HRMS Mobile Application - Product Development Document

Stage 1: Geo-Punch Attendance System

1. Project Overview

Project Name: GeoHRMS - Mobile Attendance System
Version: 1.0 (Stage 1)
Target Platform: Cross-platform Mobile (iOS & Android)
Development Timeline: 8-12 weeks
Budget: Free Development Stack

1.1 Project Objectives

- Create a mobile-first HRMS application with geo-location based attendance
- Implement secure employee identification system with unique IDs
- Enable admin approval workflow for attendance locations
- Integrate selfie capture for attendance verification
- Establish foundation for future feature expansions (Stage 2: Leave Management)

1.2 Core Features (Stage 1)

1. **Employee Registration & Authentication**
   - Unique employee ID generation
   - Secure login/logout system
   - Profile management
2. **Geo-Punch Attendance**
   - GPS-based location tracking
   - Geofencing for approved locations
   - Punch-in/Punch-out functionality
   - Location verification
3. **Selfie Verification**
   - Camera integration
   - Photo capture during punch
   - Image storage and management
4. **Admin Dashboard**
   - Location approval workflow
   - Employee management

- Attendance monitoring
- Reports generation

2. Technology Stack (2025 Recommendations)

Based on current market analysis, the following tech stack provides optimal balance of cost-effectiveness, performance, and scalability:

2.1 Frontend (Mobile App)

Framework: React Native with Expo

- **Why:** Cross-platform development, cost-effective, large community support
- **UI Library:** React Native Elements + NativeBase
- **Navigation:** React Navigation v6
- **State Management:** Redux Toolkit + RTK Query
- **Camera:** expo-camera
- **Location Services:** expo-location
- **Maps:** react-native-maps (Google Maps integration)

2.2 Backend

Runtime: Node.js with Express.js

- **Database:** PostgreSQL (Primary) + Redis (Caching)
- **ORM:** Prisma
- **Authentication:** JWT with bcrypt
- **File Storage:** Cloudinary (Free tier - 25GB)
- **API Documentation:** Swagger/OpenAPI

2.3 Infrastructure & Deployment (Free Tier)

Backend Hosting: Railway.app or Render.com (Free tier) **Database:** PostgreSQL on Railway/Render or Supabase (Free tier) **Mobile App Distribution:**

- **Testing:** Expo Go app
- **Production:** Google Play Store + Apple App Store

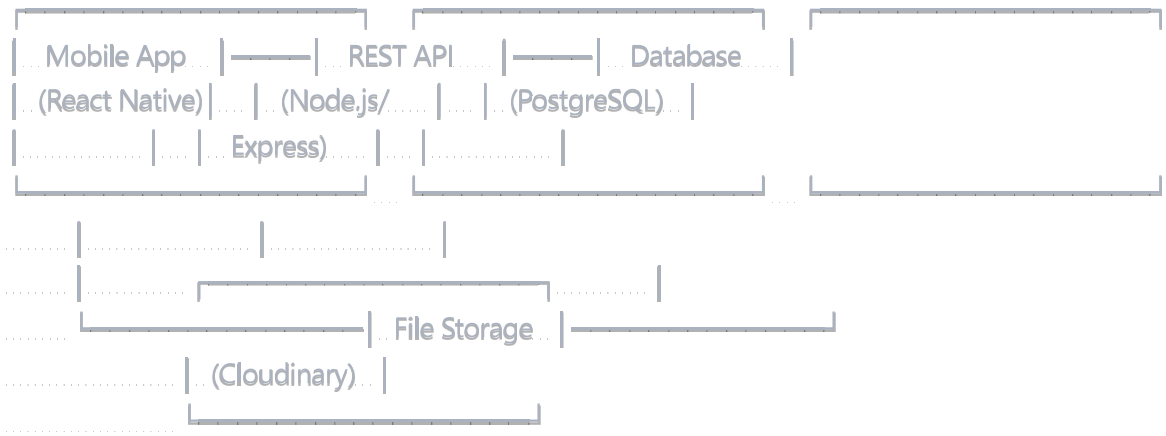
2.4 Development Tools

- **IDE:** Visual Studio Code
- **Version Control:** Git + GitHub
- **API Testing:** Postman

- **Design:** Figma (Free tier)
- **Project Management:** GitHub Projects

3. System Architecture

3.1 High-Level Architecture



3.2 Database Schema Design

Users Table:

```sql
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    employee_id VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    phone VARCHAR(20),
    role ENUM('employee', 'admin') DEFAULT 'employee',
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW()
);
```

Approved Locations Table:

```sql
```

```sql
CREATE TABLE approved_locations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(255) NOT NULL,
    address TEXT NOT NULL,
    latitude DECIMAL(10, 8) NOT NULL,
    longitude DECIMAL(11, 8) NOT NULL,
    radius INTEGER DEFAULT 100, -- meters
    is_active BOOLEAN DEFAULT true,
    created_by UUID REFERENCES users(id),
    created_at TIMESTAMP DEFAULT NOW()
);
```

Attendance Records Table:

```sql
CREATE TABLE attendance_records (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id),
    location_id UUID REFERENCES approved_locations(id),
    punch_type ENUM('in', 'out') NOT NULL,
    punch_time TIMESTAMP NOT NULL,
    latitude DECIMAL(10, 8) NOT NULL,
    longitude DECIMAL(11, 8) NOT NULL,
    selfie_url VARCHAR(500),
    is_verified BOOLEAN DEFAULT false,
    notes TEXT,
    created_at TIMESTAMP DEFAULT NOW()
);
```

4. Feature Specifications

4.1 Employee Authentication System

Unique ID Generation:

- Format: EMP-[YYYY]-[XXXX] (e.g., EMP-2025-0001)
- Auto-increment based on year
- Check uniqueness before assignment

Authentication Flow:

1. Employee registration (admin only)
2. Login with employee ID + password
3. JWT token generation (24-hour expiry)

4. Secure session management

4.2 Geo-Punch System

Location Services:

- Request location permissions on app start
- Continuous GPS tracking during work hours
- Geofencing with configurable radius (default: 100m)
- Offline capability with sync when online

Punch Process:

1. Verify user is within approved location
2. Capture current GPS coordinates
3. Take mandatory selfie
4. Submit attendance record
5. Show confirmation with timestamp

4.3 Admin Features

Location Management:

- Add/edit approved locations
- Set geofence radius
- View location on map
- Enable/disable locations

Employee Management:

- Create employee profiles
- Generate unique IDs
- Manage active/inactive status
- Reset passwords

Attendance Monitoring:

- Real-time attendance dashboard
- Daily/weekly/monthly reports
- Export data to CSV
- Anomaly detection (multiple punches, unusual locations)

5. Development Roadmap

Phase 1: Setup & Foundation (Week 1-2)

- [ ] Project setup with Expo CLI
- [ ] Backend API structure with Express.js
- [ ] Database setup with PostgreSQL
- [ ] Basic authentication system
- [ ] Environment configuration

Phase 2: Core Features (Week 3-6)

- [ ] Employee registration & login
- [ ] GPS location services integration
- [ ] Camera functionality for selfies
- [ ] Admin dashboard development
- [ ] Location management system
- [ ] Attendance recording functionality

Phase 3: Testing & Refinement (Week 7-8)

- [ ] Unit and integration testing
- [ ] User acceptance testing
- [ ] Performance optimization
- [ ] Security audit
- [ ] Bug fixes and improvements

Phase 4: Deployment (Week 9-10)

- [ ] Production environment setup
- [ ] App store preparation
- [ ] Beta testing with real users
- [ ] Final deployment
- [ ] Documentation completion

6. Step-by-Step Development Guide

Step 1: Environment Setup

bash

```
# Install Node.js (v18+)
# Install Expo CLI
npm install -g @expo/cli

# Create new Expo project
npx create-expo-app GeoHRMS --template tabs
cd GeoHRMS

# Install dependencies
npx expo install expo-location expo-camera expo-image-picker
npm install @react-navigation/native @react-navigation/stack
npm install @reduxjs/toolkit react-redux
npm install axios react-native-maps
```

Step 2: Backend Setup

bash

```
# Create backend directory
mkdir backend && cd backend

# Initialize Node.js project
npm init -y

# Install dependencies
npm install express cors helmet morgan dotenv
npm install jsonwebtoken bcryptjs
npm install prisma @prisma/client
npm install multer cloudinary
npm install joi express-rate-limit

# Development dependencies
npm install -D nodemon
```

Step 3: Database Setup

bash

```
# Initialize Prisma
npx prisma init

# Create database schema in schema.prisma
# Run migrations
npx prisma migrate dev --name init
npx prisma generate
```

Step 4: API Development

Create RESTful endpoints:

- `POST /api/auth/login` - Employee login
- `POST /api/auth/register` - Employee registration (admin only)
- `GET /api/locations` - Get approved locations
- `POST /api/locations` - Create location (admin only)
- `POST /api/attendance/punch` - Record attendance
- `GET /api/attendance/records` - Get attendance records
- `GET /api/reports/daily` - Daily attendance report

Step 5: Mobile App Development

Key screens to develop:

1. **Login Screen** - Authentication
2. **Dashboard** - Main employee interface
3. **Punch Screen** - Location + selfie capture
4. **History Screen** - Personal attendance history
5. **Admin Dashboard** - Management interface
6. **Settings Screen** - App configuration

Step 6: Testing Strategy

- **Unit Tests:** Jest for backend, React Native Testing Library for frontend
- **Integration Tests:** API endpoint testing with Supertest
- **E2E Tests:** Detox for mobile app testing
- **Manual Testing:** Device testing on iOS/Android

7. Deployment Guide

7.1 Backend Deployment (Railway.app)

1. Create Railway account
2. Connect GitHub repository
3. Configure environment variables
4. Deploy with automatic CI/CD

7.2 Mobile App Deployment

Development Testing:

```bash
# Start Expo development server
npx expo start

# Test on physical device with Expo Go app
```

Production Build:

```bash
# Build for Android
npx expo build:android

# Build for iOS (requires Apple Developer account)
npx expo build:ios
```

8. Security Considerations

8.1 Data Protection

- JWT tokens with short expiry
- Password hashing with bcrypt (12 rounds)
- Input validation and sanitization
- SQL injection prevention with Prisma ORM
- Rate limiting on API endpoints

8.2 Location Privacy

- Location data encrypted at rest
- Minimal location data retention
- User consent for location tracking
- GDPR compliance considerations

8.3 Image Security

- Secure image upload to Cloudinary
- Image size and format validation
- Malware scanning for uploaded files
- Automatic image optimization

9. Future Roadmap (Stage 2)

Leave Management System Features:

- Leave request submission
- Approval workflow
- Leave balance tracking
- Calendar integration
- Notification system
- Leave policy configuration

10. Cost Analysis

Free Tier Limitations:

- **Railway.app:** 500 hours/month, 1GB RAM
- **Cloudinary:** 25GB storage, 25k transformations
- **Supabase:** 500MB database, 50k monthly active users
- **Google Maps:** $200 free monthly credit

Scaling Costs (When needed):

- **Railway Pro:** $5/month
- **Cloudinary Plus:** $89/month
- **App Store fees:** $99/year (iOS), $25 one-time (Android)

11. Risk Assessment & Mitigation

Technical Risks:

- **GPS accuracy issues:** Implement multiple location verification methods
- **Battery drain:** Optimize location polling frequency
- **Network connectivity:** Add offline functionality with sync

Business Risks:

- **User adoption:** Conduct user training sessions
- **Data compliance:** Implement GDPR/local privacy law compliance
- **Scalability:** Design with horizontal scaling in mind

12. Success Metrics

Technical KPIs:

- App crash rate < 1%

- API response time < 500ms
- Location accuracy > 95%
- Uptime > 99.5%

Business KPIs:

- User adoption rate
- Daily active users
- Attendance compliance rate
- Admin approval efficiency

13. Support & Maintenance

Documentation Requirements:

- API documentation with Swagger
- User manual for employees
- Admin guide for system management
- Technical documentation for future developers

Maintenance Plan:

- Weekly security updates
- Monthly feature updates
- Quarterly performance reviews
- Annual technology stack assessment

Next Steps:

1. Review and approve this document
2. Set up development environment
3. Begin Phase 1 development
4. Schedule weekly progress reviews
5. Plan user testing sessions

Contact Information:

- Technical Lead: [To be assigned]
- Project Manager: [To be assigned]
- QA Lead: [To be assigned]

Document Version: 1.0

Last Updated: June 21, 2025

Next Review: July 21, 2025