Product Requirements Document (PRD)

HR & Field Activity Management System

Version: 1.0
Date: November 30, 2025
Technology Stack: Laravel PHP (Full-Stack)
Document Status: Draft for Development

1. EXECUTIVE SUMMARY

1.1 Product Overview

A comprehensive HR and field activity management system designed for organizations with hybrid workforces (office and field staff). The system enforces location-based attendance tracking and integrates field activity management with project management systems.

1.2 Business Objectives

- Ensure accurate time tracking with location verification
- Streamline field activity management (workshops and group sessions)
- Automate leave and absence management
- Provide seamless integration with external project management systems
- Enable real-time monitoring of planned vs. implemented activities

1.3 Target Users

- **Field Workers**: Activity facilitators, trainers, social workers
- **Office Staff**: Administrative personnel, managers
- **HR Administrators**: System managers, HR personnel
- **Management**: Directors, supervisors, project managers

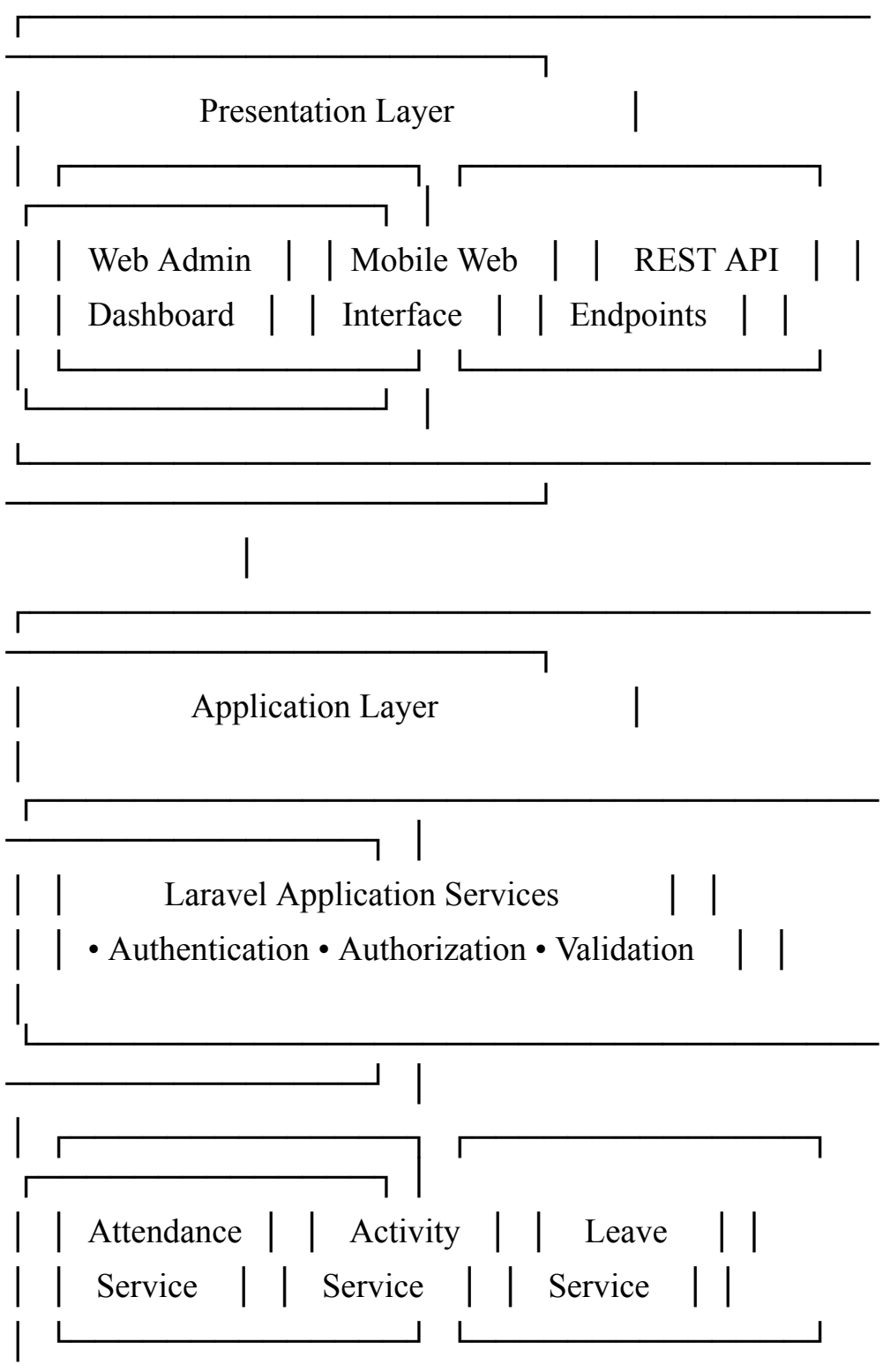
2. SYSTEM ARCHITECTURE

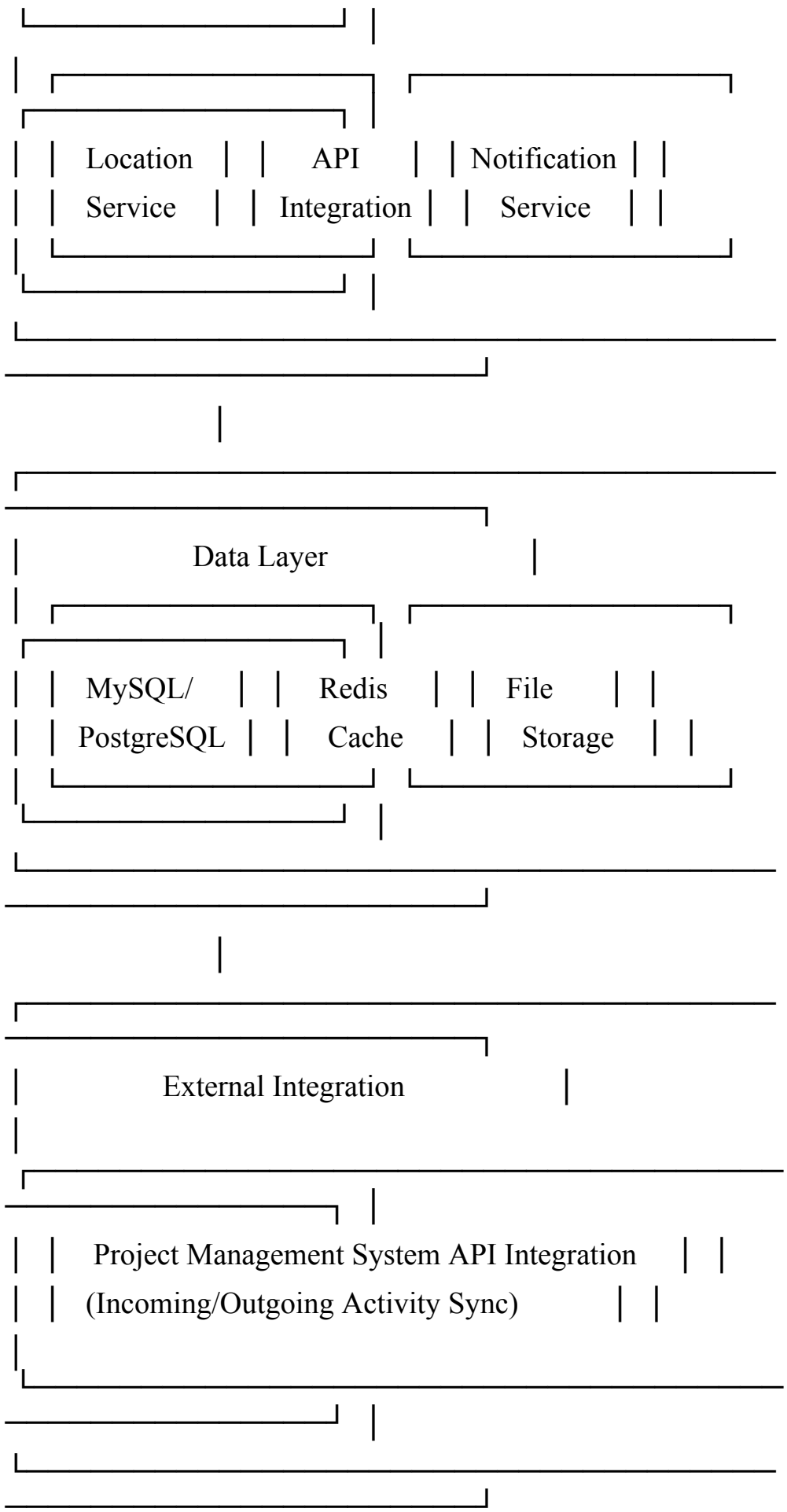
2.1 Technology Stack

- **Backend Framework**: Laravel 11.x (PHP 8.2+)
- **Frontend**: Laravel Blade + Alpine.js/Livewire

- **Database**: MySQL 8.0+ / PostgreSQL 14+
- **Authentication**: Laravel Sanctum (API) + Laravel Breeze/Jetstream (Web)
- **Queue System**: Redis + Laravel Horizon
- **Cache**: Redis
- **Storage**: Local/S3-compatible storage
- **API Documentation**: Laravel Scribe / OpenAPI 3.0

2.2 High-Level Architecture




Location Service
API Integration
Notification Service
Data Layer
MySQL/ PostgreSQL
Redis Cache
File Storage
External Integration
Project Management System API Integration
(Incoming/Outgoing Activity Sync)

3. CORE FEATURES & REQUIREMENTS

3.1 User Management & Authentication

3.1.1 User Roles

Priority: P0 (Critical)

Role	Permissions
Super Admin	Full system access, configuration management
HR Admin	User management, attendance oversight, leave approval
Manager	Team oversight, activity approval, reporting
Field Worker	Sign in/out, activity logging, leave requests
Office Staff	Sign in/out from office locations

3.1.2 Authentication Requirements

- Multi-factor authentication (optional per organization)
- Password policies (minimum 8 chars, complexity rules)
- Session timeout: 12 hours for web, 30 days for mobile
- OAuth2 support for API integration
- Remember device functionality for trusted devices

3.1.3 User Profile

- Full name, employee ID, email, phone
- Department, position, employment type
- Assigned locations (multiple)
- Work schedule definition
- Profile photo
- Emergency contact information

3.2 Location Management

3.2.1 Location Definition

Priority: P0 (Critical)

Location Types:

1. **Office Locations**: Fixed physical offices
2. **Field Locations**: Community centers, schools, public spaces
3. **Administrative Locations**: Home office for admin work

Location Attributes:

- Location name and code

- Location type
- Physical address
- GPS coordinates (latitude, longitude)
- Geofence radius (configurable: 50m - 500m)
- Operating hours
- Assigned employees
- Status (active/inactive)
- Contact person
- Special instructions

3.2.2 Geofencing Logic

Validation Rules:

1. Calculate distance between user GPS and location GPS
2. If distance <= geofence_radius: VALID
3. If distance > geofence_radius: INVALID
4. Allow manual override by HR Admin with reason
5. Log all validation attempts for audit

3.2.3 Location Assignment

- Employees can be assigned to multiple locations
- Assignment includes:
  - Primary location (default)
  - Secondary locations (optional)
  - Valid date range (from/to)
  - Days of week applicable
  - Activity types allowed at location

3.3 Attendance & Sign-In System

3.3.1 Sign-In Process

Priority: P0 (Critical)

Sign-In Flow:

1. User initiates sign-in (mobile/web)
2. System captures GPS coordinates
3. System validates against assigned locations
4. If valid location:

- Create attendance record
- Set status: "signed_in"
- Record timestamp, location, GPS

5. If invalid location:
- Show error: "Outside authorized location"
- Option to select location from assigned list
- Re-validate GPS

6. Confirm sign-in to user

Sign-In Data Capture:

- Employee ID
- Timestamp (with timezone)
- Location ID
- GPS coordinates (latitude, longitude)
- GPS accuracy (in meters)
- Sign-in type (office/field/admin)
- Device information
- IP address
- Photo (optional, for verification)

3.3.2 Sign-Out Process

Priority: P0 (Critical)

Sign-Out Flow:

1. User initiates sign-out
2. System captures GPS coordinates
3. Validate location (same as sign-in or any assigned location)
4. If valid:
- Update attendance record
- Set status: "signed_out"
- Calculate total hours

5. If invalid:
- Allow sign-out with flag for review

6. Prompt for end-of-day notes (optional)

3.3.3 Attendance Validation Rules

- Cannot sign in if already signed in
- Cannot sign out if not signed in
- Maximum work hours per day: 16 hours (configurable)
- Minimum work hours per day: 0.5 hours (configurable)
- Auto sign-out after 16 hours if not manually signed out
- Break time tracking (optional)

3.3.4 Attendance Records

Database Schema:

attendances:
- id (primary key)
- employee_id (foreign key)
- location_id (foreign key)
- sign_in_time (timestamp)
- sign_in_gps_lat (decimal)
- sign_in_gps_lng (decimal)
- sign_in_gps_accuracy (integer)
- sign_in_notes (text)
- sign_out_time (timestamp, nullable)
- sign_out_gps_lat (decimal, nullable)
- sign_out_gps_lng (decimal, nullable)
- sign_out_gps_accuracy (integer, nullable)
- sign_out_notes (text, nullable)
- total_hours (decimal, calculated)
- status (enum: signed_in, signed_out, flagged)
- requires_review (boolean)
- review_notes (text)
- reviewed_by (foreign key, nullable)
- reviewed_at (timestamp, nullable)
- created_at, updated_at

3.4 Field Activity Management

3.4.1 Activity Types

1. Workshops (One-Time Activities) Priority: P0 (Critical)

Characteristics:

- Single session event
- One-time occurrence
- Specific date and time
- Target audience: children OR adults
- Attendance tracking for participants
- Pre-planned or ad-hoc

Workshop Attributes:

- Title and description
- Date and time (start/end)
- Location
- Target audience (children/adults)
- Planned participant count
- Actual participant count
- Facilitator(s) assigned
- Activity category/theme
- Materials/resources needed
- Status (planned/in-progress/completed/cancelled)
- Photos/documentation
- Participant attendance list

2. Groups (Recurring Session Series) Priority: P0 (Critical)

Characteristics:

- Multiple sessions with same participants
- Same group throughout the series
- Scheduled over time period
- Consistent target audience
- Session-by-session tracking

Group Attributes:

- Group name and code
- Description and objectives
- Target audience (children/adults)
- Registered participants (fixed list)

- Start date and end date
- Total planned sessions
- Session schedule (weekly, bi-weekly, etc.)
- Location(s)
- Assigned facilitator(s)
- Status (active/completed/on-hold/cancelled)

Group Session Attributes:

- Session number (e.g., Session 1 of 12)
- Planned date and time
- Actual date and time
- Location
- Session topic/theme
- Participant attendance tracking
- Session notes/observations
- Photos/documentation
- Status (scheduled/completed/cancelled/rescheduled)

3. Administrative Hours Priority: P1 (High)

Characteristics:

- Non-field time tracking
- Office work, planning, meetings, training
- Location: office or home
- No participant involvement

Admin Hours Attributes:

- Date and time (start/end)
- Activity type (meeting/planning/training/documentation/other)
- Description
- Location
- Related project (optional)
- Status (planned/completed)

3.4.2 Activity Planning & Scheduling

Planning Process:

1. Create activity in system:

- Select type (workshop/group/admin)
- Define basic information
- Set location and schedule
- Assign facilitators

2. For Groups:
- Define all sessions in advance
- Register participants
- Set recurring schedule

3. For Workshops:
- Set single date/time
- Estimate participants

4. Approval workflow (optional):
- Submit for manager approval
- Manager reviews and approves/rejects
- Approved activities appear in calendar

3.4.3 Activity Implementation

Implementation Flow:

1. Facilitator signs in at location
2. Facilitator starts activity:
- Select activity from scheduled list
- Confirm location and time
- Mark as "in progress"

3. During activity:
- Track participant attendance
- Add real-time notes
- Capture photos (optional)

4. Complete activity:
- Record actual end time

- Final participant count
- Session summary/notes
- Upload documentation
- Mark as "completed"

5. Facilitator signs out

3.4.4 Participant Management

For Workshops:

- Quick attendance: count only
- Optional: participant name list
- Age group and demographics (optional)

For Groups:

- Fixed participant roster
- Participant profiles:
  - Name, age/birthdate
  - Contact information
  - Emergency contact
  - Special needs/notes
- Session-by-session attendance:
  - Present/absent/late
  - Participation notes
  - Progress tracking

3.4.5 Activity Database Schema

activities:
- id (primary key)
- type (enum: workshop, group, admin_hours)
- title (string)
- description (text)
- category_id (foreign key)
- target_audience (enum: children, adults, mixed, none)
- location_id (foreign key)
- status (enum: planned, in_progress, completed, cancelled)
- planned_start (datetime)

- planned_end (datetime)
- actual_start (datetime, nullable)
- actual_end (datetime, nullable)
- created_by (foreign key)
- approved_by (foreign key, nullable)
- approved_at (timestamp, nullable)
- created_at, updated_at

workshops (extends activities):
- activity_id (foreign key)
- planned_participant_count (integer)
- actual_participant_count (integer)
- materials_needed (text)

groups (extends activities):
- activity_id (foreign key)
- group_code (string, unique)
- total_sessions (integer)
- completed_sessions (integer)
- recurrence_pattern (json)
- start_date (date)
- end_date (date)

group_sessions:
- id (primary key)
- group_id (foreign key)
- session_number (integer)
- session_topic (string)
- planned_date (datetime)
- actual_date (datetime, nullable)
- location_id (foreign key)
- status (enum: scheduled, completed, cancelled, rescheduled)
- notes (text)

- created_at, updated_at

admin_hours:

- activity_id (foreign key)
- admin_type (enum: meeting, planning, training, documentation, other)
- related_project_id (foreign key, nullable)

participants:

- id (primary key)
- name (string)
- age (integer, nullable)
- birthdate (date, nullable)
- gender (enum, nullable)
- contact_phone (string, nullable)
- emergency_contact (string, nullable)
- special_needs (text, nullable)
- created_at, updated_at

group_participants:

- id (primary key)
- group_id (foreign key)
- participant_id (foreign key)
- enrollment_date (date)
- status (enum: active, withdrawn, completed)
- created_at, updated_at

session_attendance:

- id (primary key)
- session_id (foreign key)
- participant_id (foreign key)
- status (enum: present, absent, late, excused)
- notes (text, nullable)

- created_at, updated_at

activity_facilitators:
- id (primary key)
- activity_id (foreign key)
- employee_id (foreign key)
- role (enum: lead, assistant)
- created_at, updated_at

activity_media:
- id (primary key)
- activity_id (foreign key)
- session_id (foreign key, nullable)
- file_path (string)
- file_type (enum: photo, document, video)
- description (text, nullable)
- uploaded_by (foreign key)
- created_at, updated_at

3.5 Leave & Absence Management

3.5.1 Leave Types

Priority: P0 (Critical)

Standard Leave Types:

1. **Annual Leave** (Vacation)
    - Balance-based
    - Accrual rules
    - Carry-over policy
2. **Sick Leave**
    - Balance-based or unlimited
    - Medical certificate requirement (configurable days)
3. **Emergency Leave**
    - Limited per year
    - Immediate approval option

4. **Unpaid Leave**
    - No balance required
    - Deducted from salary
5. **Compensatory Time Off**
    - Earned from overtime
    - Time-limited usage
6. **Maternity/Paternity Leave**
    - Fixed duration
    - Legal compliance
7. **Custom Leave Types**
    - Configurable by organization

3.5.2 Leave Request Workflow

Request Process:

1. Employee submits request:
   - Leave type
   - Start date and end date
   - Number of days
   - Reason/notes
   - Attach documents (if needed)

2. System validation:
   - Check leave balance
   - Check blackout dates
   - Check existing approved leaves
   - Check minimum notice period

3. Approval routing:
   - Direct manager (required)
   - HR review (for long leaves)
   - Department head (optional)

4. Notification:

- Email/in-app to employee
- Update calendar
- Adjust leave balance

5. Status tracking:
- Pending → Approved/Rejected
- Can cancel (before start date)

3.5.3 Leave Balance Management

Balance Tracking:

- Opening balance (start of year)
- Accrued during year
- Taken (approved leaves)
- Pending (requested not approved)
- Available balance
- Expired/carried forward

Accrual Rules:

- Monthly accrual (e.g., 1.75 days/month)
- Pro-rata for new employees
- Accrual caps
- Automatic calculation via scheduled jobs

3.5.4 Absence Tracking

Types of Absences:

1. **Approved Leaves**: Pre-approved time off
2. **Unapproved Absences**: No-show without approval
3. **Late Arrivals**: Sign-in after scheduled time
4. **Early Departures**: Sign-out before scheduled time
5. **Partial Day Absence**: Missing hours

Absence Recording:

- Automatic detection from attendance
- Manual entry by HR/Manager
- Absence reason required
- Disciplinary flag (if applicable)

3.5.5 Leave Database Schema

leave_types:
- id (primary key)
- name (string)
- code (string, unique)
- requires_balance (boolean)
- requires_document (boolean)
- document_required_after_days (integer)
- max_days_per_request (integer, nullable)
- min_notice_days (integer)
- color_code (string)
- is_active (boolean)
- created_at, updated_at

leave_balances:
- id (primary key)
- employee_id (foreign key)
- leave_type_id (foreign key)
- year (integer)
- opening_balance (decimal)
- accrued (decimal)
- taken (decimal)
- pending (decimal)
- expired (decimal)
- carried_forward (decimal)
- current_balance (decimal, calculated)
- updated_at

leave_requests:
- id (primary key)
- employee_id (foreign key)
- leave_type_id (foreign key)
- start_date (date)
- end_date (date)

- total_days (decimal)
- reason (text)
- status (enum: pending, approved, rejected, cancelled)
- requested_at (timestamp)
- reviewed_by (foreign key, nullable)
- reviewed_at (timestamp, nullable)
- review_notes (text, nullable)
- created_at, updated_at

leave_documents:
- id (primary key)
- leave_request_id (foreign key)
- file_path (string)
- file_name (string)
- uploaded_at (timestamp)

absences:
- id (primary key)
- employee_id (foreign key)
- absence_date (date)
- absence_type (enum: no_show, late, early_departure, partial)
- scheduled_time (time, nullable)
- actual_time (time, nullable)
- hours_missed (decimal)
- reason (text, nullable)
- is_approved (boolean)
- disciplinary_action (boolean)
- notes (text, nullable)
- recorded_by (foreign key)
- created_at, updated_at

3.6 API Integration Layer

3.6.1 Integration Purpose

Priority: P0 (Critical)

Enable bidirectional synchronization between HR system and external project management systems for:

- Activity planning data (incoming)
- Activity implementation data (outgoing)
- Real-time status updates
- Resource allocation
- Reporting and analytics

3.6.2 RESTful API Design

Authentication:

- OAuth 2.0 Bearer Token
- API Key + Secret (alternative)
- Rate limiting: 1000 requests/hour per key
- IP whitelisting (optional)

Base URL Structure:

https://your-domain.com/api/v1/

API Endpoints:

1. Activity Management

POST /api/v1/activities

GET /api/v1/activities

GET /api/v1/activities/{id}

PUT /api/v1/activities/{id}

DELETE /api/v1/activities/{id}

POST /api/v1/activities/{id}/start

POST /api/v1/activities/{id}/complete

2. Group Management

POST /api/v1/groups

GET /api/v1/groups

GET /api/v1/groups/{id}

PUT /api/v1/groups/{id}

GET /api/v1/groups/{id}/sessions

POST /api/v1/groups/{id}/sessions

PUT /api/v1/groups/{groupId}/sessions/{sessionId}

POST /api/v1/sessions/{id}/attendance

3. Employee Data

GET /api/v1/employees

GET /api/v1/employees/{id}

GET /api/v1/employees/{id}/attendance

GET /api/v1/employees/{id}/activities

4. Attendance

POST /api/v1/attendance/sign-in

POST /api/v1/attendance/sign-out

GET /api/v1/attendance

GET /api/v1/attendance/daily

5. Leave Management

POST /api/v1/leave-requests

GET /api/v1/leave-requests

GET /api/v1/leave-requests/{id}

PUT /api/v1/leave-requests/{id}/approve

PUT /api/v1/leave-requests/{id}/reject

GET /api/v1/employees/{id}/leave-balance

6. Locations

GET /api/v1/locations

GET /api/v1/locations/{id}

POST /api/v1/locations

PUT /api/v1/locations/{id}

3.6.3 Webhook Support

Outbound Webhooks (from HR system):

- Activity completed
- Activity cancelled
- Employee signed in
- Employee signed out
- Leave request submitted
- Leave request approved/rejected

Webhook Payload Structure:

{

```
 "event": "activity.completed",
 "timestamp": "2025-11-30T10:30:00Z",
 "data": {
   "activity_id": 123,
   "type": "workshop",
   "title": "Youth Leadership Workshop",
   "actual_start": "2025-11-30T09:00:00Z",
   "actual_end": "2025-11-30T12:00:00Z",
   "location_id": 5,
   "facilitators": [45, 67],
   "participant_count": 25
 }
}
```


Webhook Configuration:

- URL endpoint (configurable)
- Authentication header
- Retry logic (3 attempts)
- Failure notification

3.6.4 Integration Scenarios

Scenario 1: Import Planned Activities

External PM System → HR System

1. PM system creates activity plan
2. PM system calls: POST /api/v1/activities
3. HR system validates and creates activity
4. Returns activity_id and status
5. Activity appears in HR system calendar

Scenario 2: Update Implementation Status

HR System → External PM System

1. Facilitator completes activity in HR system
2. HR system triggers webhook: activity.completed
3. PM system receives webhook

4. PM system updates project status
5. PM system confirms receipt

Scenario 3: Sync Attendance Data

Bidirectional Sync

1. PM system requests attendance: GET /api/v1/attendance/daily
2. HR system returns attendance records
3. PM system updates resource allocation
4. PM system sends updated schedule
5. HR system creates new planned activities

3.6.5 API Documentation

Auto-generated documentation:

- Laravel Scribe for endpoint documentation
- Interactive API explorer (Swagger UI)
- Code examples in multiple languages
- Authentication guide
- Webhook setup guide
- Error code reference

3.6.6 Data Synchronization

Sync Strategy:

- Real-time for critical data (sign-in/out)
- Batch sync for reports (daily at midnight)
- On-demand sync via API calls
- Conflict resolution: last-write-wins with audit log

Sync Monitoring:

- Sync status dashboard
- Failed sync alerts
- Data integrity checks
- Sync history log

4. REPORTING & ANALYTICS

4.1 Standard Reports

Priority: P1 (High)

1. Attendance Reports

- Daily attendance summary
- Employee attendance history
- Location-wise attendance
- Late arrivals and early departures
- Missing sign-out report
- Overtime hours report

2. Activity Reports

- Planned vs. implemented activities
- Activity completion rate by facilitator
- Participant reach (by demographics)
- Location utilization
- Activity type distribution
- Group session completion rates

3. Leave Reports

- Leave balance summary (all employees)
- Leave taken by type
- Leave requests pending approval
- Absence pattern analysis
- Leave forecast (upcoming)

4. Performance Reports

- Employee productivity (activities per day)
- Facilitator performance metrics
- Location efficiency
- Time utilization (field vs. admin)

5. Compliance Reports

- GPS validation failures
- Attendance exceptions requiring review
- Unapproved absences
- Overdue leave requests

4.2 Dashboard Widgets

Employee Dashboard:

- Today's schedule
- Upcoming activities
- Leave balance
- Attendance summary (current month)
- Pending tasks

Manager Dashboard:

- Team attendance today
- Activities in progress
- Pending approvals
- Team performance metrics
- Location status

HR Admin Dashboard:

- Organization-wide attendance
- Leave request queue
- Compliance alerts
- System health indicators
- Activity statistics

4.3 Export Functionality

- Export to Excel (XLSX)
- Export to PDF
- Export to CSV
- Email scheduled reports
- API export endpoints

5. TECHNICAL SPECIFICATIONS

5.1 Database Design Principles

Normalization:

- 3rd Normal Form (3NF)
- Avoid data redundancy
- Use foreign keys with proper constraints

Indexing Strategy:

```
-- Critical indexes for performance
CREATE INDEX idx_attendances_employee_date ON
```

attendances(employee_id, sign_in_time);

CREATE INDEX idx_activities_location_date ON activities(location_id, planned_start);

CREATE INDEX idx_leave_requests_status ON leave_requests(employee_id, status);

CREATE INDEX idx_employees_status ON employees(status, department_id);

Soft Deletes:

- Use deleted_at timestamp
- Preserve data for audit
- Allow data recovery

5.2 Laravel Implementation Details

Models & Relationships:

```
// Example: Employee Model
class Employee extends Model
{
  use HasFactory, SoftDeletes;

  protected $fillable = [
    'employee_code', 'first_name', 'last_name',
    'email', 'phone', 'department_id', 'position'
  ];

  // Relationships
  public function department() {
    return $this->belongsTo(Department::class);
  }

  public function attendances() {
    return $this->hasMany(Attendance::class);
  }

  public function activities() {
```

```
        return $this->belongsToMany(Activity::class,
'activity_facilitators');
    }

    public function locations() {
        return $this->belongsToMany(Location::class,
'employee_locations');
    }

    public function leaveRequests() {
        return $this->hasMany(LeaveRequest::class);
    }

    // Scopes
    public function scopeActive($query) {
        return $query->where('status', 'active');
    }

    // Accessors
    public function getFullNameAttribute() {
        return "{$this->first_name} {$this->last_name}";
    }
}
```

Service Layer Pattern:

```
// LocationValidationService
class LocationValidationService
{
    public function validateLocation(
        float $userLat,
        float $userLng,
        Location $location
    ): array {
        $distance = $this->calculateDistance(
```

```
            $userLat, $userLng,
            $location->latitude, $location->longitude
        );

        $isValid = $distance <= $location->geofence_radius;

        return [
            'valid' => $isValid,
            'distance' => round($distance, 2),
            'location_name' => $location->name,
            'threshold' => $location->geofence_radius
        ];
    }

    private function calculateDistance($lat1, $lng1, $lat2, $lng2):
float
    {
        // Haversine formula implementation
        $earthRadius = 6371000; // meters

        $dLat = deg2rad($lat2 - $lat1);
        $dLng = deg2rad($lng2 - $lng1);

        $a = sin($dLat/2) * sin($dLat/2) +
             cos(deg2rad($lat1)) * cos(deg2rad($lat2)) *
             sin($dLng/2) * sin($dLng/2);

        $c = 2 * atan2(sqrt($a), sqrt(1-$a));

        return $earthRadius * $c;
    }
}
```

Repository Pattern:

```
// AttendanceRepository
class AttendanceRepository
{
  protected $model;

  public function __construct(Attendance $model) {
    $this->model = $model;
  }

  public function signIn(array $data): Attendance {
    return $this->model->create($data);
  }

  public function findActiveAttendance(int $employeeId): ?
Attendance {
    return $this->model
      ->where('employee_id', $employeeId)
      ->whereNull('sign_out_time')
      ->latest()
      ->first();
  }

  public function signOut(Attendance $attendance, array $data):
bool {
    $attendance->update($data);
    $attendance->update([
```