CLOUD APPLICATION DEVELOPMENT

Title:

Cloud Based Attendance System Using Facial Recognition in Python

Presented By:
Kumar Dev 500083164

Guided By:
Dr. Harvinder Singh
Assistant Professor
Department of Systemics
School Of Computer Science

1. Introduction

- Facial recognition-based cloud-based attendance solutions are gaining popularity as a safer and more effective approach to maintain attendance data in businesses. These systems utilise machine learning algorithms to recognise and link people's faces to the relevant information in a cloud-based database. Python has a strong collection of tools and frameworks for computer vision and machine learning, making it a popular programming language for developing facial recognition systems. These systems may scale to handle a large number of users and offer administrators real-time attendance tracking and reporting capabilities by utilising the power of cloud computing. Overall, a cloud-based attendance system employing Python and facial recognition can increase security, decrease workload, and improve accuracy while handling attendance records.

2. Problem Statement

- A cloud-based attendance system using facial recognition seeks to preserve accurate real-time attendance data while boosting productivity and decreasing errors. Facial recognition is required for user authentication, system interaction, and data security. Scalable, user-friendly, and remote access should all be features of the solution.

3.Project Proposal:

- **Objective:** The Objectives of a cloud-based attendance system using Python facial recognition are to automate and simplify attendance tracking, improve accuracy and eliminate errors, offer customization and anomaly detection, provide real-time tracking and reporting, increase productivity, and enhance workplace security.
- **Target Users:** The target audience for a cloud-based attendance system using facial recognition in Python is organisations and enterprises of all sizes and in all industries that want a trustworthy and efficient way to track attendance. Institutions including universities, hospitals, offices of the government, factories, and more may fall under this category. The system allows managers, supervisors, and HR staff to keep track of attendance and generate reports. It is a convenient and easy approach to keep track of attendance because employees may use the system to check in and out.

- **Scope:** The project entails the creation of a facial recognition-based cloud-based attendance system in Python. Facial recognition technology will be used by the system to allow users to check in and leave, providing real-time and precise attendance data. The system will leverage cloud-based databases for data storage and remote access in addition to machine learning algorithms for facial recognition. To manage large volumes of attendance data while ensuring data security and privacy, the system must be scalable. To make sure that only authorized staff can access attendance data, it will also incorporate user authentication and access control.

- **Technologies:** The following technologies will be used to develop the application:
  1. Front end and Back-end: Tkinter(Used for GUI), Python
  2. Packages: NumPy, OpenCV-python, face_recognition, DateTime, dlib
  3. Cloud Storage: Amazon S3
  4. Storage: CSV file, Database
  5. Version Control: Github

Project Timeline:

- Phase 1: Planning and Design (2 weeks)
- Phase 2: Development (3 weeks)
- Phase 3: Testing and Deployment (3 weeks)

Phase 1: Planning and Design (2 weeks)

- Identify project scope and objectives
- Define project requirements and specifications
- Develop a detailed project plan and timeline
- Assign roles and responsibilities to team members
- Identify and select the technologies and tools to be used in the project (using Python)
- Design the database schema and encryption methods

Phase 2: Development (3 weeks)

- The real coding of the attendance system is done during this phase. The developer would write the code necessary to put the design phase's defined features and functionalities into action during this phase. Additionally, they would use Python libraries like OpenCV and dlib to incorporate the face recognition functionality into the system.

Phase 3: Testing and Deployment (3 weeks)

- **Testing:** In order to guarantee the correct functionality, compatibility, and performance of the cloud-based attendance system utilising Python face recognition, testing is an essential step. To find and fix any faults or errors in the system, many forms of testing should be carried out, including unit testing, integration testing, system testing, acceptability testing, and performance testing. This will guarantee that the system operates as anticipated and satisfies end-user needs.
- **Deployment:** A robust and dependable cloud-based attendance system employing Python facial recognition requires deployment, which includes setting up the necessary infrastructure, deploying code to a production environment, monitoring/logging, creating a disaster recovery plan, doing user acceptability testing, and following the right testing procedures.

Milestones:

- Milestone 1 : Completed project planning and design(2 weeks)
- Milestone 2: Setting Up Environment for the Project by Installing Required Packages and Implemented GUI Using Tkinter (3 weeks)
- Milestone 3: Setting Up a database or CSV file and Saving a Predefined Images for checking Attendance and saving it into a file.(4 weeks)
- Milestone 4: Testing and debugging completed, application deployed (4 weeks)

Roles and Responsibilities:

- Project Manager: Overall responsibility for the project, including project planning, timeline management, and team coordination.
- Developer: The facial recognition model is designed and developed, integrated with the cloud infrastructure and other systems, and then the application is deployed as part of the developer role for a cloud-based attendance system that uses facial recognition.
- Tester: Responsible for conducting testing and debugging of the application.

Functional Requirements:

User Roles:

- **Admin:** Absolute control over accounts, settings, and analytics.
- **User:** Limited access to examine own records, clock in and out, and request time off.

User Interface:

- **Login page:** Users can be authenticated using their credentials or face recognition on the login page of a cloud-based attendance system.
- **Dashboard:** Users can check their attendance history, clock in and out, and request time off.

- **Account Settings:** Users should be able to update their profile information and Reset their password.
- **Admin Dashboard:** Admins should have access to an admin dashboard where they can manage users and their Account Settings.

System Requirements:

- A Python web application-compatible server or cloud platform.
- Sufficient computation and memory to manage database operations and facial recognition algorithms.
- An SSL certificate to enable safe communication between the server and client devices, as well as a high-speed internet connection to ensure quick and dependable access to the system.
- A database management system to store and manage user information and attendance records, such as MySQL or PostgreSQL
- A Python web framework to create the web application and manage user interactions, like Django or Flask
- To find and match faces, facial recognition libraries (like OpenCV or dlib) can be used.
- A sufficient amount of storage for user data, attendance data, and system logs

Functional Requirements:

- **User Authentication:** Users should be verified by the system using their credentials or face recognition.
- **Attendance Recording:** Based on the user's face recognition information and timestamps, the system should reliably record attendance.
- **Clock In/Out**: Users should be able to clock in and out of work using face recognition or a web-based interface thanks to the technology.
- **Reporting:** The system should produce reports on attendance data and user information that administrators and users can access.
- **Access Control:** Based on the user's job and permissions, the system ought to limit access to sensitive information and functionality.
- **Scalability:** The system should be able to scale up or down as necessary to meet an increasing number of users and attendance records.

- **Security:** The system should be protected by safeguards that guard user information and stop unauthorised access or breaches.
  Scalability: The system should be able to scale up or down as necessary to meet an increasing number of users and attendance records.

Technical Design Document

Monolithic Architecture:

- With all components closely linked and running on a single server or cluster of servers, the monolithic architecture for the attendance system entails creating the complete application as a single, self-contained entity. In this architecture, the user interface, database, and other components are all deployed as a single unit from the same codebase.

- We developed the application logic for our attendance system using a Python web framework, such as Django or Flask, and we integrated facial recognition tools, such as OpenCV or dlib. The system is set up on a cloud-based server with scalable storage provided by Google Cloud Storage or Amazon S3.

Cloud Infrastructure:

- Cloud Platform: The application will be deployed on a cloud platform such as AWS or Azure.
- Load Balancer: Distributes traffic across multiple instances of the microservices to ensure high availability.
- Virtual Machines: Instances of the microservices will be deployed as virtual machines to allow for scaling and flexibility.
- Object Storage: Encrypted files will be stored on a cloud storage platform such as AWS S3 or Azure Blob Storage.

Test Plan and Test Cases

Testing Approach:

- Unit Testing: Test each microservice individually.
- Integration Testing: Test all microservices together.
- System Testing: Test the entire application end-to-end.

Test Cases:

The accuracy of the face recognition algorithm, the consistency and accuracy of facial image capture, the system's capacity to securely store and retrieve attendance data, the system's capacity to handle multiple requests, the generation of attendance reports, the testing of security features, and the system's scalability under heavier loads are possible test cases for a cloud-based attendance system using face recognition in Python. The actual test cases will be determined by the system's requirements and the facial recognition software being employed.

Expected Results:

- Accurate and consistent facial image capture: Facial images that are clear and useful under various circumstances.
- Accurate Face Recognition: Recognising registered users based on their facial characteristics.
- Identification of registered and unregistered faces
- Concurrent user requests handling: The ability to handle several requests from various users at once without performance degrading.
- Accurate Attendance Reports: creation of precise attendance reports for pupils or workers.
- Security: The application of security measures like access controls, encryption, and authentication procedures.

Maintenance and Support Plan:

Bug Fixing:

- Bug reports will be made via a support ticket system.
- Each problem will be given a priority level based on its severity.
- Depending on their importance, bugs will be rectified promptly.

Feature Updates:

- The feasibility and impact of feature requests on the application will be investigated and evaluated.

- Future updates will incorporate and publish any feasible and useful feature requests.

Technical Support:

- Technical support will be provided through a support ticket system.
- Response times will be based on the severity of the issue.
- Technical support will be available during business hours

1. **Security Updates:**

- Regular security updates will be implemented to ensure the application is secure.
- Any identified security vulnerabilities will be addressed and fixed in a timely manner.

2. **Backup and Disaster Recovery:**

- Regular backups of the application and its data will be taken to ensure that data can be restored in the event of a disaster.
- Disaster recovery procedures will be in place to ensure that the application can be quickly restored in the event of a disaster.

Documentation:

- All application-related documentation shall be kept current and available to support staff.
- Changes to the application and its supporting materials will be promptly informed to the necessary parties.

THANK YOU