

Facial Recognition System on AWS (Serverless AI Architecture)

Facial Recognition System

Programme / Academy: Cloud Computing/
Cloudboosta

April 2026



Executive Summary

- Built a **fully serverless facial recognition system on AWS**
- Combines:
 - AI (Rekognition)
 - Serverless compute (Lambda)
 - API-driven architecture
- Achieves:
 - Real-time authentication (<1 sec)
 - High scalability
 - Secure identity verification



Business Case

- Organisation: **SecureVision Technologies Ltd**
- **Business Drivers:**
 - Improve security (eliminate impersonation)
 - Reduce operational cost (no infrastructure)
 - Enable real-time identity verification
 - Scale across multiple locations
- **Outcome:**
 - Pay-per-use model
 - Fully automated system
 - Enterprise-ready deployment



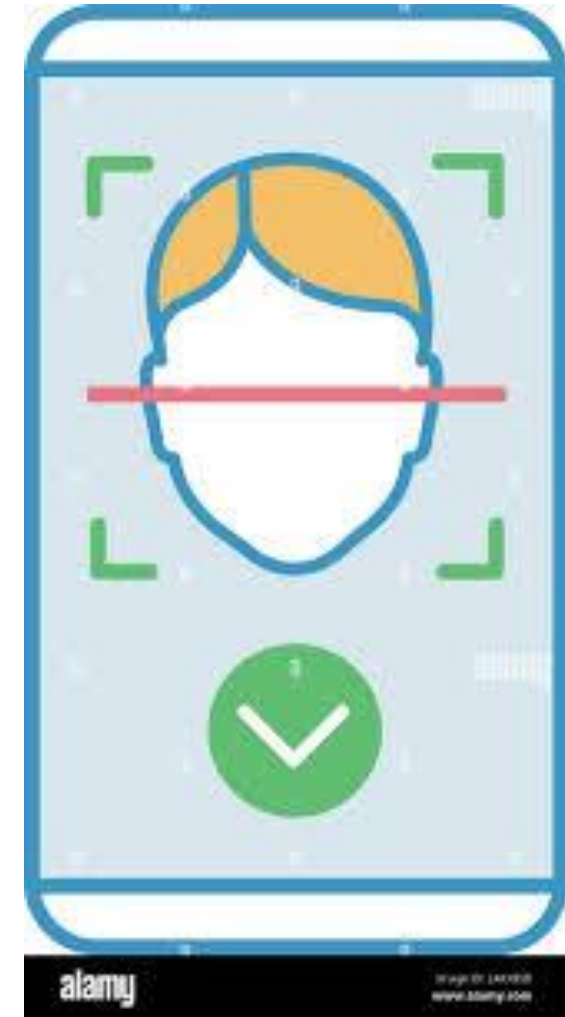
Problem Statement

- Traditional systems suffer from:
 - Identity theft (cards, PINs)
 - Poor scalability
 - Manual verification delays
 - High operational cost
- **Goal:**
Design a **secure, scalable, automated identity system** using **AI + Cloud**



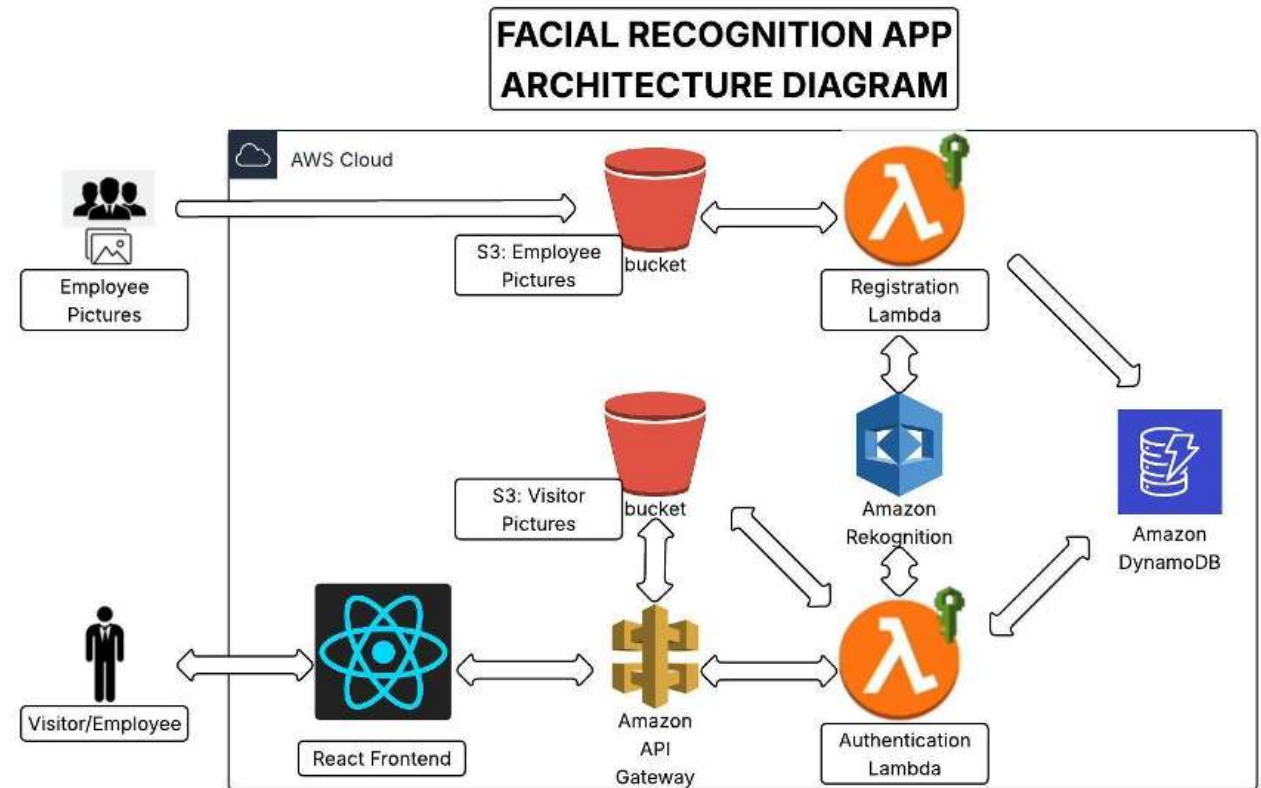
Project Objectives

- Build a **serverless facial recognition system on AWS**
- Implement **registration using S3 and Lambda**
- Enable **authentication via API Gateway and Lambda**
- Integrate **Rekognition for face matching**
- Store data in **DynamoDB**
- Apply **IAM-based security and CloudWatch monitoring**
- **Outcome:**
Scalable and secure real-time identity system



Solution Overview

- Cloud-native facial recognition system
- Serverless + AI-powered
- Hybrid architecture:
 - Event-driven (registration)
 - API-driven (authentication)
- **Core Services:**
 - CloudFront, S3, Lambda, Rekognition, DynamoDB, API Gateway

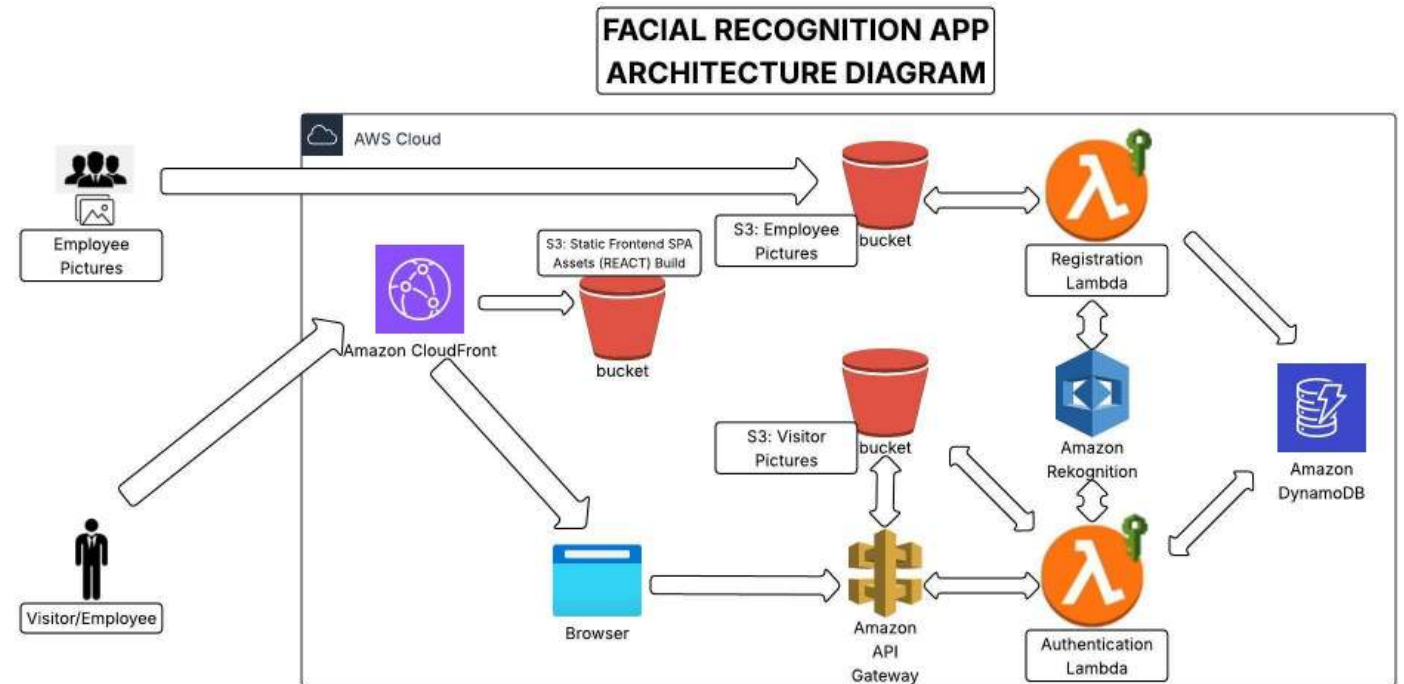


Architecture Overview



- **Layers:**

- CloudFront
- Frontend (React)
- API Gateway
- Lambda (Compute)
- S3 (Storage)
- Rekognition (AI)
- DynamoDB (Database)
- IAM (Security)
- CloudWatch (Monitoring)



Project Artifacts & Demo

- Architecture diagram
 - https://drive.google.com/file/d/1E6xGUJlrZjnUrh0M8aAP4ml746lfN-D6/view?usp=drive_link
 - https://drive.google.com/file/d/1pzApoyZF39UllTK2ml1Dixmj8ncdTJfU/view?usp=drive_link
- Video recap of executed project
 - https://drive.google.com/file/d/1SutFomJvondX73Sy9ECcivWe6W4M0c8R/view?usp=drive_link



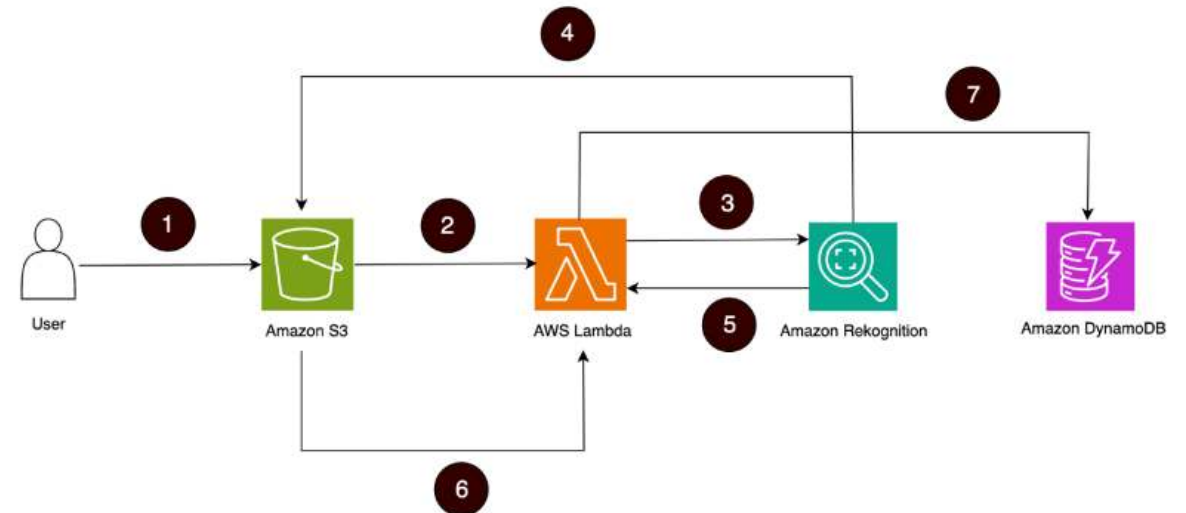
Architecture Style

- Fully serverless
- Event-driven + API-driven hybrid
- Decoupled microservices
- Stateless compute
- **Result:**
 - No infrastructure management
 - High availability
 - Elastic scaling



Registration Workflow (Event-Driven)

- User uploads image
- Stored in S3
- S3 triggers Lambda
- Rekognition indexes face
- Face ID stored in DynamoDB
- **Key Insight:** Fully automated pipeline



Authentication Workflow (API-Driven)

- User uploads or captures image via browser (CloudFront-served app)
- Browser sends request to API Gateway
- Lambda processes image
- Rekognition compares face
- DynamoDB retrieves identity
- **Output:**
 - Match → Access granted
 - No match → Access denied



Technology Stack

AWS Lambda
(Python 3.14)

Amazon
Rekognition (AI
engine)

Amazon S3
(storage)

DynamoDB
(NoSQL DB)

API Gateway
(REST API)

CloudFront-
delivered (React
Frontend)

CloudWatch
(monitoring)

IAM (security)

Key Design Principles

- Decoupling (independent services)
- Stateless compute (Lambda)
- Managed services (no servers)
- Event-driven automation
- **Impact:**
 - Scalability
 - Fault isolation
 - Simplicity



Security Architecture

- IAM role-based access control
- Least privilege principle
- S3 public access blocked
- API secured with keys
- Temporary credentials (STS)
- **Result:** Secure-by-design system



Performance & Scalability

- Lambda auto-scales
- DynamoDB on-demand scaling
- S3 unlimited storage
- Sub-second response time
- **Outcome:**
 - Handles high concurrency
 - Zero infrastructure limits



Observability & Monitoring

- CloudWatch logs:
 - Execution time
 - Errors
 - Memory usage
- Enables:
 - Debugging
 - Performance tuning
 - System auditing

Challenges Likely to be Encountered

- Image quality affecting recognition
- False positives/negatives
- IAM permission complexity
- API integration (CORS issues)
- Lambda cold starts

How Challenges Were Overcome

- Applied IAM roles for controlled access
- Enabled CORS in API Gateway
- Used CloudWatch for debugging
- Designed stateless architecture to reduce failure impact
- **Result:** Stable and production-ready system



Lessons Learned

- Serverless requires strong design discipline
- Event-driven architecture simplifies automation
- AI works well but needs calibration (accuracy, thresholds)
- Observability (CloudWatch) is essential for debugging
- Clean frontend–backend separation is critical
- IAM is central to system security
- Architecture thinking matters more than coding



AWS Well-Architected Pillars Applied



1. Operational Excellence

CloudWatch monitoring
Structured logging



2. Security

IAM roles
S3 access control
API security



3. Reliability

Serverless architecture
No single point of failure

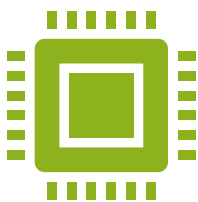
AWS Well-Architected Pillars Applied



4. Performance Efficiency

Lambda auto-scaling

Rekognition optimised AI



5. Cost Optimisation

Pay-per-use model

No idle infrastructure



6. Sustainability

Serverless reduces
resource waste

Business Value & ROI

- Reduced infrastructure cost
- Eliminated manual verification
- Improved operational efficiency
- Scalable enterprise deployment
- **Value Gains:**
 - Faster authentication
 - Lower operational overhead
 - Increased security compliance



Real-World Applications

- Corporate access control
- Banking (KYC)
- Healthcare identity systems
- Retail security
- Logistics verification



Results & Validation

- End-to-end system validated
- All services integrated successfully
- High accuracy recognition
- Reliable API responses
- **Evidence:**
 - CloudWatch logs
 - UI testing
 - DynamoDB records



Key Achievements



Fully serverless
AI system



Real-time
identity
verification



Secure IAM-
based
architecture

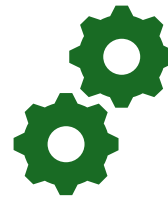


Production-
grade cloud
solution

Conclusion



**Successfully built a
scalable, secure AI
system**



Demonstrated:

Serverless architecture
AI integration
DevOps practices



**Final Insight:
Cloud + AI enables
intelligent, real-time
systems at scale**

Frontend Interface (User Input Layer)

d1msbxhxflfm9d.cloudfront.net

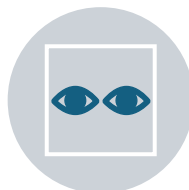
Facial Recognition System

Choose File No file chosen

Upload

Snapshot

Live



Displays the **user-facing interface** of the facial recognition system



Built using a **React-based frontend (Vite development server)**



Allows users to input **first name, last name, and upload an image**



Serves as the **entry point for authentication and registration workflows**



Sends user data and image to the backend via **API Gateway**



Provides a simple and intuitive interface for **real-time interaction**

Facial Recognition Match Results

Facial Recognition System

Status: Success

Time: 4/26/2026, 1:16:19 PM

Choose File jeff-bezos-2.jpg

UploadSnapshotLive



Employee Image

Facial Recognition System

Status: Failure

Time: 4/26/2026, 1:18:03 PM

Choose File kevin-spacey-2.jpg

UploadSnapshotLive



Visitor Image

Thank You