Location-Locked Attendance System: Implementation Guide for DataGuard HRIS

Document Version: 1.0
Date: February 15, 2026
Target: Product Development Team
Platform: DataGuard Employee Portal (dataguardng.com/employee-portal)

Executive Summary

This document provides comprehensive technical specifications for implementing a location-locked, identity-verified time and attendance system within the DataGuard Employee Portal. The system prevents time theft, buddy punching, and remote clock-in abuse through mandatory GPS geofencing, device binding, one-time password (OTP) verification, and policy-backed controls[1][2][3].

Core Security Principle: Staff must be physically present at approved work locations AND identity-verified before the system records attendance.

Key Anti-Abuse Mechanisms:

- GPS geofencing with configurable radius per site
- IP address whitelisting for office networks
- Device fingerprinting and registration
- OTP verification at clock-in
- Anomaly detection and audit logging
- Policy enforcement with consequences

System Requirements Overview

Functional Requirements

- Clock-in/clock-out ONLY allowed within designated geofences (office, warehouse, client sites)
- Real-time GPS coordinate validation before displaying clock-in button
- Device binding requiring one-time registration per staff device
- OTP sent to staff phone number for first-time device or suspicious activity
- Automatic blocking of duplicate clock-ins and missing clock-outs
- Manager approval required for manual corrections or off-location overrides
- Comprehensive audit logging of all attendance actions
- HR dashboard with real-time "who's clocked in" and anomaly alerts

Non-Functional Requirements

- GPS accuracy: ±10-20 meters (standard mobile GPS)
- Clock-in response time: $<$3 seconds from button tap to confirmation
- Support 100+ concurrent clock-in requests during peak times (8:00-9:00 AM)
- Offline-capable: Queue clock-in requests if network temporarily unavailable
- Cross-browser compatibility: Chrome, Firefox, Safari, Edge (mobile and desktop)
- NDPR compliance: Obtain consent for location tracking, encrypt location data[4]

Architecture Overview

Technology Stack

Frontend:

- Next.js 14+ (React framework) with geolocation API
- FingerprintJS library for device identification

- Mapbox or Leaflet.js for geofence visualization (admin config)
- TailwindCSS for UI components

Backend:

- Supabase PostgreSQL database with PostGIS extension for geospatial queries
- Supabase Edge Functions for serverless API endpoints
- Haversine formula for distance calculations
- Twilio or BulkSMS Nigeria for OTP delivery

Security:

- HTTPS/TLS 1.3 for all communications
- AES-256 encryption for stored GPS coordinates
- JWT tokens with 30-minute expiry for session management
- IP address logging and validation

Database Schema

Sites Table

```
CREATE TABLE sites (
id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
name VARCHAR(100) NOT NULL,
site_type VARCHAR(50), -- 'office', 'warehouse', 'client_site'
latitude DECIMAL(10, 8) NOT NULL,
longitude DECIMAL(11, 8) NOT NULL,
radius_meters INTEGER NOT NULL DEFAULT 150,
ip_whitelist TEXT[], -- optional office IP ranges
is_active BOOLEAN DEFAULT true,
created_at TIMESTAMPTZ DEFAULT NOW(),
updated_at TIMESTAMPTZ DEFAULT NOW()
);
```

Sample Data:

```
INSERT INTO sites (name, site_type, latitude, longitude,
radius_meters, ip_whitelist) VALUES
('Lagos HQ', 'office', 6.524379, 3.379206, 150, ARRAY['197.210.0.0/16',
'102.89.0.0/16']),
('Ikeja Warehouse', 'warehouse', 6.599832, 3.339201, 200, NULL),
```

('PremiumTrust Bank Site', 'client_site', 6.431562, 3.425891, 100, NULL);

Time Entries Table

```
CREATE TABLE time_entries (
id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
staff_id UUID NOT NULL REFERENCES staff(id) ON DELETE CASCADE,
site_id UUID NOT NULL REFERENCES sites(id),
entry_date DATE NOT NULL,

-- Clock-in data
clock_in_timestamp TIMESTAMPTZ NOT NULL,
clock_in_latitude DECIMAL(10, 8),
clock_in_longitude DECIMAL(11, 8),
clock_in_ip INET,
clock_in_device_fingerprint TEXT,
clock_in_distance_meters DECIMAL(8, 2), -- distance from site center

-- Clock-out data
clock_out_timestamp TIMESTAMPTZ,
clock_out_latitude DECIMAL(10, 8),
clock_out_longitude DECIMAL(11, 8),
clock_out_ip INET,
clock_out_device_fingerprint TEXT,
clock_out_distance_meters DECIMAL(8, 2),

-- Break tracking
break_minutes INTEGER DEFAULT 0,

-- Status and validation
status VARCHAR(20) DEFAULT 'PENDING', -- 'PENDING', 'APPROVED',
'FLAGGED', 'REJECTED'
validation_notes TEXT,
approved_by UUID REFERENCES staff(id),
approved_at TIMESTAMPTZ,

-- Audit
created_at TIMESTAMPTZ DEFAULT NOW(),
updated_at TIMESTAMPTZ DEFAULT NOW(),
```

```
-- Constraints
CONSTRAINT unique_staff_date UNIQUE(staff_id, entry_date),
CONSTRAINT clock_out_after_clock_in CHECK(clock_out_timestamp
IS NULL OR clock_out_timestamp > clock_in_timestamp)
);

CREATE INDEX idx_time_entries_staff_date ON time_entries(staff_id,
entry_date DESC);
CREATE INDEX idx_time_entries_status ON time_entries(status);
CREATE INDEX idx_time_entries_site ON time_entries(site_id);
```

Device Fingerprints Table

```
CREATE TABLE device_fingerprints (
id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
staff_id UUID NOT NULL REFERENCES staff(id) ON DELETE CASCADE,
fingerprint_hash TEXT NOT NULL,
device_name VARCHAR(100), -- e.g., "iPhone 13", "Chrome on
Windows"
user_agent TEXT,
is_verified BOOLEAN DEFAULT false,
registered_at TIMESTAMPTZ DEFAULT NOW(),
last_used_at TIMESTAMPTZ DEFAULT NOW(),

CONSTRAINT unique_staff_fingerprint UNIQUE(staff_id,
fingerprint_hash)
);

CREATE INDEX idx_fingerprints_staff ON
device_fingerprints(staff_id);
CREATE INDEX idx_fingerprints_hash ON
device_fingerprints(fingerprint_hash);
```

Attendance Audit Log

```
CREATE TABLE attendance_audit_log (
id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
staff_id UUID NOT NULL REFERENCES staff(id),
time_entry_id UUID REFERENCES time_entries(id),
action VARCHAR(50) NOT NULL, -- 'CLOCK_IN', 'CLOCK_OUT',
'MANUAL_EDIT', 'APPROVAL', 'REJECTION'
actor_id UUID REFERENCES staff(id), -- who performed the action
```

```
action_details JSONB, -- flexible field for additional context
ip_address INET,
timestamp TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_audit_staff ON attendance_audit_log(staff_id,
timestamp DESC);
CREATE INDEX idx_audit_entry ON
attendance_audit_log(time_entry_id);
```

OTP Verification Table

```
CREATE TABLE otp_verifications (
id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
staff_id UUID NOT NULL REFERENCES staff(id),
phone_number VARCHAR(15) NOT NULL,
otp_code VARCHAR(6) NOT NULL,
device_fingerprint TEXT NOT NULL,
purpose VARCHAR(50) DEFAULT 'CLOCK_IN', -- 'CLOCK_IN',
'DEVICE_REGISTRATION'
is_verified BOOLEAN DEFAULT false,
expires_at TIMESTAMPTZ NOT NULL,
verified_at TIMESTAMPTZ,
created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_otp_staff ON otp_verifications(staff_id, created_at
DESC);
```

Implementation Specification

Phase 1: Geofencing and Location Validation

Frontend: GPS Coordinate Capture

Component: AttendanceWidget.jsx

```
import { useState, useEffect } from 'react';
import FingerprintJS from '@fingerprintjs/fingerprintjs';
```

export default function AttendanceWidget({ staffId, assignedSiteId }) {
const [gpsStatus, setGpsStatus] = useState('DETECTING'); //
DETECTING, INSIDE, OUTSIDE, ERROR
const [userCoords, setUserCoords] = useState(null);
const [clockedIn, setClockedIn] = useState(false);
const [deviceFingerprint, setDeviceFingerprint] = useState(null);

// Get device fingerprint on mount
useEffect(() => {
const getFingerprint = async () => {
const fp = await FingerprintJS.load();
const result = await fp.get();
setDeviceFingerprint(result.visitorId);
};
getFingerprint();
}, []);

// Get GPS location
const checkLocation = () => {
if (!navigator.geolocation) {
setGpsStatus('ERROR');
alert('GPS not supported on this device');
return;
}

```
setGpsStatus('DETECTING');
navigator.geolocation.getCurrentPosition(
  async (position) => {
    const coords = {
      latitude: position.coords.latitude,
      longitude: position.coords.longitude,
      accuracy: position.coords.accuracy
    };
    setUserCoords(coords);

    // Validate location with backend
    const response = await fetch('/api/attendance/validate-location', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
```

```
      body: JSON.stringify({
        staffId,
        siteId: assignedSiteId,
        coords,
        deviceFingerprint
      })
    });

    const data = await response.json();
    if (data.isInsideGeofence) {
      setGpsStatus('INSIDE');
    } else {
      setGpsStatus('OUTSIDE');
    }
  },
  (error) => {
    console.error('GPS error:', error);
    setGpsStatus('ERROR');
    alert('Cannot access GPS. Please enable location services.');
  },
  { enableHighAccuracy: true, timeout: 10000, maximumAge: 0 }
);
```

};

// Clock-in handler
const handleClockIn = async () => {
if (gpsStatus !== 'INSIDE' || !userCoords) {
alert('You must be at the approved work location to clock in');
return;
}

```
  const response = await fetch('/api/attendance/clock-in', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
      staffId,
      siteId: assignedSiteId,
```

```
    coords: userCoords,
    deviceFingerprint,
    userAgent: navigator.userAgent
  })
});

const data = await response.json();
if (data.requiresOTP) {
  // Prompt for OTP input
  const otp = prompt('Enter the OTP sent to your phone:');
  await verifyOTP(otp, data.verificationId);
} else if (data.success) {
  setClockedIn(true);
  alert(`Clocked in successfully at ${data.siteName}`);
} else {
  alert(`Clock-in failed: ${data.message}`);
}
```

};

const verifyOTP = async (otp, verificationId) => {
const response = await fetch('/api/attendance/verify-otp', {
method: 'POST',
headers: { 'Content-Type': 'application/json' },
body: JSON.stringify({ verificationId, otp })
});

```
const data = await response.json();
if (data.success) {
  setClockedIn(true);
  alert('Clock-in successful after OTP verification');
} else {
  alert('Invalid OTP. Please try again.');
}
```

};

useEffect(() => {
// Check location on component mount and every 60 seconds
checkLocation();
const interval = setInterval(checkLocation, 60000);
return () => clearInterval(interval);
}, []);

return (

Time & Attendance

```
{gpsStatus === 'DETECTING' && (
  <div className="text-yellow-600">Detecting your location...</div>
)}

{gpsStatus === 'OUTSIDE' && (
  <div className="text-red-600 font-semibold">
    ✖ You are outside the approved work location. Clock-in disabled.
  </div>
)}

{gpsStatus === 'ERROR' && (
  <div className="text-red-600">
    ⚠ Location services unavailable. Please enable GPS and refresh.
  </div>
)}

{gpsStatus === 'INSIDE' && !clockedIn && (
  <div>
    <div className="text-green-600 mb-4">✓ Location verified. You may cloc
    <button
      onClick={handleClockIn}
      className="bg-blue-600 text-white px-6 py-3 rounded-lg font-semibold h
    >
      Clock In
    </button>
```

```
    </div>
  )}

  {clockedIn && (
    <div className="text-green-600">
      <div className="font-bold">✅ Clocked In</div>
      <div>Time: {new Date().toLocaleTimeString()}</div>
      <button className="mt-4 bg-red-600 text-white px-6 py-3 rounded-lg">
        Clock Out
      </button>
    </div>
  )}
</div>
```

);
}

Backend: Location Validation API

Endpoint: /api/attendance/validate-location

```
import { createClient } from '@supabase/supabase-js';

const supabase = createClient(process.env.SUPABASE_URL,
process.env.SUPABASE_KEY);

// Haversine formula to calculate distance between two GPS points
function haversineDistance(lat1, lon1, lat2, lon2) {
const R = 6371e3; // Earth radius in meters
const φ1 = (lat1 * Math.PI) / 180;
const φ2 = (lat2 * Math.PI) / 180;
const Δφ = ((lat2 - lat1) * Math.PI) / 180;
const Δλ = ((lon2 - lon1) * Math.PI) / 180;

const a =
Math.sin(Δφ / 2) * Math.sin(Δφ / 2) +
Math.cos(φ1) * Math.cos(φ2) * Math.sin(Δλ / 2) * Math.sin(Δλ / 2);
const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

return R * c; // Distance in meters
}
```

```
export default async function handler(req, res) {
if (req.method !== 'POST') {
return res.status(405).json({ error: 'Method not allowed' });
}

const { staffId, siteId, coords, deviceFingerprint } = req.body;

// 1. Fetch site details
const { data: site, error: siteError } = await supabase
.from('sites')
.select('*')
.eq('id', siteId)
.eq('is_active', true)
.single();

if (siteError || !site) {
return res.status(404).json({ error: 'Site not found' });
}

// 2. Calculate distance from site center
const distance = haversineDistance(
coords.latitude,
coords.longitude,
site.latitude,
site.longitude
);

// 3. Check if inside geofence
const isInsideGeofence = distance <= site.radius_meters;

// 4. Optional: IP whitelist check for office sites
let ipValid = true;
if (site.ip_whitelist && site.ip_whitelist.length > 0) {
const clientIp = req.headers['x-forwarded-for'] ||
req.connection.remoteAddress;
ipValid = site.ip_whitelist.some(range => isIpInRange(clientIp, range));
}

return res.status(200).json({
isInsideGeofence: isInsideGeofence && ipValid,
distance: Math.round(distance),
```

```
siteName: site.name,
siteRadius: site.radius_meters,
coords: { latitude: coords.latitude, longitude: coords.longitude }
});
}

function isIpInRange(ip, range) {
// Simple IP range check (implement CIDR notation parser for
production)
// For MVP, exact match or startsWith
return ip.startsWith(range.split('/')[0].substring(0, 10));
}
```

Phase 2: Device Binding and OTP Verification

Backend: Clock-In with Device Check

Endpoint: /api/attendance/clock-in

```
import { createClient } from '@supabase/supabase-js';
import { sendOTP } from '../utils/sms'; // Your SMS provider integration

const supabase = createClient(process.env.SUPABASE_URL,
process.env.SUPABASE_KEY);

export default async function handler(req, res) {
if (req.method !== 'POST') {
return res.status(405).json({ error: 'Method not allowed' });
}

const { staffId, siteId, coords, deviceFingerprint, userAgent } =
req.body;

// 1. Check if staff already clocked in today
const today = new Date().toISOString().split('T')[0];
const { data: existingEntry } = await supabase
.from('time_entries')
.select('*')
.eq('staff_id', staffId)
.eq('entry_date', today)
.is('clock_out_timestamp', null)
.single();
```

```
if (existingEntry) {
return res.status(400).json({
success: false,
message: 'You are already clocked in. Clock out first.'
});
}

// 2. Validate location (re-check server-side for security)
const { data: site } = await supabase
.from('sites')
.select('*')
.eq('id', siteId)
.single();

const distance = haversineDistance(
coords.latitude,
coords.longitude,
site.latitude,
site.longitude
);

if (distance > site.radius_meters) {
// Log failed attempt
await supabase.from('attendance_audit_log').insert({
staff_id: staffId,
action: 'CLOCK_IN_FAILED',
action_details: { reason: 'OUTSIDE_GEOFENCE', distance, site_id:
siteId },
ip_address: req.headers['x-forwarded-for'] ||
req.connection.remoteAddress
});
```

```
  return res.status(403).json({
    success: false,
    message: 'Location validation failed. You are outside the approved area.'
  });
```

```
}
```

// 3. Check device registration
const { data: registeredDevice } = await supabase
.from('device_fingerprints')
.select('*')
.eq('staff_id', staffId)
.eq('fingerprint_hash', deviceFingerprint)
.eq('is_verified', true)
.single();

if (!registeredDevice) {
// New device detected - require OTP
const { data: staff } = await supabase
.from('staff')
.select('phone_number')
.eq('id', staffId)
.single();

```
const otpCode = Math.floor(100000 + Math.random() * 900000).toString();
const expiresAt = new Date(Date.now() + 10 * 60 * 1000); // 10 minutes

// Store OTP
const { data: otpRecord } = await supabase
  .from('otp_verifications')
  .insert({
    staff_id: staffId,
    phone_number: staff.phone_number,
    otp_code: otpCode,
    device_fingerprint: deviceFingerprint,
    purpose: 'CLOCK_IN',
    expires_at: expiresAt
  })
  .select()
  .single();

// Send OTP via SMS
await sendOTP(staff.phone_number, otpCode);

return res.status(200).json({
```

```
    success: false,
    requiresOTP: true,
    verificationId: otpRecord.id,
    message: 'New device detected. OTP sent to your phone.'
  });
```

}

// 4. Record clock-in
const { data: timeEntry, error } = await supabase
.from('time_entries')
.insert({
staff_id: staffId,
site_id: siteId,
entry_date: today,
clock_in_timestamp: new Date().toISOString(),
clock_in_latitude: coords.latitude,
clock_in_longitude: coords.longitude,
clock_in_ip: req.headers['x-forwarded-for'] ||
req.connection.remoteAddress,
clock_in_device_fingerprint: deviceFingerprint,
clock_in_distance_meters: distance,
status: 'PENDING'
})
.select()
.single();

if (error) {
return res.status(500).json({ success: false, message: 'Database error',
error });
}

// 5. Log successful clock-in
await supabase.from('attendance_audit_log').insert({
staff_id: staffId,
time_entry_id: timeEntry.id,
action: 'CLOCK_IN',
action_details: { site_name: site.name, distance },
ip_address: req.headers['x-forwarded-for'] ||

```
req.connection.remoteAddress
});

// 6. Update device last used
await supabase
.from('device_fingerprints')
.update({ last_used_at: new Date().toISOString() })
.eq('fingerprint_hash', deviceFingerprint)
.eq('staff_id', staffId);

return res.status(200).json({
success: true,
timeEntryId: timeEntry.id,
siteName: site.name,
clockInTime: timeEntry.clock_in_timestamp
});
}

function haversineDistance(lat1, lon1, lat2, lon2) {
// Same implementation as Phase 1
const R = 6371e3;
const φ1 = (lat1 * Math.PI) / 180;
const φ2 = (lat2 * Math.PI) / 180;
const Δφ = ((lat2 - lat1) * Math.PI) / 180;
const Δλ = ((lon2 - lon1) * Math.PI) / 180;

const a =
Math.sin(Δφ / 2) * Math.sin(Δφ / 2) +
Math.cos(φ1) * Math.cos(φ2) * Math.sin(Δλ / 2) * Math.sin(Δλ / 2);
const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

return R * c;
}
```

Backend: OTP Verification

Endpoint: /api/attendance/verify-otp

```
import { createClient } from '@supabase/supabase-js';

const supabase = createClient(process.env.SUPABASE_URL,
process.env.SUPABASE_KEY);
```

```
export default async function handler(req, res) {
if (req.method !== 'POST') {
return res.status(405).json({ error: 'Method not allowed' });
}

const { verificationId, otp } = req.body;

// 1. Fetch OTP record
const { data: otpRecord, error } = await supabase
.from('otp_verifications')
.select('*')
.eq('id', verificationId)
.single();

if (error || !otpRecord) {
return res.status(404).json({ success: false, message: 'OTP record not
found' });
}

// 2. Check if OTP expired
if (new Date() > new Date(otpRecord.expires_at)) {
return res.status(400).json({ success: false, message: 'OTP expired.
Request a new one.' });
}

// 3. Verify OTP code
if (otpRecord.otp_code !== otp) {
return res.status(400).json({ success: false, message: 'Invalid OTP
code' });
}

// 4. Mark OTP as verified
await supabase
.from('otp_verifications')
.update({ is_verified: true, verified_at: new Date().toISOString() })
.eq('id', verificationId);

// 5. Register device as verified
await supabase.from('device_fingerprints').insert({
staff_id: otpRecord.staff_id,
fingerprint_hash: otpRecord.device_fingerprint,
```

```
device_name: 'Web Browser', // Can extract from user agent
is_verified: true,
registered_at: new Date().toISOString(),
last_used_at: new Date().toISOString()
});

return res.status(200).json({
success: true,
message: 'Device registered successfully. You may now clock in.'
});
}
```

Phase 3: Clock-Out and Break Tracking

Backend: Clock-Out API

Endpoint: /api/attendance/clock-out

```
import { createClient } from '@supabase/supabase-js';

const supabase = createClient(process.env.SUPABASE_URL,
process.env.SUPABASE_KEY);

export default async function handler(req, res) {
if (req.method !== 'POST') {
return res.status(405).json({ error: 'Method not allowed' });
}

const { staffId, coords, deviceFingerprint } = req.body;

// 1. Find today's active time entry
const today = new Date().toISOString().split('T')[0];
const { data: timeEntry, error } = await supabase
.from('time_entries')
.select(', sites()')
.eq('staff_id', staffId)
.eq('entry_date', today)
.is('clock_out_timestamp', null)
.single();

if (error || !timeEntry) {
return res.status(404).json({
```

```
success: false,
message: 'No active clock-in found for today'
});
}

// 2. Validate location (same geofence requirement)
const distance = haversineDistance(
coords.latitude,
coords.longitude,
timeEntry.sites.latitude,
timeEntry.sites.longitude
);

if (distance > timeEntry.sites.radius_meters) {
return res.status(403).json({
success: false,
message: 'You must be at the work location to clock out'
});
}

// 3. Update time entry with clock-out data
const { data: updatedEntry } = await supabase
.from('time_entries')
.update({
clock_out_timestamp: new Date().toISOString(),
clock_out_latitude: coords.latitude,
clock_out_longitude: coords.longitude,
clock_out_ip: req.headers['x-forwarded-for'] ||
req.connection.remoteAddress,
clock_out_device_fingerprint: deviceFingerprint,
clock_out_distance_meters: distance,
updated_at: new Date().toISOString()
})
.eq('id', timeEntry.id)
.select()
.single();

// 4. Calculate total hours worked
const clockInTime = new Date(updatedEntry.clock_in_timestamp);
const clockOutTime = new Date(updatedEntry.clock_out_timestamp);
```

```
const hoursWorked = ((clockOutTime - clockInTime) / (1000 * 60 *
60)).toFixed(2);

// 5. Log clock-out
await supabase.from('attendance_audit_log').insert({
staff_id: staffId,
time_entry_id: timeEntry.id,
action: 'CLOCK_OUT',
action_details: { hours_worked: hoursWorked, site_name:
timeEntry.sites.name },
ip_address: req.headers['x-forwarded-for'] ||
req.connection.remoteAddress
});

return res.status(200).json({
success: true,
hoursWorked,
clockOutTime: updatedEntry.clock_out_timestamp,
message: Clocked out successfully. Total hours: ${hoursWorked}
});
}

function haversineDistance(lat1, lon1, lat2, lon2) {
const R = 6371e3;
const φ1 = (lat1 * Math.PI) / 180;
const φ2 = (lat2 * Math.PI) / 180;
const Δφ = ((lat2 - lat1) * Math.PI) / 180;
const Δλ = ((lon2 - lon1) * Math.PI) / 180;

const a =
Math.sin(Δφ / 2) * Math.sin(Δφ / 2) +
Math.cos(φ1) * Math.cos(φ2) * Math.sin(Δλ / 2) * Math.sin(Δλ / 2);
const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

return R * c;
}
```

Phase 4: Anti-Abuse Rules and Anomaly Detection

Backend: Scheduled Anomaly Detection Job

Function: /api/cron/detect-anomalies (run daily at 11:00 PM)

```
import { createClient } from '@supabase/supabase-js';

const supabase = createClient(process.env.SUPABASE_URL,
process.env.SUPABASE_KEY);

export default async function handler(req, res) {
const today = new Date().toISOString().split('T')[0];
const anomalies = [];

// 1. Same device used by multiple staff within 30 minutes
const { data: deviceSharing } = await
supabase.rpc('detect_device_sharing', { check_date: today });

if (deviceSharing && deviceSharing.length > 0) {
anomalies.push({
type: 'DEVICE_SHARING',
description: 'Multiple staff used same device within 30 minutes',
details: deviceSharing
});
}

// 2. Clock-ins at geofence edge (within 10m of boundary) 3+ times
const { data: edgeClockIns } = await supabase
.from('time_entries')
.select('staff_id, staff(full_name), clock_in_distance_meters')
.gte('entry_date', new Date(Date.now() - 7 * 24 * 60 * 60 *
1000).toISOString().split('T')[0])
.gte('clock_in_distance_meters', 140) // Within 10m of 150m radius
.lte('clock_in_distance_meters', 150);

const edgeAbusers = {};
edgeClockIns.forEach(entry => {
if (!edgeAbusers[entry.staff_id]) {
edgeAbusers[entry.staff_id] = { count: 0, name: entry.staff.full_name };
}
```

```
edgeAbusers[entry.staff_id].count++;
});

Object.entries(edgeAbusers).forEach(([staffId, data]) => {
if (data.count >= 3) {
anomalies.push({
type: 'GEOFENCE_EDGE_ABUSE',
description: ${data.name} clocked in at geofence edge ${data.count} times in past week,
staff_id: staffId
});
}
});

// 3. Clock-in during approved leave
const { data: leaveViolations } = await supabase
.from('time_entries')
.select(', staff(full_name), leave_requests!inner()')
.eq('entry_date', today)
.eq('leave_requests.status', 'APPROVED')
.gte('leave_requests.start_date', today)
.lte('leave_requests.end_date', today);

if (leaveViolations && leaveViolations.length > 0) {
anomalies.push({
type: 'LEAVE_VIOLATION',
description: 'Staff clocked in while on approved leave',
details: leaveViolations.map(v => ({ name: v.staff.full_name, staff_id: v.staff_id }))
});
}

// 4. Missing clock-outs (clocked in yesterday but no clock-out)
const yesterday = new Date(Date.now() - 24 * 60 * 60 * 1000).toISOString().split('T')[0];
const { data: missingClockOuts } = await supabase
.from('time_entries')
.select('staff_id, staff(full_name, email)')
.eq('entry_date', yesterday)
.is('clock_out_timestamp', null);
```

```
if (missingClockOuts && missingClockOuts.length > 0) {
anomalies.push({
type: 'MISSING_CLOCK_OUT',
description: 'Staff forgot to clock out yesterday',
details: missingClockOuts
});
}

// 5. Flag suspicious entries in database
for (const anomaly of anomalies) {
if (anomaly.type === 'DEVICE_SHARING' || anomaly.type ===
'GEOFENCE_EDGE_ABUSE') {
// Flag time entries for HR review
const staffIds = anomaly.details.map(d => d.staff_id);
await supabase
.from('time_entries')
.update({ status: 'FLAGGED', validation_notes: anomaly.description })
.in('staff_id', staffIds)
.eq('entry_date', today);
}
}

// 6. Send email alert to HR
if (anomalies.length > 0) {
await sendEmailAlert('hr@dataguardng.com', 'Attendance Anomalies
Detected', anomalies);
}

return res.status(200).json({
success: true,
anomaliesDetected: anomalies.length,
anomalies
});
}

async function sendEmailAlert(to, subject, anomalies) {
// Implement email sending (SendGrid, AWS SES, etc.)
console.log('Sending email to:', to);
console.log('Anomalies:', JSON.stringify(anomalies, null, 2));
}
```

Database Function: detect_device_sharing

CREATE OR REPLACE FUNCTION detect_device_sharing(check_date DATE)
RETURNS TABLE(device_fingerprint TEXT, staff_ids UUID[], staff_names TEXT[], clock_in_times TIMESTAMPTZ[])

AS
$BEGIN RETURN QUERY SELECT t.clock_in_device_fingerpri$
LANGUAGE plpgsql;

Phase 5: HR Dashboard and Reporting

Frontend: HR Attendance Dashboard

Component: HRAttendanceDashboard.jsx

import { useEffect, useState } from 'react';

export default function HRAttendanceDashboard() {
const [todayStats, setTodayStats] = useState({});
const [activeStaff, setActiveStaff] = useState([]);
const [anomalies, setAnomalies] = useState([]);

useEffect(() => {
fetchTodayStats();
fetchActiveStaff();
fetchAnomalies();

```
// Refresh every 5 minutes
const interval = setInterval(() => {
  fetchTodayStats();
  fetchActiveStaff();
}, 5 * 60 * 1000);

return () => clearInterval(interval);
```

}, []);

const fetchTodayStats = async () => {
const res = await fetch('/api/hr/attendance-stats');

```
const data = await res.json();
setTodayStats(data);
};

const fetchActiveStaff = async () => {
const res = await fetch('/api/hr/active-staff');
const data = await res.json();
setActiveStaff(data);
};

const fetchAnomalies = async () => {
const res = await fetch('/api/hr/anomalies');
const data = await res.json();
setAnomalies(data);
};

return (
<div className="p-6">
```

HR Attendance Dashboard

```
{/* Stats Cards */}
<div className="grid grid-cols-4 gap-4 mb-8">
  <div className="bg-green-100 p-4 rounded-lg">
    <div className="text-3xl font-bold text-green-700">{todayStats.clockedIn
    <div className="text-sm text-green-600">Clocked In Today</div>
  </div>
  <div className="bg-blue-100 p-4 rounded-lg">
    <div className="text-3xl font-bold text-blue-700">{todayStats.activeNow
    <div className="text-sm text-blue-600">Currently Active</div>
  </div>
  <div className="bg-yellow-100 p-4 rounded-lg">
    <div className="text-3xl font-bold text-yellow-700">{todayStats.lateArriv
    <div className="text-sm text-yellow-600">Late Arrivals</div>
  </div>
  <div className="bg-red-100 p-4 rounded-lg">
    <div className="text-3xl font-bold text-red-700">{anomalies.length || 0}
    <div className="text-sm text-red-600">Anomalies Detected</div>
  </div>
```

```
</div>

{/* Anomaly Alerts */}
{anomalies.length > 0 && (
  <div className="bg-red-50 border border-red-300 p-4 rounded-lg mb-6">
    <h2 className="text-lg font-bold text-red-700 mb-2">⚠ Anomalies Detecte
    <ul className="list-disc list-inside">
      {anomalies.map((anomaly, idx) => (
        <li key={idx} className="text-red-600">{anomaly.description}</li>
      ))}
    </ul>
  </div>
)}

{/* Currently Active Staff */}
<div className="bg-white p-6 rounded-lg shadow">
  <h2 className="text-xl font-bold mb-4">Currently Clocked In ({activeStaff.
  <table className="w-full">
    <thead>
      <tr className="border-b">
        <th className="text-left p-2">Staff Name</th>
        <th className="text-left p-2">Site</th>
        <th className="text-left p-2">Clock-In Time</th>
        <th className="text-left p-2">Duration</th>
      </tr>
    </thead>
    <tbody>
      {activeStaff.map((staff) => (
        <tr key={staff.id} className="border-b">
          <td className="p-2">{staff.full_name}</td>
          <td className="p-2">{staff.site_name}</td>
          <td className="p-2">{new Date(staff.clock_in_time).toLocaleTimeStrin
          <td className="p-2">{staff.duration}</td>
        </tr>
      ))}
    </tbody>
  </table>
```

```
  </div>
</div>
```

);
}

Backend: HR Dashboard APIs

Endpoint: /api/hr/attendance-stats

```
import { createClient } from '@supabase/supabase-js';

const supabase = createClient(process.env.SUPABASE_URL,
process.env.SUPABASE_KEY);

export default async function handler(req, res) {
const today = new Date().toISOString().split('T')[0];

// Total clocked in today
const { count: clockedIn } = await supabase
.from('time_entries')
.select('*', { count: 'exact', head: true })
.eq('entry_date', today);

// Currently active (clocked in but not clocked out)
const { count: activeNow } = await supabase
.from('time_entries')
.select('*', { count: 'exact', head: true })
.eq('entry_date', today)
.is('clock_out_timestamp', null);

// Late arrivals (clock-in after 9:00 AM)
const { count: lateArrivals } = await supabase
.from('time_entries')
.select('*', { count: 'exact', head: true })
.eq('entry_date', today)
.gte('clock_in_timestamp', ${today}T09:00:00);

return res.status(200).json({
clockedIn: clockedIn || 0,
activeNow: activeNow || 0,
lateArrivals: lateArrivals || 0
```

});
}

Endpoint: /api/hr/active-staff

import { createClient } from '@supabase/supabase-js';

const supabase = createClient(process.env.SUPABASE_URL,
process.env.SUPABASE_KEY);

export default async function handler(req, res) {
const today = new Date().toISOString().split('T')[0];

const { data: activeStaff } = await supabase
.from('time_entries')
.select('*, staff(full_name, staff_id), sites(name)')
.eq('entry_date', today)
.is('clock_out_timestamp', null)
.order('clock_in_timestamp', { ascending: false });

const staffWithDuration = activeStaff.map((entry) => {
const clockInTime = new Date(entry.clock_in_timestamp);
const now = new Date();
const durationMs = now - clockInTime;
const hours = Math.floor(durationMs / (1000 * 60 * 60));
const minutes = Math.floor((durationMs % (1000 * 60 * 60)) / (1000 * 60));

```
return {
  id: entry.id,
  full_name: entry.staff.full_name,
  staff_id: entry.staff.staff_id,
  site_name: entry.sites.name,
  clock_in_time: entry.clock_in_timestamp,
  duration: `${hours}h ${minutes}m`
};
```

});

```
return res.status(200).json(staffWithDuration);
}
```

Policy and Enforcement

Attendance and Time-Theft Policy

Policy Statement:

DataGuard requires accurate time and attendance records for all staff. Buddy punching (clocking in/out for another employee), falsifying attendance records, or circumventing location verification systems constitutes time theft and is grounds for disciplinary action up to and including termination[5][6].

Definitions:

- **Buddy Punching**: Having another employee clock in or out on your behalf.
- **Location Spoofing**: Using GPS manipulation tools or VPNs to falsify location data.
- **Credential Sharing**: Providing login details to another person for attendance purposes.

Prohibited Actions:

- Sharing login credentials with any other person
- Using GPS spoofing apps or location manipulation tools
- Asking another employee to clock in/out on your behalf
- Tampering with device fingerprints or security mechanisms
- Making false statements in attendance correction requests

Consequences:

Offense	Disciplinary Action
First offense	Written warning + mandatory policy training
Second offense	Suspension without pay (3-5 days) + final written warning
Third offense	Termination of employment
Severe cases (e.g., organized fraud)	Immediate termination + potential legal action

Table 1: Progressive discipline for attendance violations

Staff Acknowledgment:

All staff must sign an acknowledgment form during onboarding and system launch, confirming they:

- Understand the attendance policy
- Consent to GPS location tracking for attendance purposes
- Agree not to share credentials or engage in time theft
- Accept consequences for policy violations

Manager Responsibilities

Line Managers must:

- Review and approve/reject attendance correction requests within 24 hours
- Investigate flagged anomalies (same device usage, geofence edge patterns)
- Report suspected time theft to HR immediately
- Conduct monthly team attendance reviews and address patterns (chronic lateness, frequent missing clock-outs)
- Maintain confidentiality of attendance investigations

HR Responsibilities:

- Configure and maintain geofences for all sites
- Review flagged attendance entries daily
- Conduct disciplinary proceedings for policy violations
- Generate monthly compliance reports for management

- Ensure NDPR compliance for location data processing[4]
- Train new staff on attendance system during onboarding

Testing and Validation

Test Scenarios

Positive Tests (Should Succeed)

Scenario	Test Steps	Expected Outcome
Valid clock-in	Staff at Lagos HQ, registered device, within geofence	Clock-in recorded, confirmation shown
Valid clock-out	Same location as clock-in, same day	Clock-out recorded, hours calculated
OTP verification	New device, correct OTP entered	Device registered, clock-in proceeds
Manager approval	Correction request submitted, manager approves	Attendance record updated, audit logged

Table 2: Positive test cases

Negative Tests (Should Fail)

Scenario	Test Steps	Expected Outcome
Outside geofence	Staff 500m from office, attempts clock-in	Blocked: "Outside approved location"
Duplicate clock-in	Already clocked in today, attempts again	Blocked: "Already clocked in"
Wrong OTP	New device, incorrect OTP entered 3 times	Account locked for 15 minutes
Buddy punching	Same device, two staff within 30 minutes	Flagged for HR review, both accounts investigated
Leave violation	Staff on approved leave, attempts clock-in	Allowed but flagged: "On leave - review required"

Table 3: Negative test cases

Load Testing Requirements

- Simulate 100 concurrent clock-ins between 8:00-8:05 AM
- Target response time: $<$3 seconds per request
- Database connection pooling: Minimum 20 connections
- API rate limiting: 10 requests per minute per user (prevent rapid-fire abuse)

User Acceptance Testing (UAT)

Week 1-2: Alpha Testing

- 10 volunteer staff (5 Core, 5 Support) from IT and HR
- Test all scenarios: clock-in, clock-out, OTP, corrections
- Collect feedback on UX, GPS accuracy, error messages

Week 3: Beta Testing

- Full department (e.g., Operations team, 20-30 staff)
- One week of live usage alongside existing manual system

- Daily standup meetings to address issues
- HR monitors dashboard and anomaly detection

Deployment and Rollout Plan

Pre-Launch (Week 1-2)

- Configure geofences for all DataGuard sites in production database
- Import existing staff records and assign default site_id
- Set up SMS provider integration and test OTP delivery
- HR prepares communication materials (policy document, user guides, training videos)
- IT conducts security audit and penetration testing

Launch (Week 3)

Day 1 (Monday):

- 8:00 AM: System goes live
- 9:00 AM: All-hands announcement via email and WhatsApp
- 10:00 AM: HR conducts live demo sessions (3 sessions throughout the day)

Day 2-5 (Tuesday-Friday):

- IT support on standby for troubleshooting
- HR monitors dashboard continuously, addresses anomalies in real-time
- Daily feedback collection via WhatsApp support group

Post-Launch (Week 4+)

- Week 4: First monthly attendance report generated and reviewed by management
- Week 5: Policy enforcement begins (warnings issued for first offenses)
- Week 8: System performance review and optimization based on usage patterns
- Month 3: Full compliance audit and success metrics evaluation

---

Maintenance and Support

Daily Operations

- Automated anomaly detection runs at 11:00 PM WAT
- Backup of attendance data at 2:00 AM WAT
- HR reviews flagged entries and anomaly reports by 10:00 AM daily

Weekly Tasks

- IT reviews audit logs for security incidents
- HR generates weekly attendance summary for management
- Performance monitoring: API response times, error rates, GPS accuracy

Monthly Tasks

- Device fingerprint cleanup (remove devices not used in 90+ days)
- OTP log purging (delete verified OTPs older than 30 days)
- Geofence accuracy review (adjust radius if needed based on false positives)
- User feedback survey and feature prioritization

Support Structure

Tier 1: Staff Self-Service

- In-app help tooltips and FAQs
- Video tutorial: "How to Clock In/Out"
- WhatsApp support group for peer assistance

Tier 2: HR Help Desk

- Email: attendance-support@dataguardng.com
- Response time: 4 hours during business hours
- Handles: Correction requests, OTP issues, account lockouts

Tier 3: IT Development Team

- Email: it-dev@dataguardng.com

- Response time: Critical bugs within 2 hours
- Handles: System errors, security issues, feature requests

Security and Compliance

Data Privacy (NDPR Compliance)

Location Data Processing:

DataGuard processes employee GPS coordinates for legitimate business purposes (attendance verification). Staff consent is obtained through:

- Explicit opt-in during onboarding and system launch
- Clear privacy notice explaining data collection, storage, and retention
- Right to access location history via export function
- Right to object (with understanding that clock-in will not function without GPS)

Data Retention:

- Active employees: Attendance records retained for 7 years (Nigeria Labour Act requirement)[7]
- Exited employees: Records archived and anonymized after 7 years
- GPS coordinates: Encrypted AES-256 at rest, only accessible to HR administrators
- Audit logs: Retained for 3 years for compliance investigations

Data Subject Rights:

- Right to access: Staff can view and export their attendance history
- Right to rectification: Correction requests available for errors
- Right to erasure: Upon employment termination + 7 years
- Right to data portability: Export in CSV/JSON format

Security Measures

- **Encryption**: AES-256 for GPS coordinates, TLS 1.3 for transmission
- **Access Control**: Role-based permissions, principle of least privilege
- **Audit Logging**: All CRUD operations logged with timestamp, IP, user ID
- **OTP Expiry**: 10-minute validity window, maximum 3 attempts
- **Device Registration**: Limited to 3 devices per staff member
- **Session Management**: JWT tokens expire after 30 minutes, refresh tokens after 7 days
- **Rate Limiting**: 10 clock-in attempts per hour per user
- **SQL Injection Prevention**: Parameterized queries, Supabase RLS policies

Vulnerability Mitigation

Threat	Risk	Mitigation
GPS Spoofing	Fake location to clock in remotely	Cross-verify with IP geolocation; flag mismatches[2]
Credential Theft	Stolen passwords used by unauthorized person	OTP verification; device binding[3]
Insider Collusion	Managers approve fake corrections	Audit logs reviewed by HR; random sampling
DDoS Attack	System unavailable during peak times	Rate limiting; Cloudflare DDoS protection
Database Breach	Attendance records exposed	Encryption at rest; regular penetration testing

Table 4: Security threat matrix and mitigations

Future Enhancements

Phase 2 Features (Months 3-6)

- **Facial Recognition**: Capture selfie at clock-in for biometric verification (optional)[1]
- **Offline Clock-In**: Queue requests when network unavailable, sync when restored
- **Break Management**: Start/end break buttons with time tracking
- **Shift Scheduling**: Pre-assign shifts, validate clock-in against schedule
- **Overtime Tracking**: Automatic flagging when daily hours exceed threshold

Phase 3 Features (Months 6-12)

- **Mobile App**: Native iOS and Android apps with push notifications
- **Beacon Integration**: Bluetooth beacons for indoor location accuracy
- **Wearable Support**: Smartwatch clock-in via Apple Watch, Wear OS
- **Predictive Analytics**: AI-powered detection of attendance fraud patterns
- **Payroll Integration**: Automatic sync of hours worked to payroll system

---

Appendix A: SMS Provider Integration

Example: Twilio Integration (sms.js)

```
import twilio from 'twilio';

const client = twilio(
process.env.TWILIO_ACCOUNT_SID,
process.env.TWILIO_AUTH_TOKEN
);

export async function sendOTP(phoneNumber, otpCode) {
try {
```

```
const message = await client.messages.create({
body: Your DataGuard attendance OTP is: ${otpCode}. Valid for 10 minutes. Do not share this code.,
from: process.env.TWILIO_PHONE_NUMBER,
to: phoneNumber
});

  console.log('OTP sent:', message.sid);
  return { success: true, messageSid: message.sid };

} catch (error) {
console.error('SMS error:', error);
return { success: false, error: error.message };
}
}
```

Alternative: BulkSMS Nigeria

```
export async function sendOTP(phoneNumber, otpCode) {
const response = await fetch('https://www.bulksmsnigeria.com/api/v1/sms/create', {
method: 'POST',
headers: {
'Content-Type': 'application/json',
'Authorization': Bearer ${process.env.BULKSMS_API_KEY}
},
body: JSON.stringify({
from: 'DataGuard',
to: phoneNumber,
body: Your DataGuard attendance OTP is: ${otpCode}. Valid for 10 minutes.,
dnd: '2' // Override DND for important messages
})
});

return await response.json();
}
```

Appendix B: Geofence Configuration UI (Admin)

Component: SiteManagement.jsx (HR Admin Only)

```
import { useState, useEffect } from 'react';
import { MapContainer, TileLayer, Circle, Marker, useMapEvents }
from 'react-leaflet';

export default function SiteManagement() {
const [sites, setSites] = useState([]);
const [newSite, setNewSite] = useState({
name: '',
latitude: 6.5244,
longitude: 3.3792,
radius: 150,
site_type: 'office'
});

useEffect(() => {
fetchSites();
}, []);

const fetchSites = async () => {
const res = await fetch('/api/admin/sites');
const data = await res.json();
setSites(data);
};

const saveSite = async () => {
await fetch('/api/admin/sites', {
method: 'POST',
headers: { 'Content-Type': 'application/json' },
body: JSON.stringify(newSite)
});
fetchSites();
alert('Site saved successfully');
};
```

return (

Geofence Configuration

```
<div className="grid grid-cols-2 gap-6">
  <div>
    <h2 className="text-lg font-bold mb-2">Add New Site</h2>
    <input
      placeholder="Site Name"
      value={newSite.name}
      onChange={(e) => setNewSite({ ...newSite, name: e.target.value })}
      className="border p-2 w-full mb-2"
    />
    <input
      placeholder="Latitude"
      type="number"
      step="0.000001"
      value={newSite.latitude}
      onChange={(e) => setNewSite({ ...newSite, latitude: parseFloat(e.target.va
      className="border p-2 w-full mb-2"
    />
    <input
      placeholder="Longitude"
      type="number"
      step="0.000001"
      value={newSite.longitude}
      onChange={(e) => setNewSite({ ...newSite, longitude: parseFloat(e.target.v
      className="border p-2 w-full mb-2"
    />
    <input
      placeholder="Radius (meters)"
      type="number"
      value={newSite.radius}
      onChange={(e) => setNewSite({ ...newSite, radius: parseInt(e.target.value
      className="border p-2 w-full mb-4"
    />
```

```
    <button onClick={saveSite} className="bg-blue-600 text-white px-4 py-2 r
      Save Site
    </button>
  </div>

  <div>
    <h2 className="text-lg font-bold mb-2">Map Preview</h2>
    <MapContainer center={[newSite.latitude, newSite.longitude]} zoom={15}
      <TileLayer url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png" />
      <Circle
        center={[newSite.latitude, newSite.longitude]}
        radius={newSite.radius}
        pathOptions={{ color: 'blue' }}
      />
      <Marker position={[newSite.latitude, newSite.longitude]} />
    </MapContainer>
  </div>
</div>

<div className="mt-8">
  <h2 className="text-lg font-bold mb-2">Existing Sites</h2>
  <table className="w-full border">
    <thead>
      <tr className="bg-gray-100">
        <th className="p-2">Name</th>
        <th className="p-2">Coordinates</th>
        <th className="p-2">Radius</th>
        <th className="p-2">Type</th>
      </tr>
    </thead>
    <tbody>
      {sites.map((site) => (
        <tr key={site.id} className="border-t">
          <td className="p-2">{site.name}</td>
          <td className="p-2">{site.latitude}, {site.longitude}</td>
          <td className="p-2">{site.radius}m</td>
          <td className="p-2">{site.site_type}</td>
        </tr>
```

```
        ))}
      </tbody>
    </table>
  </div>
</div>
```

);
}

Appendix C: References

[1] Workstatus. (2025). Location-based attendance tracking with geofencing. https://www.workstatus.io/blog/workforce-management/location-based-attendance-tracking/

[2] Zeba Academy. (2025). Geofencing and location-based attendance. https://zeba.pro/blogs/geofencing-and-location-based-attendance

[3] Buddy Punch. (2025). What is buddy punching and 7 ways to prevent it. https://buddypunch.com/blog/buddy-punching-what-it-is-how-to-prevent-it-time-clock/

[4] Express Global Employment. (2023). Global HR compliance in Nigeria. https://expressglobalemployment.com/countries/global-hr-compliance-in-nigeria/

[5] OLOID. (2025). What is buddy punching and how to prevent it? https://www.oloid.com/blog/buddy-punching

[6] Qandle. (2015). Buddy punching prevention: Complete guide for HR teams. https://www.qandle.com/glossary-buddy-punching

[7] Doheny Services. (2025). Labour law compliance in Nigeria: 2025 guide for employers. https://jobs.doheneyservices.com/blog/labour-law-compliance-in-nigeria-2025-guide-for-employers

---

Document Approval

Role	Name	Signature	Date
Product Owner (HR Lead)	_____	_____	_____
Solution Developer (IT Lead)	_____	_____	_____
Executive Sponsor	_____	_____	_____

END OF DOCUMENT