**HRGenie - Complete Documentation**

**Table of Contents**

1. System Overview
2. Architecture
3. Frontend Application
4. Backend API
5. AI Features
6. Database Schema
7. Deployment
8. API Endpoints
9. Workflow
10. Security & Privacy

**System Overview**

HRGenie is a comprehensive Human Resource Management System (HRMS) built with modern web technologies. It provides end-to-end HR functionality including employee management, attendance tracking, payroll processing, performance reviews, recruitment, and leave management.

**Key Features:**

* Multi-role access (Admin, HR, Manager, Employee)
* Real-time attendance tracking with geolocation
* Automated payroll processing
* Performance management with bias detection
* Recruitment pipeline management
* Leave management system
* Advanced analytics and reporting
* AI-powered insights and bias mitigation

**Technology Stack:**

* **Frontend:** Next.js 14, React, TypeScript, Tailwind CSS
* **Backend:** Node.js, Express.js, TypeScript
* **Database:** PostgreSQL with Supabase
* **Authentication:** Supabase Auth with JWT
* **AI/ML:** Custom bias detection algorithms
* **Deployment:** Vercel (frontend & backend)
* **Styling:** Framer Motion, Lucide Icons

**Architecture**

**System Architecture Diagram**

text

┌─────────────────┐ ┌──────────────────┐ ┌─────────────────┐

│ Frontend │ │ Backend API │ │ Database │

│ (Next.js) │◄──►│ (Express.js) │◄──►│ (Supabase) │

│ │ │ │ │ │

│ - Dashboard │ │ - REST API │ │ - PostgreSQL │

│ - Employee Mgmt │ │ - Auth Middleware│ │ - Row Level │

│ - Attendance │ │ - Business Logic │ │ Security │

│ - Payroll │ │ - AI Services │ │ - Real-time │

│ - Performance │ │ - File Handling │ │ Subscriptions │

└─────────────────┘ └──────────────────┘ └─────────────────┘

│ │ │

│ │ │

┌─────────────────┐ ┌──────────────────┐ ┌─────────────────┐

│ Authentication │ │ File Storage │ │ AI Services │

│ (Supabase) │ │ (Supabase) │ │ (Custom) │

│ │ │ │ │ │

│ - JWT Tokens │ │ - Resumes │ │ - Bias Detection│

│ - Role Management│ │ - Payslips │ │ - Analytics │

│ - Social Login │ │ - Documents │ │ - Predictions │

└─────────────────┘ └──────────────────┘ └─────────────────┘

**Data Flow**

1. **Client Request** → Next.js Frontend
2. **API Call** → Express.js Backend with JWT validation
3. **Database Operation** → Supabase PostgreSQL with RLS
4. **AI Processing** → Custom algorithms for bias detection
5. **Response** → Filtered data based on user role

**Frontend Application**

**Structure**

text

src/

├── components/

│ ├── page-components/ # Main page components

│ │ ├── DashboardPage.tsx

│ │ ├── EmployeesPage.tsx

│ │ ├── AttendancePage.tsx

│ │ ├── PayrollPage.tsx

│ │ ├── PerformancePage.tsx

│ │ ├── RecruitmentPage.tsx

│ │ └── LeavesPage.tsx

│ ├── ui/ # Reusable UI components

│ ├── providers/ # Context providers

│ └── three/ # 3D background components

├── hooks/ # Custom React hooks

├── types/ # TypeScript type definitions

└── utils/ # Utility functions

**Key Components**

**DashboardLayout**

* Responsive layout with collapsible sidebar
* Role-based menu filtering
* Theme support (light/dark/system)
* Mobile-first design

typescript

// Role-based menu filtering

const getFilteredMenuItems = () => {

const rolePermissions = {

admin: allMenuItems,

hr: ['dashboard', 'employees', 'recruitment', 'payroll', 'leaves', 'attendance', 'performance'],

manager: ['dashboard', 'employees', 'payroll', 'leaves', 'attendance', 'performance'],

employee: ['dashboard', 'payroll', 'leaves', 'attendance', 'performance']

}

return menuItems.filter(item => rolePermissions[user.role].includes(item.id))

}

**PerformancePage**

* Multi-view interface (Employee, Manager, Admin)
* Real-time performance metrics
* Bias detection and mitigation
* Goal tracking and reviews

**RecruitmentPage**

* Job opening management
* Candidate pipeline with Kanban view
* Application scoring system
* Bulk operations

**State Management**

* React hooks for local state
* Supabase for real-time data
* Context API for theme and auth
* Local storage for persistence

**Styling & UX**

* Tailwind CSS for utility-first styling
* Framer Motion for animations
* Lucide React for icons
* Responsive design with mobile support
* Accessibility-compliant components

**Backend API**

**Architecture**

text

src/

├── controllers/ # Route handlers

│ ├── attendanceController.ts

│ ├── authController.ts

│ ├── payrollController.ts

│ ├── performanceController.ts

│ └── recruitmentController.ts

├── middleware/ # Custom middleware

│ ├── auth.ts

│ └── validation.ts

├── routes/ # API routes

├── types/ # TypeScript types

└── utils/ # Helper functions

**Key Controllers**

**AttendanceController**

typescript

// Features:

- Clock in/out with location tracking

- Shift-based attendance calculation

- Regularization requests

- Monthly attendance summaries

- Manager and admin views

**AuthController**

typescript

// Features:

- Supabase authentication integration

- JWT token generation

- Role-based access control

- Social login (Google)

- Profile management

**PerformanceController**

typescript

// Features:

- Goal management

- Performance reviews

- 360-degree feedback

- Bias detection algorithms

- Skills assessment

**Authentication Flow**

1. User login via Supabase Auth
2. JWT token generation with role claims
3. Token validation on each API request
4. Role-based route protection
5. Automatic token refresh

**AI Features**

**Bias Detection System**

**Types of Bias Detected:**

* Recency Bias (overweighting recent events)
* Similarity Bias (favoring similar individuals)
* Central Tendency (avoiding extremes)
* Leniency/Strictness Bias
* Halo/Horns Effect

**Implementation:**

typescript

// Bias detection in performance reviews

const detectBias = (reviewData: ReviewData): BiasAlert[] => {

const alerts: BiasAlert[] = [];

// Check for recency bias

if (hasRecencyBias(reviewData.recentEvents, reviewData.historicalPerformance)) {

alerts.push({

type: 'recency\_bias',

severity: 'high',

details: 'Review seems overly influenced by recent events'

});

}

// Check for similarity bias

if (hasSimilarityBias(reviewData.reviewer, reviewData.reviewee)) {

alerts.push({

type: 'similarity\_bias',

severity: 'medium',

details: 'Potential similarity bias detected'

});

}

return alerts;

}

**AI-Powered Analytics**

* Performance trend prediction
* Attrition risk assessment
* Recruitment success forecasting
* Workforce planning insights

**Database Schema**

**Core Tables**

**Employees Table**

sql

CREATE TABLE employees (

id uuid PRIMARY KEY DEFAULT uuid\_generate\_v4(),

uid text UNIQUE, -- Supabase Auth ID

name text NOT NULL,

email text NOT NULL UNIQUE,

phone text,

department text,

position text,

role text NOT NULL DEFAULT 'employee',

manager\_id uuid REFERENCES employees(id),

status text DEFAULT 'active',

created\_at timestamp DEFAULT now(),

updated\_at timestamp DEFAULT now()

);

**Attendance Records**

sql

CREATE TABLE attendance\_records (

id uuid PRIMARY KEY DEFAULT uuid\_generate\_v4(),

employee\_id uuid REFERENCES employees(id),

attendance\_date date NOT NULL,

clock\_in\_time timestamptz,

clock\_out\_time timestamptz,

hours\_worked numeric,

status text CHECK (status IN ('Present', 'Absent', 'Late', ...)),

work\_location text,

created\_at timestamptz DEFAULT now()

);

**Performance Management**

sql

CREATE TABLE reviews (

id uuid PRIMARY KEY DEFAULT uuid\_generate\_v4(),

employee\_id uuid REFERENCES employees(id),

overall\_rating numeric,

summary text,

status text DEFAULT 'pending',

finalized\_at timestamptz

);

CREATE TABLE bias\_alerts (

id uuid PRIMARY KEY DEFAULT uuid\_generate\_v4(),

review\_id uuid REFERENCES reviews(id),

detected\_bias\_type text NOT NULL,

details text,

status text DEFAULT 'new'

);

**Relationships**

* **Employees** → Self-referencing for manager hierarchy
* **Attendance** → Many-to-one with Employees
* **Performance** → Complex relationships with Reviews, Goals, Feedback
* **Payroll** → Separate audit trail and line items

**Deployment**

**Frontend Deployment (Vercel)**

* **URL:** <https://hr-genie-jrta.vercel.app/>
* **Framework:** Next.js 14
* **Features:** Automatic CI/CD, edge optimization, serverless functions

**Backend Deployment (Vercel)**

* **URL:** <https://hr-genie-u2cc.vercel.app/>
* **Runtime:** Node.js Express server
* **API Base:** /api/\* routes

**Environment Variables**

env

# Frontend

NEXT\_PUBLIC\_SUPABASE\_URL=your\_supabase\_url

NEXT\_PUBLIC\_SUPABASE\_ANON\_KEY=your\_anon\_key

NEXT\_PUBLIC\_BACKEND\_URL=your\_backend\_url

# Backend

SUPABASE\_URL=your\_supabase\_url

SUPABASE\_SERVICE\_ROLE\_KEY=your\_service\_role\_key

JWT\_SECRET=your\_jwt\_secret

**Database (Supabase)**

* **PostgreSQL** with Row Level Security (RLS)
* **Authentication** built-in
* **Storage** for files and documents
* **Real-time** subscriptions

**API Endpoints**

**Authentication**

text

POST /api/auth/login # User login

POST /api/auth/register # User registration

POST /api/auth/google # Google OAuth

GET /api/auth/user # Current user profile

**Attendance**

text

GET /api/attendance/employee-view # Employee attendance dashboard

GET /api/attendance/manager-view # Team attendance overview

GET /api/attendance/admin-view # Organization-wide attendance

POST /api/attendance/clock-in # Clock in with location

POST /api/attendance/clock-out # Clock out

POST /api/attendance/regularize # Submit regularization request

POST /api/attendance/approve-regularization # Approve/reject requests

**Performance**

text

GET /api/performance/employee-view # Employee performance dashboard

GET /api/performance/manager-view # Team performance management

GET /api/performance/admin-view # Admin performance overview

POST /api/performance/goals # Create/update goals

POST /api/performance/reviews # Submit performance reviews

POST /api/performance/feedback # Give feedback

POST /api/performance/bias-mitigation # Log bias mitigation actions

**Recruitment**

text

GET /api/recruitment/jobs # List job openings

POST /api/recruitment/jobs # Create new job

GET /api/recruitment/applications # List applications with filters

POST /api/recruitment/applications # Submit application

POST /api/recruitment/applications/:id/status # Update application status

POST /api/recruitment/applications/bulk-update # Bulk status update

GET /api/recruitment/applications/export # Export applications

**Payroll**

text

GET /api/payroll/runs # List payroll runs

POST /api/payroll/runs # Create new payroll run

GET /api/payroll/runs/:id # Get payroll run details

POST /api/payroll/runs/:id/process # Process payroll run

GET /api/payroll/payslips/:id # Download payslip

**Workflow**

**Employee Daily Workflow**

1. **Clock In** → Location-based attendance tracking
2. **Task Management** → View assigned goals and tasks
3. **Leave Requests** → Apply for leaves with documentation
4. **Performance Tracking** → Update goal progress
5. **Clock Out** → Automatic hours calculation

**Manager Workflow**

1. **Team Overview** → Monitor team attendance and performance
2. **Approval Workflow** → Process leave and regularization requests
3. **Performance Management** → Conduct reviews and provide feedback
4. **Resource Planning** → Team capacity and workload management

**HR Admin Workflow**

1. **Employee Lifecycle** → Onboarding to offboarding
2. **Payroll Processing** → Monthly payroll runs and compliance
3. **Recruitment** → Job posting to candidate onboarding
4. **Analytics** → Organization-wide reporting and insights
5. **System Management** → Role and policy configuration

**Recruitment Pipeline**

text

Applied → Screening → Interview → Offer → Hired

↓ ↓ ↓ ↓ ↓

Submit Resume Technical HR Onboarding

Screen Interview Offer

**Security & Privacy**

**Authentication & Authorization**

* **Supabase Auth** with JWT tokens
* **Role-based access control** (Admin, HR, Manager, Employee)
* **Route protection** at both frontend and backend levels
* **Session management** with secure token storage

**Data Protection**

typescript

// Row Level Security in Supabase

CREATE POLICY "Employees can view own data" ON employees

FOR SELECT USING (auth.uid() = id);

CREATE POLICY "Managers can view team data" ON employees

FOR SELECT USING (manager\_id = auth.uid() OR id = auth.uid());

CREATE POLICY "HR can view all employee data" ON employees

FOR SELECT USING (

EXISTS (SELECT 1 FROM employees WHERE id = auth.uid() AND role IN ('hr', 'admin'))

);

**Privacy Features**

* **Data encryption** at rest and in transit
* **GDPR compliance** for personal data
* **Audit trails** for sensitive operations
* **Data retention policies**
* **Secure file storage** for documents

**Security Measures**

* **CORS configuration** for API security
* **Input validation** and sanitization
* **SQL injection prevention** with parameterized queries
* **XSS protection** with React's built-in escaping
* **Rate limiting** on authentication endpoints

**Compliance**

* **Payroll data** protection with financial compliance
* **Employee data** privacy following labor laws
* **Audit logs** for all financial transactions
* **Data export** capabilities for regulatory requirements