CLOUD APPLICATION DEVELOPMENT

Title: Cloud based Attendance System Using Facial Recognition in Python

Presented by: Kumar Dev (500083164)
Guided by: Dr. Harvinder Singh

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.

Problem Statement:

The goal of a facial recognition cloud-based attendance system is to create a solution that accurately maintains attendance data in real-time for a company or institution while reducing errors and boosting productivity. The system shall make use of a cloud-based platform to enable remote access and management of attendance data and make use of facial recognition technology to rapidly and reliably identify and authenticate users. The solution must interact with existing systems, be scalable to handle massive volumes of attendance data, and guarantee data security and privacy. The solution should also be user-friendly, making it simple for administrators and end users to interact with the system and explore it.

Task Dependencies: The project has various task dependencies, including:

1. Define the parameters and scope of the project.
2. For the application's hosting, do some research and select a cloud platform (such as AWS, Azure, or GCP).
3. Create a virtual machine or container instance in the cloud platform and configure it to run the application.
4. Install and set up OpenCV, Dlib, and Face Recognition, as well as other applications and libraries required for Python facial recognition.
5. Create and test facial recognition algorithms to identify faces in photos or video streams that were acquired by a webcam or other camera device.
6. Integrate the facial recognition algorithms with a database that contains user profiles, schedules, and other pertinent data in addition to attendance data.
7. Create and construct a user interface for the attendance system that enables users to view attendance reports and manage their accounts, such as a web application or mobile app.
8. To guarantee that the system is reliable and satisfies user needs, do extensive testing and quality assurance.
9. Install the system on the cloud platform, then carry out any regular upkeep and updates that are required.

Chosen Task-based Application Model: The chosen application model for this project is Monolithic Application Model Where the entire programme is created as a single, seamless entity. The user interface, database, and all other elements are seamlessly integrated into a single application, including the facial recognition algorithms. For smaller applications with straightforward functionality and a low degree of complexity, this paradigm is appropriate.

Application Development Process: Here are the Application Development Process for Cloud based Attendance System using Facial Recognition in Python:

1. **Requirement Gathering:** In order to comprehend the features, functionality, and scope of the attendance system, the project team is gathering requirements from stakeholders at this stage. This entails determining the users, their functions, and the needs for data accessibility, security, and storage.
2. **Design And Architecture:** The system's architecture and components are defined using the requirements acquired in the earlier step. In this phase, the user interface and facial recognition algorithms are developed together with the database schema for the system.
3. **Development:** The development team begins creating the system's components as soon as the design and architecture are complete. The system's components are constructed in this stage using the selected task-based application model, which includes creating facial recognition algorithms and integrating them with the rest of the system.
4. **Testing:** The system is tested when development is finished to make sure it complies with specifications and

works as intended. This involves evaluating the user interface, database, and facial recognition algorithms. Before deployment, this stage's objective is to locate and address any defects or problems.

5. **Deployment:** Following testing, the system is set up on a cloud computing platform like AWS, Azure, or GCP. This entails putting in place the required infrastructure and setting up the system for use in a production environment.
6. **Maintenance:** The system is maintained after deployment to make sure it keeps working properly. This entails routine testing, problem fixes, and updates to make sure the system remains current and safe.

Implementation of Task Model: Python's cloud-based attendance system with facial recognition was built utilising the task-based architecture because it enables modular development and upkeep. Updates can be made without affecting the system as a whole, and each task or function can be designed and tested separately. This method offers an organised manner to make that all essential features are present and functioning properly. The task-based architecture is adaptable and effective, making it the best choice for creating a complicated system like a cloud-based attendance system using Python facial recognition.

Integration with Web Services: APIs are required for connecting web-based services with a facial recognition-based cloud-based attendance system. They serve as an intermediate layer between various systems and offer a standardised interface for data transmission. To allow the attendance system to accept and send data to web services, developers must create an API module. They must also make sure that the API is adequately documented and secure to allow other systems to communicate with it.

Development of Parameter Sweep Applications: Developers create a cloud-based attendance system using Python scripting, define the parameters they wish to test, and then automate the testing procedure. While cloud-based services like AWS can run numerous virtual machines simultaneously to shorten testing time, the script changes parameter settings and logs outcomes. The outcomes can be analysed and visualised using data analytics tools like pandas and matplotlib to pinpoint the optimal parameter values for optimum performance.

Test and Deploy: To guarantee that it complied with the client's needs, the application underwent extensive testing. Both human and automated testing were done during the testing procedure. Elastic Beanstalk was used to deploy the application on AWS, making scaling and deployment simple.

Understanding requirements, building the architecture, making the components, testing, deploying on the cloud, and managing the system are all steps in the development of a cloud-based attendance system employing facial recognition in Python. Stakeholder cooperation is essential to the iterative process.

In conclusion, Python's cloud-based attendance system with facial recognition is a creative and effective way to track attendance. It offers simple accessibility, scalability, and affordability because to the usage of cutting-edge technologies like cloud computing and computer vision. An organised and effective strategy for system development and optimisation is provided by the task-based architecture, integration with web services, and parameter sweep apps. A CI/CD pipeline's implementation ensures complete testing and validation prior to deployment, lowering the possibility of system defects and failures. Overall, the technology offers businesses an automated and dependable method for tracking attendance.