**Infosys Internship 4.0 Project Documentation**

**Project Documentation: College – Automated Attendance**

**Introduction**

This project aims to automate the attendance marking process using facial recognition technology. The system captures images of students in a classroom, identifies them, and marks their attendance in a database. The project leverages Python, PostgreSQL, and various machine learning libraries. The team members involved are:

* **RAGHURAJ PRATAP YADAV**

The project focuses on automating attendance marking for educational institutions. It includes:

* Capturing images using a camera
* Identifying students using facial recognition
* Storing attendance records in a database
* Generating and sending attendance reports

**Exclusions**:

* Integrating with other existing attendance systems
* Real-time attendance monitoring
* Extensive testing in varied lighting conditions

**Requirements**

**Functional Requirements:**

* Capture and process images to identify students.
* Store attendance records in a PostgreSQL database.
* Generate daily, weekly, and monthly attendance reports.
* Send reports to faculty and director via email.

**Non-Functional Requirements:**

* The system should be able to handle images with multiple faces.
* The system should be robust against variations in lighting and image quality.
* The system should ensure data security and privacy.

**User Stories:**

* As a teacher, I want to receive daily attendance reports via email.
* As an administrator, I want to be able to view and download attendance records.

**Technical Stack**

* **Programming Languages**: Python
* **Frameworks/Libraries**: OpenCV, DeepFace, Pandas, NumPy
* **Databases**: PostgreSQL
* **Tools/Platforms**: Email (SMTP), Schedule, SQLAlchemy

**Architecture/Design**

**System Architecture:**

* **Image Capture**: Captures images at scheduled intervals.
* **Face Recognition**: Uses DeepFace to identify students.
* **Database**: Stores attendance records.
* **Report Generation**: Generates and sends reports via email.

**Design Decisions:**

* **Model**: Used VGG-Face for face recognition due to its high accuracy.
* **Database**: Chose PostgreSQL for its robustness and ease of integration with Python.
* **Email Service**: Utilized SMTP with Office365 for sending emails.

**Development**

**Technologies and Frameworks:**

* Python for scripting and development.
* OpenCV for image processing.
* DeepFace for facial recognition.
* Pandas and NumPy for data manipulation.
* SQLAlchemy for database operations.
* Schedule for task scheduling.
* SMTP for email notifications.

**Coding Standards and Best Practices:**

* Followed PEP8 for Python coding standards.
* Modularized the code for maintainability.
* Implemented error handling and logging.

**Challenges:**

* Handling images with multiple faces: Addressed by adjusting the face detection thresholds.
* Ensuring accurate face recognition: Improved by training on a larger dataset of student images.

**Testing**

**Testing Approach:**

* **Unit Tests**: Tested individual functions for face detection and recognition.
* **Integration Tests**: Tested the integration between image capture, recognition, and database storage.
* **System Tests**: End-to-end testing of the entire workflow from image capture to report generation.

**Results:**

* Successfully identified and marked attendance for over 90% of the test images.
* Detected and resolved minor bugs related to image processing and database connectivity.

**Deployment**

**Deployment Process:**

* Set up the PostgreSQL database on a local server.
* Configured the Python environment with necessary libraries.
* Scheduled the image capture and processing script using the Schedule library.
* Automated the report generation and email sending process.

**Deployment Instructions:**

1. Install PostgreSQL and create the necessary database and tables.
2. Set up the Python environment and install required libraries.
3. Configure the email settings in config.py.
4. Run the scheduling script to start the daily, weekly, and monthly tasks.

**User Guide**

**Using the Application:**

1. Place the camera in the classroom to capture images.
2. Ensure the image file names follow the HH-MM-SS.jpg format.
3. The system will automatically process the images and mark attendance.
4. Reports will be generated and sent to the configured email addresses.

**Troubleshooting Tips:**

* If the system fails to recognize faces, check the image quality and lighting conditions.
* Ensure the database is running and accessible.
* Check the email configuration if reports are not being sent.

**Conclusion**

The Auto Attendance System successfully automates the attendance marking process, providing a reliable and efficient solution for educational institutions. Key achievements include high accuracy in face recognition and seamless integration with email services for report distribution.

**Lessons Learned**:

* Importance of high-quality images for accurate face recognition.
* Need for thorough testing in varied environments to ensure robustness.
* The value of modular code design for ease of maintenance and scalability.

**Future Improvements**:

* Integration with real-time attendance monitoring systems.
* Enhancement of face recognition accuracy under different lighting conditions.
* Expansion of the system to support multiple classes and subjects.

**Appendices**

* DAILY REPORT:

A screenshot of a table

Description automatically generated

* MONTHLY REPORT : A screenshot of a spreadsheet

  Description automatically generated
* SQL CONNECTION :   
  A black screen with colorful lines

  Description automatically generated
* DAILY REPORT GENERATOR :  
  A computer screen shot of a program code

  Description automatically generated
* EMBEDDING GENERATOR :  
  A screen shot of a computer program

  Description automatically generated
* STUDENT IDENTIFIER:  
  A black screen with many text

  Description automatically generated with medium confidence
* FLOWCHART :   
  A diagram of a flowchart

  Description automatically generated