



# **Smart Attendance Tracker with Geofencing**

## **A Technical Report**

CHAVA AJAY  
CH.EN.U4AIE22009

# Smart Attendance Tracker with Geofencing

## Executive Summary

The Smart Attendance Tracker with Geofencing is a modern web application that helps organizations manage employee attendance using location-based verification. This system makes sure employees can only mark their attendance when they are physically present at the correct workplace location.

## What the System Does

The system uses GPS technology to create virtual boundaries around work locations. Employees can only check in or check out when they are inside these boundaries. This prevents fake attendance marking from remote locations.

## Key Features

### For Employees:

- Login and register with secure accounts
- Mark attendance only when at the right location
- View personal attendance history
- Read company announcements

### For Admins:

- Create and manage work location boundaries
- View all employee attendance records
- Post company announcements
- See user statistics and reports

## Technology Overview

The system is built using modern web technologies:

- **Frontend:** React with TypeScript for the user interface
- **Backend:** Node.js with Express for server operations
- **Database:** MongoDB for storing data
- **Security:** JWT tokens for secure login

# **1. Introduction**

## **Problem Statement**

Traditional attendance systems have many problems. Paper-based systems can be lost or faked. Basic digital systems allow employees to mark attendance from anywhere, even from home. This leads to dishonest attendance marking and makes it hard for managers to track who is actually at work.

Location-aware attendance systems solve these problems by using GPS to verify that employees are physically present at the workplace before they can mark attendance.

## **Project Goals**

This project aims to create a system that:

- Provides secure attendance marking with location verification
- Restricts attendance to specific geographic areas (geofences)
- Gives administrators full oversight of attendance data
- Allows easy communication through announcements
- Works on both desktop and mobile devices

## **Project Scope**

The system assumes that:

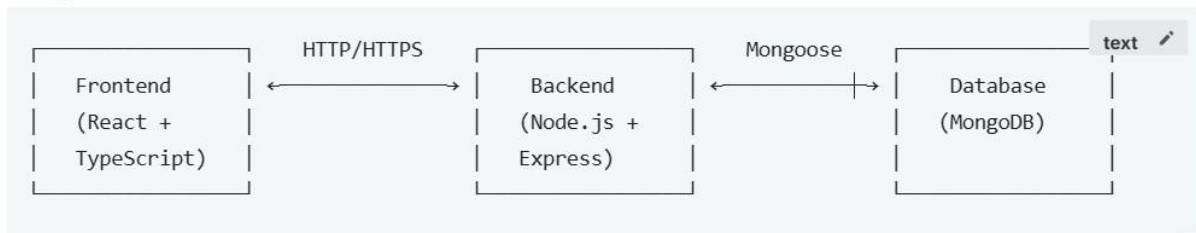
- Users have devices with GPS capabilities
- Internet connection is available
- Web browsers support location services
- Users grant location permissions to the application

---

## 2. System Overview

### High-Level Architecture

The system follows a three-tier architecture:



#### Frontend Components:

- User interface built with React
- TypeScript for better code quality
- Responsive design for all devices
- Location services integration

#### Backend Components:

- Express server handling API requests
- JWT authentication system
- Geofencing validation logic
- RESTful API endpoints

#### Database:

- MongoDB for flexible data storage
- User accounts and roles
- Attendance records
- Geofence definitions
- Announcements

## Deployment Modes

### Development Mode:

- Frontend runs on React development server
- Backend runs on separate Express server
- CORS enabled for cross-origin requests

### Production Mode:

- Express server serves built React application
  - Single server deployment
  - Optimized for performance
- 

## 3. Functional Requirements

### Employee Functions

#### Account Management:

- Register new account with email and password
- Login with secure JWT token authentication
- View and update profile information

#### Attendance Management:

- Get current GPS location using browser
- Select from available geofenced locations
- Mark check-in only when inside the geofence boundary
- Mark check-out from the same location
- View personal attendance history with dates and times

#### Communication:

- Read latest company announcements
- View announcements by priority level

#### Administrator Functions

#### User Management:

- Access admin dashboard with role-based permissions
- View list of all users in the system
- See user statistics grouped by role

#### Geofence Management:

- Create new geofenced locations with GPS coordinates
- Set radius for each location (in meters)

- Update existing geofence details
- Deactivate geofences (soft delete)
- View list of all geofences

#### **Attendance Oversight:**

- View all employee attendance records
- Filter attendance by date, user, or location
- Export attendance data for reporting
- View attendance statistics

#### **Communication Management:**

- Create new announcements
- Set priority levels (low, medium, high)
- Edit existing announcements
- Deactivate old announcements

---

## **4. Non-Functional Requirements**

### **Security Requirements**

#### **Authentication:**

- JWT token-based authentication
- Password hashing using bcrypt
- Secure token storage in browser
- Automatic logout on token expiry

#### **Authorization:**

- Role-based access control
- Protected routes for authenticated users
- Admin-only sections and functions
- Server-side permission validation

### **Reliability Requirements**

- Server-side validation for all inputs
- Error handling for network issues
- Clear error messages for users
- Data integrity checks

### **Usability Requirements**

- Modern, clean user interface
- Responsive design for mobile devices
- Loading indicators for better user experience
- Success and error message notifications

- Easy navigation between sections

## **Performance Requirements**

- Fast database queries with pagination
- Efficient location processing
- Quick authentication response
- Smooth user interface interactions

## **Maintainability Requirements**

- Modular code structure
- Separate controllers, routes, and models
- TypeScript for type safety
- Clear documentation and comments

---

# **5. Detailed Design and Implementation**

## **5.1 Frontend Implementation**

### **Technology Stack:**

- React 18 for user interface
- TypeScript for type safety
- Axios for API communication
- React Router for navigation

### **Application Structure:**

The main app routing is handled in `App.tsx` with protected routes that require authentication:

```

function App() {
  return (
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/login" element={<Login />} />
      <Route path="/register" element={<Register />} />
      <Route path="/employee" element={
        <ProtectedRoute>
          <EmployeeDashboard />
        </ProtectedRoute>
      } />
      <Route path="/admin" element={
        <ProtectedRoute>
          <AdminDashboard />
        </ProtectedRoute>
      } />
    </Routes>
  );
}

```

### Authentication Context:

Global authentication state is managed through `AuthContext.tsx` which provides:

- User login function
- User registration function
- Logout function
- Current user information
- Loading states

### API Client Configuration:

The `api.ts` file sets up axios with:

- Base URL configuration
- Authorization header injection
- Automatic token refresh
- Error handling and redirects

### Key Pages:

1. **Home Page:** Landing page with system information
2. **Login/Register:** Authentication forms with role selection
3. **Employee Dashboard:**
  - Location services integration
  - Geofence selection
  - Check-in/check-out workflow
  - Attendance history view
  - Announcements display



#### 4. Admin Dashboard:

- User statistics overview ○
- Geofence management ○
- Attendance logs review ○
- Announcement management

**5.2**

## Backend Implementation

### Server Setup:

The main server file `server.js` handles:

- Environment configuration
- Database connection
- Middleware setup (CORS, JSON parsing)
- Route mounting
- Production build serving **Authentication Middleware:**

The `auth.js` middleware provides:

```
// Protect routes - verify JWT token
const protect = async (req, res, next) => {
  // Extract token from Authorization header
  // Verify token with JWT secret
  // Attach user to request object
  // Continue to next middleware
};

// Admin only access
const adminOnly = (req, res, next) => {
  // Check if user role is admin
  // Allow access or return forbidden error
};

// Generate JWT token
const generateToken = (id) => {
  return jwt.sign({ id }, process.env.JWT_SECRET, {
    expiresIn: '30d'
  });
};
```

### Database Models:

### User Model:

```
{
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true }, // bcrypt hashed
  role: { type: String, enum: ['admin', 'employee'], default: 'employee' },
  createdAt: { type: Date, default: Date.now }
}
```

### Geofence Model:

```
{
  name: { type: String, required: true },
  center: {
    latitude: { type: Number, required: true },
    longitude: { type: Number, required: true }
  },
  radius: { type: Number, required: true }, // in meters
  description: String,
  isActive: { type: Boolean, default: true },
  createdAt: { type: Date, default: Date.now }
}
```

### Attendance Model:

```
{
  userId: { type: ObjectId, ref: 'User', required: true },
  geofenceId: { type: ObjectId, ref: 'Geofence', required: true },
  status: { type: String, enum: ['check-in', 'check-out'], required: true },
  location: {
    latitude: { type: Number, required: true },
    longitude: { type: Number, required: true }
  },
  timestamp: { type: Date, default: Date.now },
  notes: String
}
```

### Announcement Model:

```

{
  title: { type: String, required: true },
  message: { type: String, required: true },
  priority: { type: String, enum: ['low', 'medium', 'high'], default: 'medium' },
  createdBy: { type: ObjectId, ref: 'User', required: true },
  isActive: { type: Boolean, default: true },
  createdAt: { type: Date, default: Date.now }
}

```

## 5.3 Geofencing Logic Implementation

### Frontend Location Services:

The employee dashboard uses browser geolocation APIs:

```

// Get current position
navigator.geolocation.getCurrentPosition(
  (position) => {
    const { latitude, longitude } = position.coords;
    setUserLocation({ latitude, longitude });
  },
  (error) => {
    console.error('Location error:', error);
  },
  { enableHighAccuracy: true, timeout: 10000 }
);

// Watch position changes
const watchId = navigator.geolocation.watchPosition(
  (position) => {
    // Update user location in real-time
  }
);

```

### Backend Validation:

The attendance controller performs server-side geofence validation:

```
const markAttendance = async (req, res) => {
  const { geofenceId, status, location, notes } = req.body;

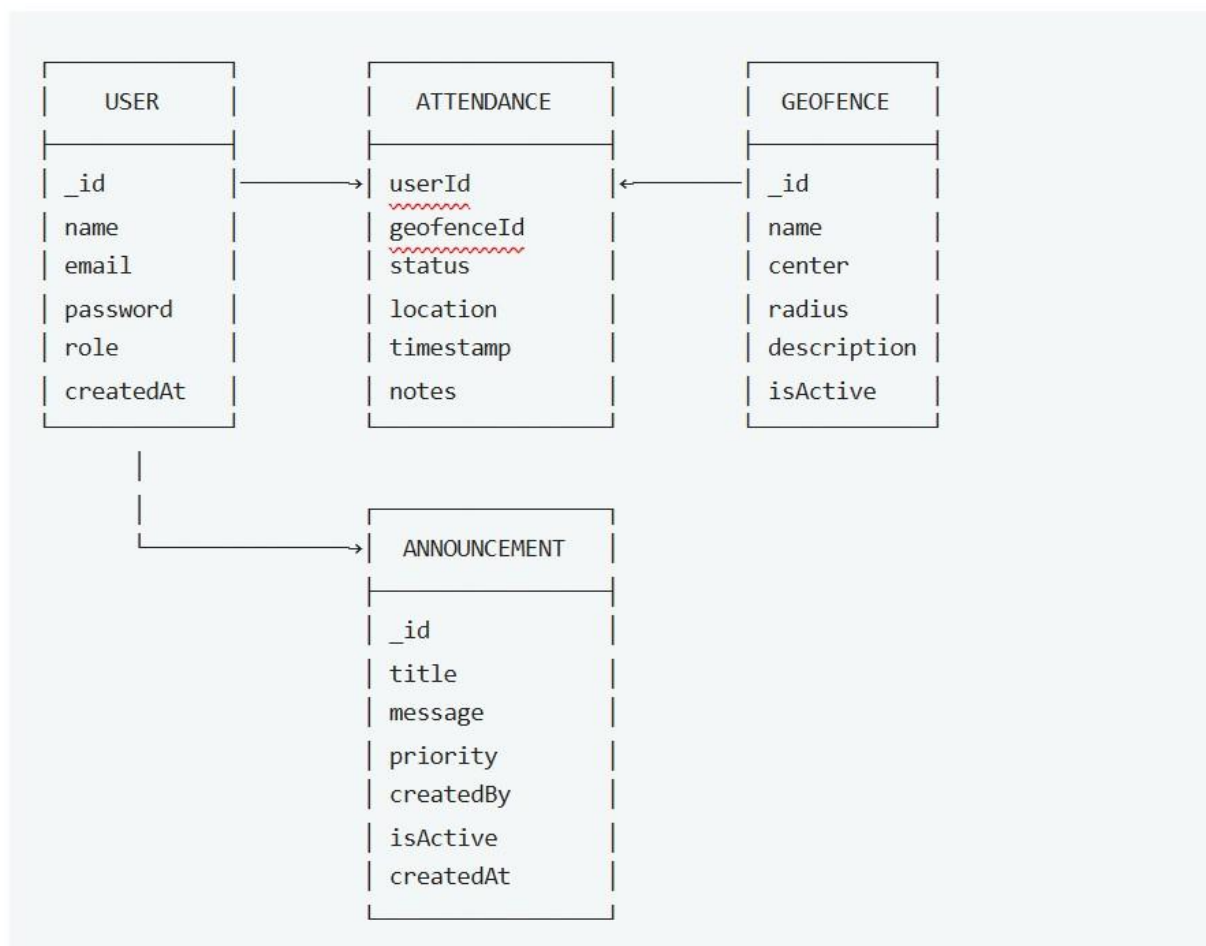
  // Get geofence details
  const geofence = await Geofence.findById(geofenceId);

  // Check if user is within geofence
  const isWithinGeofence = geolib.isPointWithinRadius(
    { latitude: location.latitude, longitude: location.longitude },
    { latitude: geofence.center.latitude, longitude: geofence.center.longitude },
    geofence.radius
  );
};
```

javascript

## 6. Data Model and Relationships

### Entity Relationship Diagram



### Relationship Types

- User to Attendance:** One-to-Many (One user can have many attendance records)

- **Geofence to Attendance:** One-to-Many (One geofence can have many attendance records)
- **User to Announcement:** One-to-Many (One admin user can create many announcements)

---

## 7. API Endpoints Documentation

### Authentication Endpoints (/api/auth)

Method	Endpoint	Access	Description
POST	/register	Public	Register new user account
POST	/login	Public	Login user and get JWT token
GET	/me	Protected	Get current user profile
GET	/users	Admin Only	Get all users list
GET	/count	Admin Only	Get user count by role

#### Example Request/Response:

```
// POST /api/auth/login
Request: {
  "email": "user@example.com",
  "password": "password123"
}

Response: {
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "_id": "64f1234567890abcdef12345",
    "name": "John Doe",
    "email": "user@example.com",
    "role": "employee"
  }
}
```

## Geofence Endpoints ( [/api/geofence](#) )

Method	Endpoint	Access	Description
GET	/	Protected	Get all active geofences
GET	/:id	Protected	Get specific geofence
POST	/	Admin Only	Create new geofence
PUT	/:id	Admin Only	Update geofence
DELETE	/:id	Admin Only	Soft delete geofence

## Attendance Endpoints ( [/api/attendance](#) )

Method	Endpoint	Access	Description
POST	/mark	Protected	Mark attendance (check-in/out)
GET	/my-history	Protected	Get user's attendance history
GET	/all	Admin Only	Get all attendance records
GET	/stats	Admin Only	Get attendance statistics

## Announcement Endpoints ( [/api/announcements](#) )

Method	Endpoint	Access	Description
GET	/	Protected	Get all active announcements
GET	/:id	Protected	Get specific announcement
POST	/	Admin Only	Create new announcement
PUT	/:id	Admin Only	Update announcement
DELETE	/:id	Admin Only	Soft delete announcement

---

## 8. End-to-End User Flows

### Employee Check-in Flow

Employee → Browser Location → Select Geofence → Validate Position → Mark Attendance → Database → UI Update

#### Step-by-Step Process:

1. Employee opens dashboard
2. Browser requests location permission
3. GPS coordinates are obtained
4. System shows available geofences
5. Employee selects target geofence
6. System validates if employee is within boundary
7. If valid, check-in button becomes enabled
8. Employee clicks check-in
9. API call sent to backend with location data
10. Server validates location again
11. Attendance record created in database
12. Success message shown to employee

### Admin Geofence Creation Flow

Admin Dashboard → Geofence Form → GPS Coordinates → Set Radius → Save → Database → Available to Employees

#### Step-by-Step Process:

1. Admin accesses geofence management section
2. Clicks "Create New Geofence" button
3. Fills out geofence form:
  - Name and description
  - GPS coordinates (latitude/longitude)
  - Radius in meters
4. Submits form
5. Backend validates admin permissions
6. Geofence saved to database
7. Immediately available for employee check-ins



## User Registration and Login Flow

Registration Form → Server Validation → Password Hashing → Database → JWT Token → Dashboard Access

---

## 9. Security Implementation

### Authentication Security

#### JWT Token Management:

- Tokens expire after 30 days
- Stored securely in browser localStorage
- Automatically included in API requests
- Server validates token on each protected request

#### Password Security:

- All passwords hashed using bcrypt
- Minimum password requirements enforced
- No plain text password storage

### Authorization Security

#### Role-Based Access Control:

- Middleware checks user roles on each request
- Admin-only functions protected at API level
- Frontend also enforces role-based UI

#### API Security:

- CORS configured for development and production
- Input validation on all endpoints
- Error messages don't reveal sensitive information

### Data Protection

- Geolocation data encrypted in transit
  - User personal information protected
  - Attendance records linked to user IDs only
-



## 10. Setup and Deployment Guide

### Prerequisites

- Node.js version 14 or higher
- MongoDB database (local or cloud)
- Modern web browser with location services

### Backend Configuration

Create `.env` file in backend directory:

```
PORT=5000
MONGODB_URI=mongodb://localhost:27017/smart-attendance-tracker
JWT_SECRET=your_secure_secret_key_here
NODE_ENV=development
```

### Frontend Configuration

Create `.env` file in frontend directory:

```
REACT_APP_API_URL=http://localhost:5000/api
```

### Development Setup Commands Backend

Setup:

```
cd backend
npm install
npm run dev
```

### Frontend Setup:

```
cd frontend
npm install
npm start
```

### Production Deployment

Build Process:

```
# Build frontend
cd frontend
npm run build

# Start production server
cd ../backend
npm start
```

### **Production Recommendations:**

- Use process manager like PM2
  - Set up reverse proxy with nginx
  - Enable SSL/TLS certificates
  - Use environment variables for sensitive data
  - Set up database backups
  - Monitor server performance
- 

## **11. Testing and Validation**

### **Manual Testing Scenarios**

#### **Authentication Testing:**

- Valid login credentials → Success
- Invalid credentials → Error message
- Expired token → Redirect to login
- Admin access → Admin dashboard visible
- Employee access → Employee dashboard only

#### **Geofencing Testing:**

- Inside geofence boundary → Check-in enabled
- Outside geofence boundary → Check-in disabled
- Accurate GPS signal → Precise validation
- Weak GPS signal → Error handling

#### **Attendance Testing:**

- Valid check-in → Attendance record created
- Duplicate check-in → Error prevented
- Check-out without check-in → Error handled
- Check-in from outside geofence → Blocked

#### **Admin Function Testing:**

- Create geofence → Available to employees
- Update geofence → Changes reflected
- Delete geofence → Soft deleted (hidden)

- View attendance → All records visible

## Automated Testing

The frontend includes testing framework setup with `react-scripts test` for:

- Component rendering tests
  - User interaction tests
  - API integration tests
  - Form validation tests
- 

# 12. User Interface and Experience

## Design Principles

### Modern Aesthetics:

- Gradient backgrounds and glass effects
- Clean typography and spacing
- Consistent color scheme
- Professional appearance

### Responsive Design:

- Works on desktop computers
- Optimized for tablets
- Mobile-friendly interface
- Touch-friendly buttons

### User Experience Features:

- Loading spinners during API calls
- Success and error notifications
- Clear navigation breadcrumbs
- Helpful tooltips and hints

## Key Interface Screenshots

### Employee Dashboard Sections:

- Welcome banner with user information
  - Location status and GPS indicator
  - Available geofences with distance
  - Check-in/check-out buttons
  - Recent attendance history table
  - Company announcements panel
- ### Admin Dashboard Sections:

- Overview with statistics cards
  - User management table with roles
  - Geofence management with map view
  - Attendance logs with filtering
  - Announcement creation and editing
- 

## **13. System Limitations and Future Enhancements**

### **Current Limitations**

#### **Technical Limitations:**

- Requires internet connection for all functions
- Depends on GPS accuracy of user device
- Browser compatibility for location services
- Single geofence shape (circles only)

#### **Functional Limitations:**

- No offline attendance capability
- Limited reporting and analytics
- No integration with payroll systems
- Basic announcement system

### **Recommended Future Enhancements**

#### **Mobile Applications:**

- Native iOS and Android apps
- Better GPS accuracy and battery optimization
- Push notifications for announcements
- Offline attendance with sync capability

#### **Advanced Features:**

- Polygon-shaped geofences for complex areas
- Dynamic geofence radius based on GPS accuracy
- Photo verification with check-in
- Biometric authentication integration

#### **Analytics and Reporting:**

- Advanced attendance analytics
- Exportable reports (PDF, Excel)
- Attendance trends and patterns
- Integration with HR systems

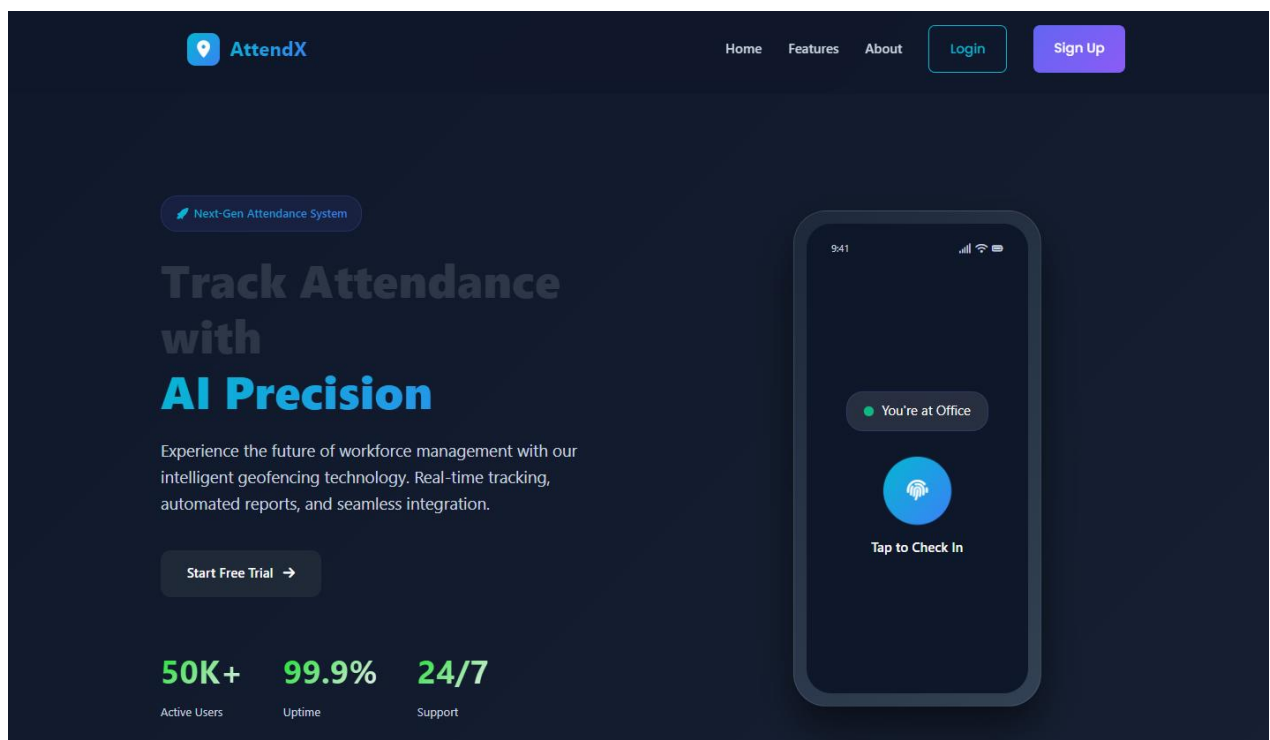
## System Improvements:

- Single Sign-On (SSO) integration
- Multi-language support
- Audit logs for admin actions
- Automated backup and recovery

---

## 14. Final Output Images

### Home Page:



## Login Page:


### GeoAttend


- GPS-based attendance tracking
- Real-time monitoring
- Analytics & reporting
- Secure & tamper-proof system

[← Back to Home](#)

## Welcome Back

Sign in to your account to continue

Employee

Administrator

Email Address

Enter your email

Password

Enter your password

Sign In as Employee

[Forgot password?](#) | [Sign up here](#)

Or continue with

## Register Page:

### GeoAttend

- GPS-based attendance tracking
- Real-time monitoring
- Analytics & reporting
- Secure & tamper-proof system

## Create Account

Get started with your free account

Full Name

Enter your full name

Email Address

Enter your email

Role

Employee

Password

Enter your password

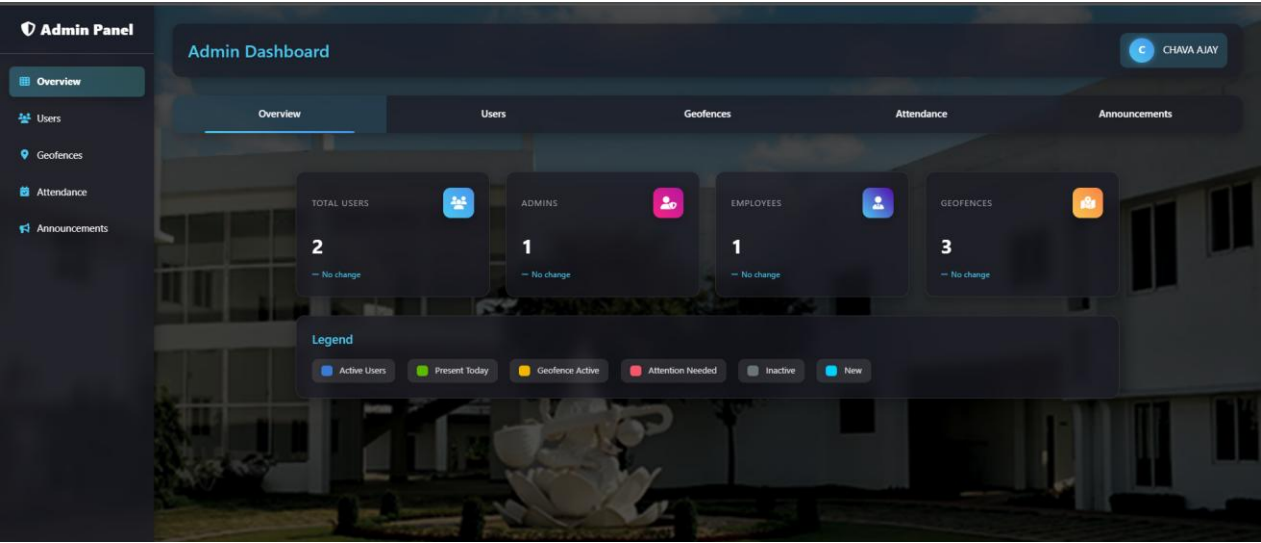
Confirm Password

Confirm your password

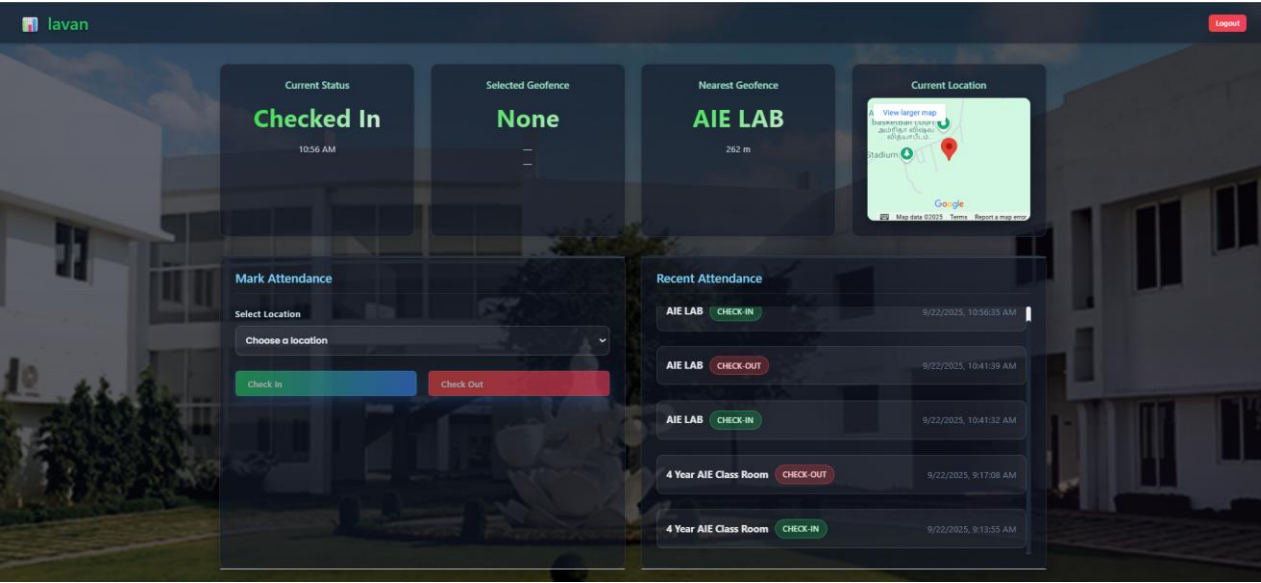
Create Account

Already have an account? [Sign in here](#)

AdminDashboard Page:



EmployeeDashboard Page:



## 15. Conclusion:

The Smart Attendance Tracker with Geofencing successfully addresses the critical need for accurate, location-based attendance management in modern organizations. By combining robust web technologies with GPS-based verification, the system provides a reliable solution for preventing attendance fraud while maintaining ease of use.

### Technical Achievement

The project demonstrates successful implementation of:

- Full-stack web development using MERN stack
- Real-time geolocation processing and validation
- Secure authentication and role-based authorization
- Modern, responsive user interface design
- RESTful API architecture with proper error handling

### Business Value

The system delivers significant value through:

- Enhanced attendance accuracy and fraud prevention
- Streamlined administrative processes
- Improved employee accountability
- Cost-effective deployment and maintenance
- Scalable architecture for organizational growth

### Future Potential

With the foundation established, the system provides an excellent platform for future enhancements including mobile applications, advanced analytics, and integration with existing enterprise systems. The modular architecture and modern technology stack ensure that the system can evolve with changing organizational needs.

This project represents a successful merger of location technology with traditional attendance management, creating a solution that is both technically sound and practically valuable for organizations seeking to modernize their workforce management processes.

---