user-friendly graphical interface for the interview system.

Features of the Frontend:

- **User Interface:** Built with Tkinter for interactive and responsive UI.

- **Real-time Video Processing:** Displays video feed and processes facial expressions.
- **Live Emotion Detection:** Shows real-time confidence and stress levels.
- **AI-Powered Questioning System:** Generates interview questions dynamically.
- **Voice-to-Text Conversion:** Allows candidates to answer verbally, reducing typing effort.
- **Performance Dashboard:** Provides visual feedback on interview performance.
- **Report Generation:** Generates and exports candidate assessment reports.

Installation & Prerequisites

- **Operating System:** Windows/Linux/MacOS
- **Python Version:** 3.x
- **Required Libraries:** OpenCV, NumPy, TensorFlow, Tkinter, SpeechRecognition, Pandas, Scikit-learn
- **Setup:** Install Python and required dependencies, then run the Tkinter-based UI application.

BACKEND (MySQL)

Introduction

MySQL is a widely used open-source relational database management system (RDBMS) that ensures secure and efficient data handling. It stores candidate responses, face recognition results, and interview performance records.

Features of the Backend:

- **Candidate Database:** Stores interview details, responses, and performance reports.
- **Secure Data Management:** Ensures data integrity and quick retrieval.
- **Fast Query Processing:** Optimized SQL queries for real-time assessment.
- **Scalability:** Supports multiple concurrent interviews.

Installation & Configuration

- Install MySQL server and MySQL Workbench.
- Configure database schema for storing candidate responses.
- Use Python MySQL connector to link frontend with database.

CONCLUSION

This AI-based robotic interview system automates the recruitment process, reducing HR workload and improving accuracy. By leveraging machine learning techniques like facial recognition, NLP, and deep learning, the system provides an efficient and unbiased method of assessing candidates. The frontend built with Python (Tkinter) ensures ease of use, while MySQL guarantees robust data management. This project can be further enhanced with deep learning models for better accuracy and integration with cloud-based AI services.

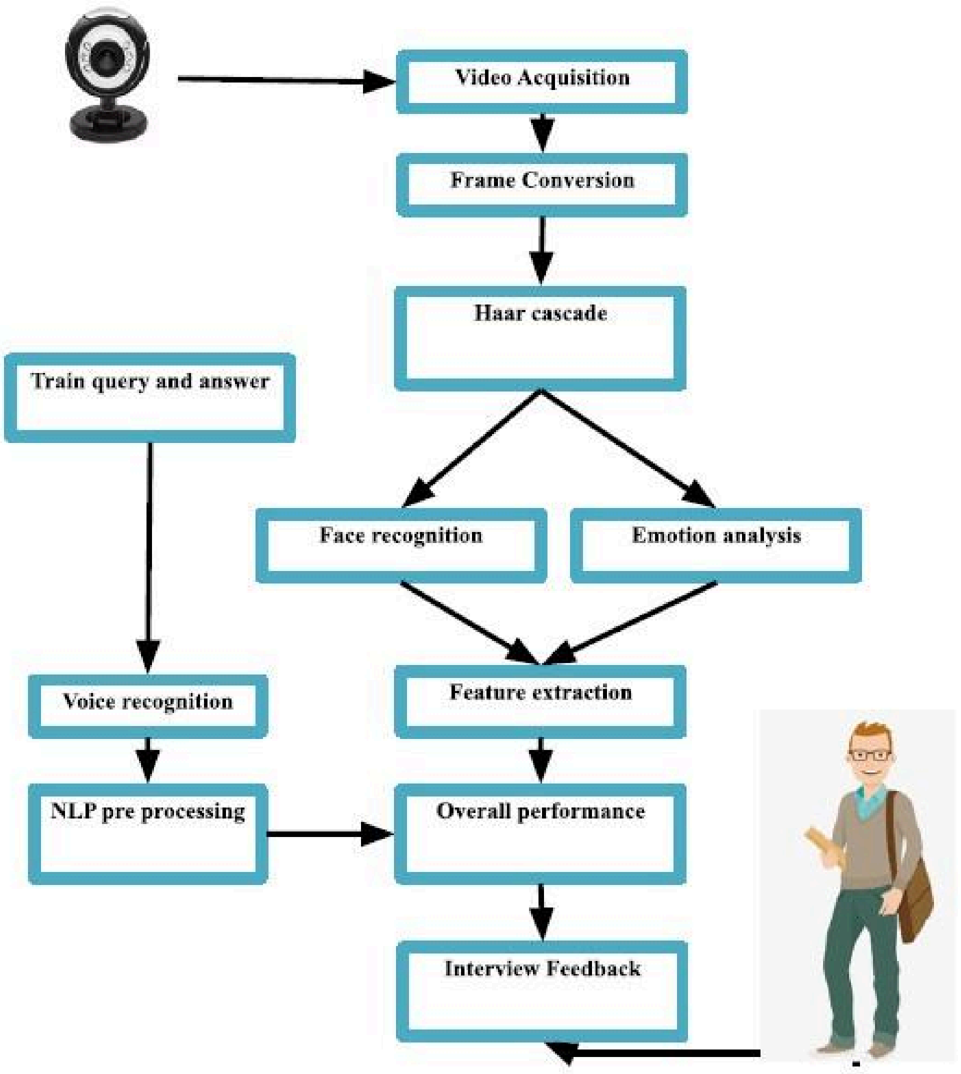
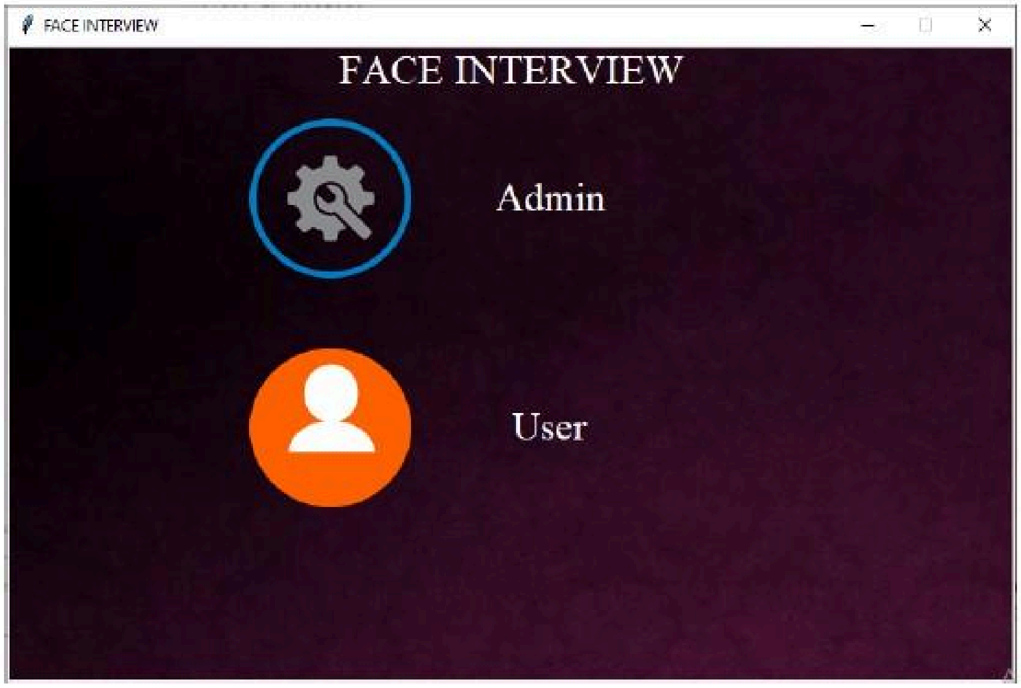
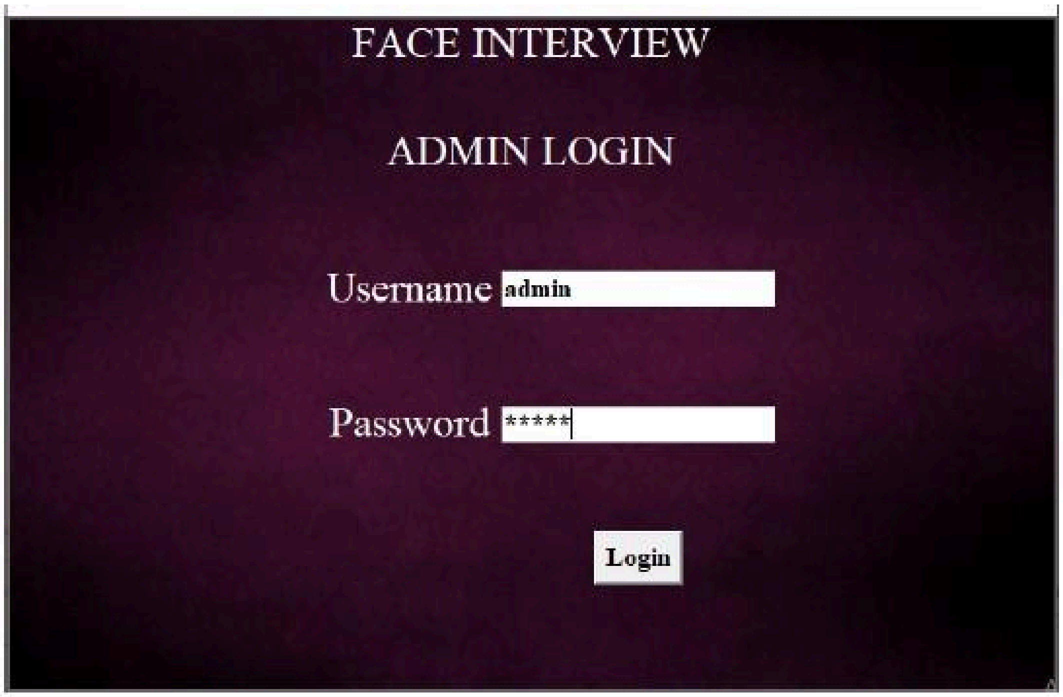


Fig 5.3.1 Architectural Diagram

OUTPUT SCREENS



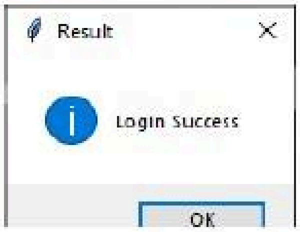
Home Page


FACE INTERVIEW

ADMIN LOGIN

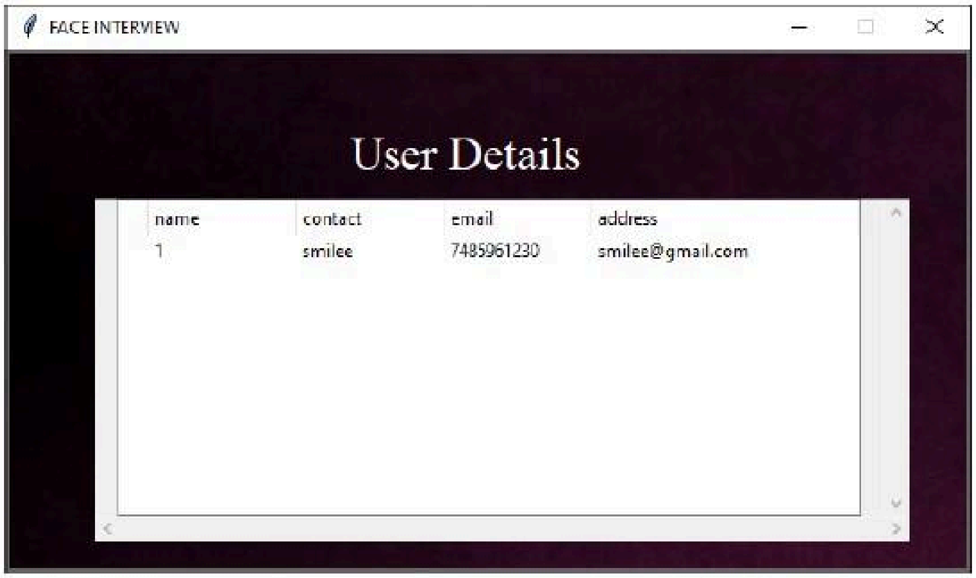
Username admin

Password *****


Result

Login Success

Admin login

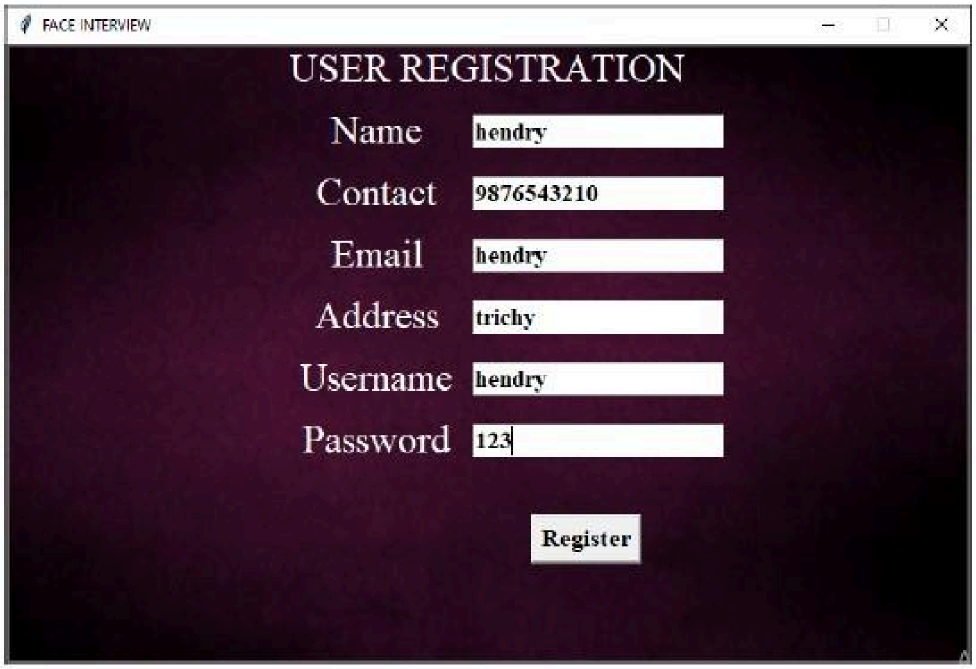


FACE INTERVIEW

User Details

name	contact	email	address
1	smilee	7485961230	smilee@gmail.com

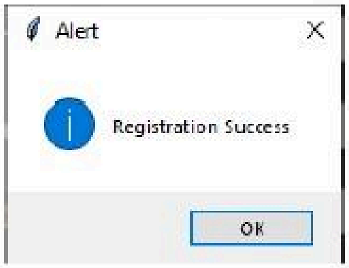
Company view candidate details



FACE INTERVIEW

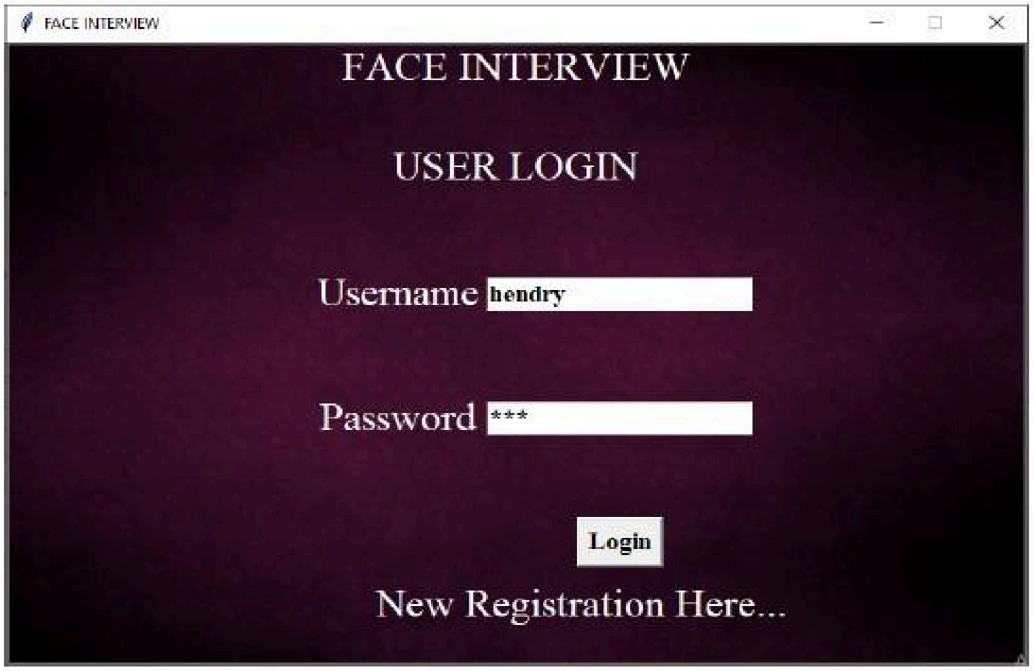
USER REGISTRATION

Name	hendry
Contact	9876543210
Email	hendry
Address	trichy
Username	hendry
Password	123


Alert

Registration Success

User Registration

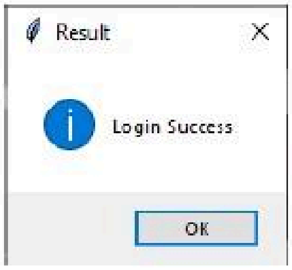

FACE INTERVIEW

FACE INTERVIEW

USER LOGIN

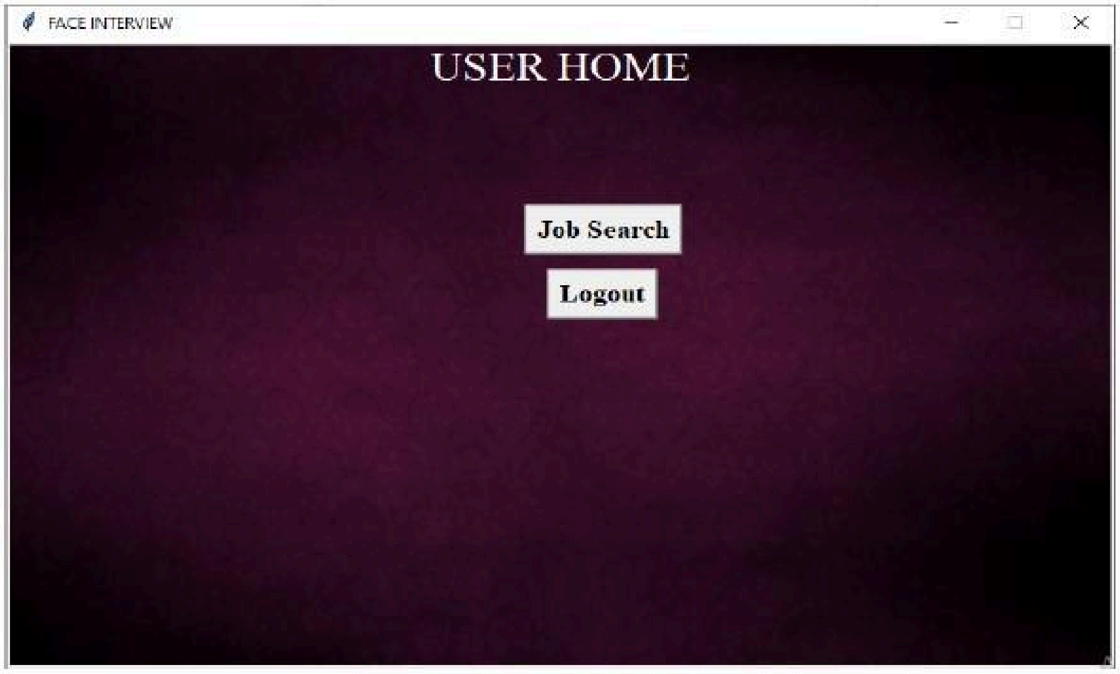
Username hendry

Password ***

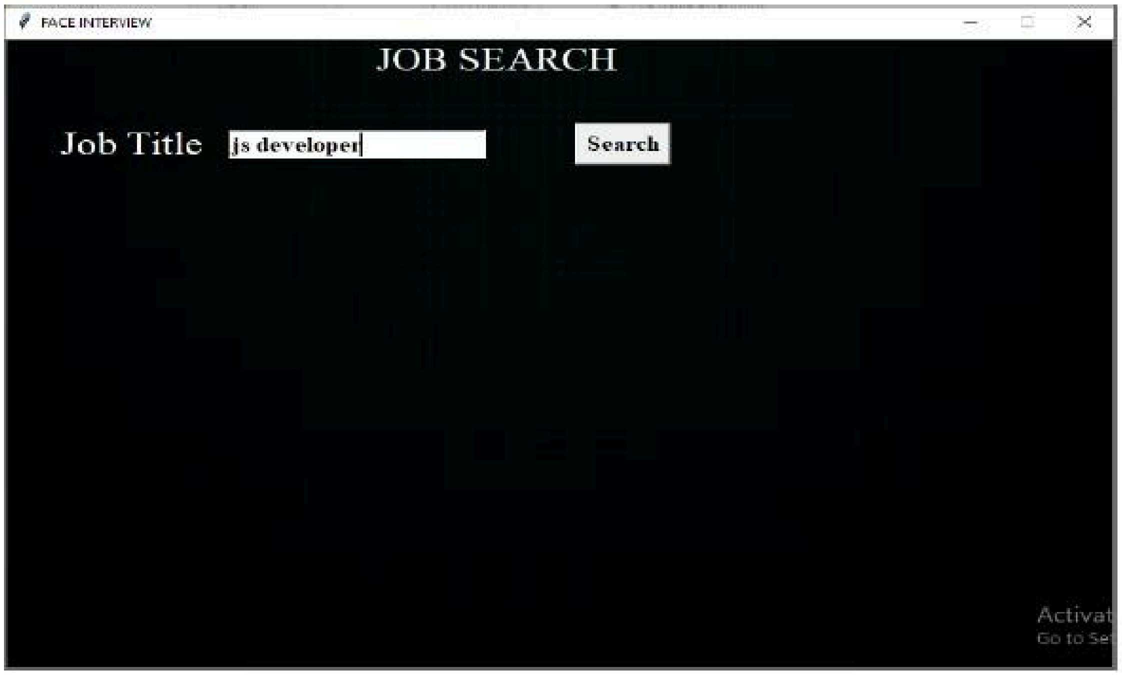

Result

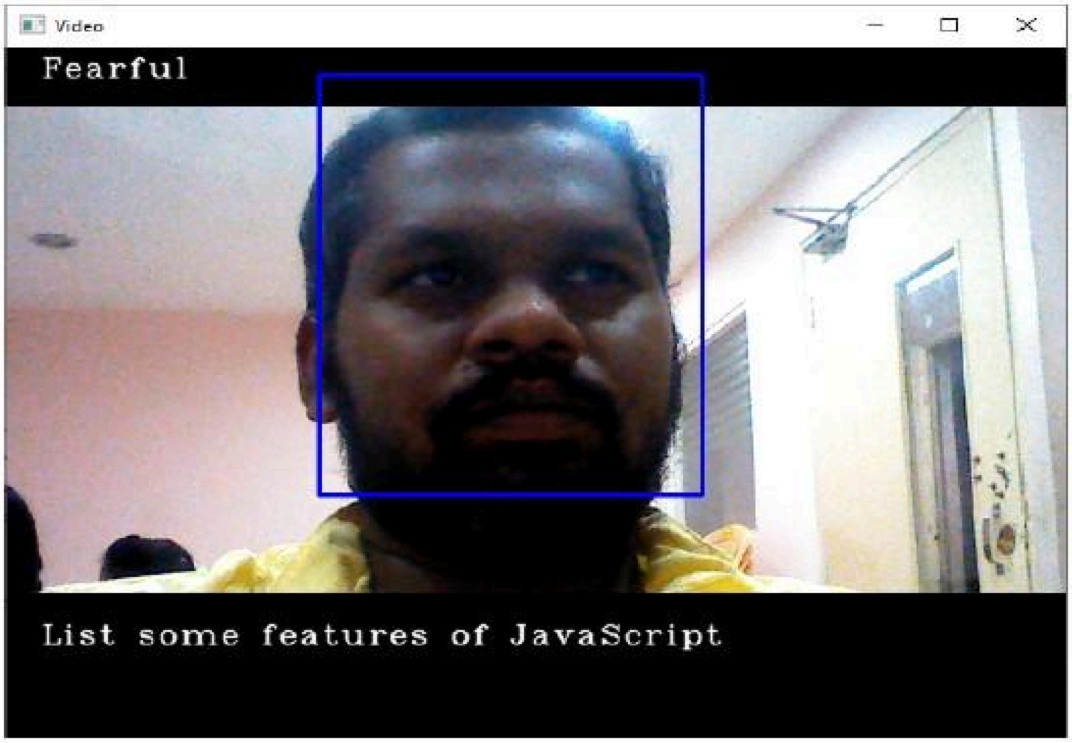
Login Success

User Login



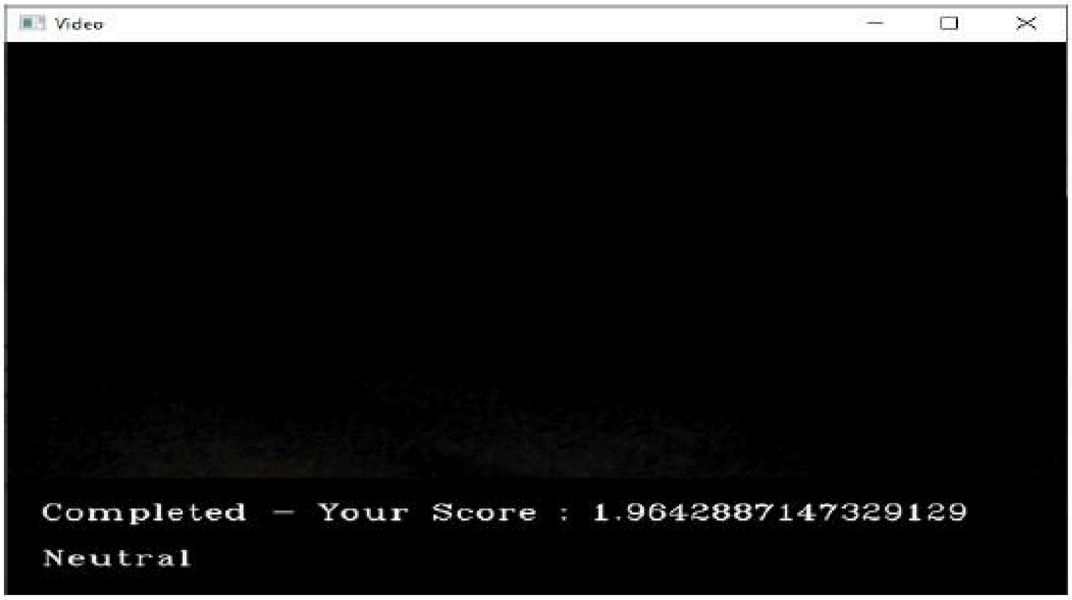
User home page





```
List some features of JavaScript
```

Emotion recognition with answering system



```
Completed - Your Score : 1.9642887147329129
Neutral
```

Score analysis

Mini Project:- Weather Application

Weather Application

Overview

The Weather Application is a Spring Boot-based REST API that fetches real-time weather data and forecasts for a given city using external APIs. It uses caching to optimize performance and reduce API calls.

Features

- Fetch current weather information for a specified city.
- Retrieve weather forecasts for multiple days.
- Caching to reduce redundant API calls.
- Centralized logging for monitoring application activity.

Technologies Used

- **Backend:** Spring Boot
- **Caching:** Spring Cache
- **HTTP Client:** RestTemplate
- **Logging:** Logback with SLF4J
- **APIs:**
  - OpenWeatherMap API (for current weather)
  - WeatherAPI (for forecasts)

Endpoints

1. Get Current Weather

- **URL:** `/api/weather`
- **Method:** `GET`
- **Query Parameters:**
  - `city` (String): Name of the city.

Example Request:
GET http://localhost:8080/api/weather?city=Thiruvarur

Example Response:
```
{
  "weather": "Sunny",
  "temperature": 32
}
```

2. Get Weather Forecast

- **URL:** `/api/forecast`
- **Method:** `GET`
- **Query Parameters:**
  - `city` (String): Name of the city.
  - `days` (int): Number of days to forecast.

Example Request:
GET http://localhost:8080/api/forecast?city=Chennai&days=3

Example Response:
```
[
  { "date": "2024-12-05", "avgTemp": 30, "maxTemp": 34, "minTemp": 27, "condition": "Cloudy" },
  ...
    { "date": "2024-12-08", "avgTemp": 28.05, "maxTemp": 32, "minTemp": 25, "condition":
"Cloudy" }
]
```

Setup and Installation

Prerequisites

- Java 17 or later
- Maven 3.6 or later

Steps

Clone the repository:
git clone https://github.com/your-repo/weather-app.git

1. cd weather-app
2. **Update API keys:**

   - Replace `API_KEY` in `WeatherService` with your OpenWeatherMap API key.
   - Replace `API_KEY1` in `WeatherService` with your WeatherAPI key.
3. **Build the application:**
   mvn clean install
4. **Run the application:**
   mvn spring-boot:run
5. **Access the application:**
   - Visit: `http://localhost:8080`

Logging

- Logs are stored in `src/weather-app.log`.
- **Log format:**
  yyyy-MM-dd HH:mm:ss [thread] LEVEL Logger - Message
- Logging configuration can be adjusted in `logback-spring.xml`.

Caching

- **Cache Names:**
  - `weatherCache` for current weather data.
  - `forecastCache` for weather forecast data.
- Cache hits and misses are logged.

Error Handling

Invalid city name or unavailable data:

```
{
  "error": "City not found. Please check the city name and try again."
}
```

General API or server errors:

```
{
  "error": "Failed to fetch weather data. Please try again later."
}
```

Monitoring

The application tracks:

- Total API requests.
- Cache hits and misses.

Future Improvements

- Integrate a frontend for visualization.
- Add support for more weather parameters.
- Expand caching mechanisms.