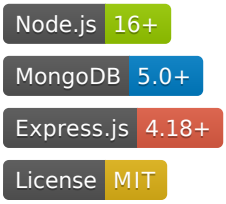
SmartPay HRMS - HR Payroll Management System



📋 Table of Contents

🎯 Overview

SmartPay HRMS is a comprehensive web-based Human Resource Management and Payroll System built with modern technologies. It streamlines HR operations, automates payroll calculations, and provides detailed reporting capabilities for organizations of all sizes.

PROF

🎪 Key Objectives

- **Streamline HR Operations**: Centralized management of departments, positions, grades, and employees
- **Automate Payroll Processing**: Automated salary calculations with PAYE, pension, and overtime handling
- **Generate Compliance Reports**: PDF payslips, Excel reports, and bank-ready payment files
- **Self-Service Portal**: Employee portal for overtime submission and profile management
- **Data Security**: Secure document storage and role-based access control

✨ Features

👨‍💼 HR/Admin Features

- **Department Management**: Create and manage organizational departments

- **Position Management**: Define positions within departments with grade assignments
- **Grade Structure**: Configure salary grades with automatic tax and pension calculations
- **Employee Onboarding**: Complete employee registration with document upload
- **Dashboard Analytics**: Real-time counts and statistics of all HR metrics
- **Payroll Processing**: One-click monthly payroll generation
- **Report Generation**: Comprehensive PDF and Excel reports
- **Bank Integration**: Generate bank diskette files for salary transfers

👥 Employee Features

- **Profile Management**: View and update personal information
- **Overtime Submission**: Submit monthly overtime hours
- **Payslip Access**: Download personal payslips
- **Document Access**: View uploaded documents

📊 Reporting Features

- **Individual Payslips**: PDF payslips for each employee
- **Payroll Summary**: Monthly payroll reports with detailed breakdowns
- **Bank Diskette**: CSV/Excel files formatted for bank processing
- **Employee Reports**: Comprehensive employee data exports
- **Overtime Reports**: Monthly overtime tracking and analysis

🛠️ Technology Stack

Backend

- **Runtime**: Node.js 16+
- **Framework**: Express.js 4.18+
- **Database**: MongoDB 5.0+
- **ODM**: Mongoose 7.0+
- **Authentication**: express-session + bcryptjs
- **File Upload**: Multer
- **PDF Generation**: Puppeteer / PDFKit
- **Excel Generation**: ExcelJS

PROF

Frontend

- **Template Engine**: EJS (Embedded JavaScript)
- **CSS Framework**: Bootstrap 5.3+
- **JavaScript**: Vanilla JS + jQuery
- **Icons**: Font Awesome 6
- **Charts**: Chart.js (for dashboard analytics)

Development Tools

- **Process Manager**: PM2
- **Linting**: ESLint

- **Code Formatting**: Prettier
- **Version Control**: Git
- **Environment**: dotenv

🏗️ System Architecture

```
┌─────────────────────────────────────────────────────────┐
│                  SmartPay HRMS Architecture             │
├─────────────────────────────────────────────────────────┤
│  Frontend (EJS Templates)                               │
│  ├── Admin Dashboard                                    │
│  ├── Employee Portal                                    │
│  └── Reporting Interface                                │
├─────────────────────────────────────────────────────────┤
│  Backend (Node.js + Express)                            │
│  ├── Authentication Middleware                          │
│  ├── API Routes                                         │
│  ├── Business Logic                                     │
│  └── File Processing                                    │
├─────────────────────────────────────────────────────────┤
│  Database Layer (MongoDB)                               │
│  ├── Users Collection                                   │
│  ├── Departments Collection                             │
│  ├── Positions Collection                               │
│  ├── Grades Collection                                  │
│  ├── Employees Collection                               │
│  └── Payroll Collection                                 │
├─────────────────────────────────────────────────────────┤
│  File System                                            │
│  ├── Document Storage                                   │
│  ├── Generated Reports                                  │
│  └── Temp Files                                         │
└─────────────────────────────────────────────────────────┘
```

PROF

🚀 Installation Guide

Prerequisites

- Node.js 16+ installed
- MongoDB 5.0+ running locally or cloud instance
- Git for version control

Step 1: Clone Repository

```
git clone https://github.com/yourusername/smartpay-hrms.git
cd smartpay-hrms
```

Step 2: Install Dependencies

```
npm install
```

Step 3: Environment Configuration

Create a `.env` file in the root directory:

```
# Server Configuration
PORT=3000
NODE_ENV=development

# Database Configuration
MONGODB_URI=mongodb://localhost:27017/smartpay_hrms
DB_NAME=smartpay_hrms

# Session Configuration
SESSION_SECRET=your-super-secret-session-key-here
SESSION_EXPIRY=7d

# File Upload Configuration
MAX_FILE_SIZE=5MB
UPLOAD_PATH=./uploads

# PDF Generation
PDF_ENGINE=puppeteer

# Email Configuration (Optional)
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your-email@gmail.com
SMTP_PASS=your-app-password

# Default Admin Credentials
ADMIN_EMAIL=admin@smartpay.com
ADMIN_PASSWORD=SmartPay@2025
```

PROF

Step 4: Initialize Database

```
npm run seed
```

Step 5: Start Development Server

```
npm run dev
```

The application will be available at `http://localhost:3000`

Step 6: Production Setup

```
npm run build
npm start
```

Database Schema

Users Collection

```
{
  _id: ObjectId,
  email: String, // Unique
  password: String, // Hashed
  role: String, // 'admin' | 'hr' | 'employee'
  isActive: Boolean,
  createdAt: Date,
  updatedAt: Date
}
```

Departments Collection

```
{
  _id: ObjectId,
  name: String, // Unique
  description: String,
  headOfDepartment: ObjectId, // Reference to Employee
  isActive: Boolean,
  createdAt: Date,
  updatedAt: Date
}
```

Grades Collection

```
{
  _id: ObjectId,
  name: String, // Unique, e.g., "Grade A", "Senior Level"
  baseSalary: Number,
  payeePercent: Number, // PAYE tax percentage
  pensionPercent: Number, // Pension contribution percentage
  overtimeRate: Number, // Hourly overtime rate
  allowances: {
```

```
    transport: Number,
    housing: Number,
    medical: Number,
    other: Number
  },
  isActive: Boolean,
  createdAt: Date,
  updatedAt: Date
}
```

Positions Collection

```
{
  _id: ObjectId,
  name: String, // e.g., "Senior Accountant"
  departmentId: ObjectId, // Reference to Department
  gradeId: ObjectId, // Reference to Grade
  description: String,
  requirements: [String],
  isActive: Boolean,
  createdAt: Date,
  updatedAt: Date
}
```

Employees Collection

```
{
  _id: ObjectId,
  employeeId: String, // Unique employee identifier
  userId: ObjectId, // Reference to User
  personalInfo: {
    firstName: String,
    lastName: String,
    middleName: String,
    dateOfBirth: Date,
    gender: String,
    maritalStatus: String,
    nationality: String,
    phoneNumber: String,
    alternatePhone: String,
    email: String,
    address: {
      street: String,
      city: String,
      state: String,
      zipCode: String,
      country: String
    }
```

PROF

```
  },
  employmentInfo: {
    positionId: ObjectId, // Reference to Position
    department: String, // Denormalized for quick access
    startDate: Date,
    employmentType: String, // 'full-time' | 'part-time' | 'contract'
    status: String // 'active' | 'suspended' | 'terminated'
  },
  bankInfo: {
    bankName: String,
    accountNumber: String,
    accountName: String,
    routingNumber: String
  },
  documents: [{
    name: String,
    type: String, // 'id' | 'passport' | 'resume' | 'contract' | 'other'
    filePath: String,
    uploadDate: Date
  }],
  overtimeRecords: [{
    month: String, // Format: 'YYYY-MM'
    hours: Number,
    submittedDate: Date,
    approvedBy: ObjectId, // Reference to User
    status: String // 'pending' | 'approved' | 'rejected'
  }],
  emergencyContact: {
    name: String,
    relationship: String,
    phoneNumber: String,
    address: String
  },
  createdAt: Date,
  updatedAt: Date
}
```


Payroll Collection

```
{
  _id: ObjectId,
  employeeId: ObjectId, // Reference to Employee
  payrollMonth: String, // Format: 'YYYY-MM'
  salaryDetails: {
    baseSalary: Number,
    allowances: {
      transport: Number,
      housing: Number,
      medical: Number,
      other: Number
```

```
    },
    overtime: {
      hours: Number,
      rate: Number,
      amount: Number
    },
    grossPay: Number
  },
  deductions: {
    paye: {
      percent: Number,
      amount: Number
    },
    pension: {
      percent: Number,
      amount: Number
    },
    other: [{
      name: String,
      amount: Number
    }],
    totalDeductions: Number
  },
  netPay: Number,
  paymentStatus: String, // 'pending' | 'processed' | 'paid'
  paymentDate: Date,
  processedBy: ObjectId, // Reference to User
  createdAt: Date,
  updatedAt: Date
}
```

📡 API Endpoints

Authentication Routes

PROF

```
POST   /auth/login           - User login
POST   /auth/logout          - User logout
GET    /auth/profile         - Get current user profile
PUT    /auth/profile         - Update user profile
```

Department Routes

```
GET    /api/departments      - Get all departments
POST   /api/departments      - Create new department
GET    /api/departments/:id  - Get department by ID
PUT    /api/departments/:id  - Update department
DELETE /api/departments/:id  - Delete department
```

Grade Routes

```
GET    /api/grades           - Get all grades
POST   /api/grades           - Create new grade
GET    /api/grades/:id       - Get grade by ID
PUT    /api/grades/:id       - Update grade
DELETE /api/grades/:id       - Delete grade
```

Position Routes

```
GET    /api/positions        - Get all positions
POST   /api/positions        - Create new position
GET    /api/positions/:id    - Get position by ID
PUT    /api/positions/:id    - Update position
DELETE /api/positions/:id    - Delete position
```

Employee Routes

```
GET    /api/employees        - Get all employees
POST   /api/employees        - Create new employee
GET    /api/employees/:id    - Get employee by ID
PUT    /api/employees/:id    - Update employee
DELETE /api/employees/:id    - Delete employee
POST   /api/employees/:id/documents - Upload employee document
GET    /api/employees/:id/payslips  - Get employee payslips
```

Payroll Routes

PROF

```
GET    /api/payroll          - Get payroll records
POST   /api/payroll/process  - Process monthly payroll
GET    /api/payroll/:month   - Get payroll for specific month
GET    /api/payroll/reports/:month - Generate payroll report
GET    /api/payroll/bankfile/:month - Generate bank diskette
```

Overtime Routes

```
GET    /api/overtime         - Get overtime records
POST   /api/overtime         - Submit overtime hours
PUT    /api/overtime/:id     - Update overtime record
DELETE /api/overtime/:id     - Delete overtime record
```

📅 Development Timeline (2 Weeks)

Week 1: Foundation & Core Features

Day 1: Project Setup & Infrastructure

- [x] Initialize Node.js project with Express
- [x] Set up MongoDB connection with Mongoose
- [x] Configure EJS template engine
- [x] Create basic folder structure
- [x] Set up environment configuration
- [x] Initialize Git repository

Day 2: Authentication System

- [x] Implement user authentication middleware
- [x] Create login/logout functionality
- [x] Set up session management
- [x] Create user roles (admin, hr, employee)
- [x] Implement password hashing with bcrypt

Day 3: Database Models

- [x] Create Mongoose schemas for all collections
- [x] Set up model relationships and validations
- [x] Create database indexes for performance
- [x] Implement model methods and statics

Day 4: Grade Management System

- [x] Create grade CRUD operations
- [x] Implement salary calculation logic
- [x] Set up PAYE and pension percentage handling
- [x] Create overtime rate configuration

PROF

Day 5: Department & Position Management

- [x] Build department management interface
- [x] Create position assignment system
- [x] Link positions to departments and grades
- [x] Implement hierarchical data display

Day 6: Employee Management

- [x] Create comprehensive employee form
- [x] Implement file upload with Multer
- [x] Build employee profile pages

- [x] Set up document management system

Day 7: Admin Dashboard

- [x] Create dashboard with statistics
- [x] Implement data visualization with Chart.js
- [x] Build navigation and user interface
- [x] Add responsive design elements

Week 2: Payroll & Advanced Features

Day 8: Employee Portal

- [x] Create employee login interface
- [x] Build overtime submission form
- [x] Implement monthly overtime tracking
- [x] Create employee profile management

Day 9: Payroll Calculation Engine

- [x] Implement complex payroll algorithms
- [x] Handle base salary, allowances, and overtime
- [x] Calculate PAYE tax and pension deductions
- [x] Create net pay computation

Day 10: Payslip Generation

- [x] Set up PDF generation with Puppeteer
- [x] Design professional payslip template
- [x] Implement batch payslip generation
- [x] Add download and email functionality

PROF

Day 11: Reporting System

- [x] Create Excel export functionality
- [x] Build comprehensive payroll reports
- [x] Implement employee data exports
- [x] Add filtering and search capabilities

Day 12: Bank Integration

- [x] Create bank diskette file generator
- [x] Implement CSV/Excel bank formats
- [x] Add custom bank template support
- [x] Validate bank file formatting

Day 13: Advanced Features

- ☑ Implement audit logging
- ☑ Add data backup functionality
- ☑ Create system settings page
- ☑ Build user management interface

Day 14: Testing & Deployment

- ☑ Comprehensive system testing
- ☑ Performance optimization
- ☑ Security vulnerability assessment
- ☑ Documentation completion
- ☑ Production deployment setup

📖 Usage Guide

For HR Administrators

1. Setting Up the System

1. **Login** with admin credentials
2. **Create Departments**: Add all organizational departments
3. **Define Grades**: Set up salary grades with tax percentages
4. **Create Positions**: Define positions and assign to departments/grades
5. **Add Employees**: Register employees with complete information

2. Managing Payroll

1. **Review Overtime**: Check and approve employee overtime submissions
2. **Process Payroll**: Click "Process Monthly Payroll" button
3. **Generate Reports**: Download payslips and summary reports
4. **Create Bank File**: Generate bank diskette for salary transfers

PROF

3. Generating Reports

- **Individual Payslips**: PDF format for each employee
- **Payroll Summary**: Excel report with all employee details
- **Bank Diskette**: CSV file formatted for bank processing
- **Employee Reports**: Comprehensive employee data exports

For Employees

1. Accessing the Portal

1. **Login** with provided credentials
2. **View Profile**: Check personal and employment information
3. **Submit Overtime**: Enter monthly overtime hours
4. **Download Payslips**: Access current and historical payslips

2. Updating Information

- **Personal Details**: Update contact information
- **Bank Details**: Modify bank account information
- **Emergency Contacts**: Update emergency contact details

📸 Screenshots

Admin Dashboard


Admin Dashboard

Employee Management


Employee Management

Payroll Processing


Payroll Processing

Payslip Generation


Payslip Sample

🧪 Testing

Running Tests

```
# Run all tests
npm test

# Run specific test suites
npm run test:unit
npm run test:integration
npm run test:api

# Generate coverage report
npm run coverage
```

PROF

Test Structure

```
tests/
├── unit/
│   ├── models/
│   ├── controllers/
│   └── utils/
├── integration/
│   ├── auth.test.js
```

```
│   ├── employees.test.js
│   └── payroll.test.js
└── fixtures/
    ├── users.json
    └── employees.json
```

Manual Testing Checklist

Authentication

- [ ] Admin login/logout
- [ ] Employee login/logout
- [ ] Session management
- [ ] Password reset functionality

HR Management

- [ ] Department CRUD operations
- [ ] Grade management with calculations
- [ ] Position assignments
- [ ] Employee registration with file uploads

Payroll Processing

- [ ] Overtime submission and approval
- [ ] Monthly payroll calculation
- [ ] Payslip generation (PDF)
- [ ] Report generation (Excel)
- [ ] Bank file creation (CSV)

🚀 Deployment

PROF

Production Environment Setup

1. Server Requirements

- **OS**: Ubuntu 20.04 LTS or CentOS 8
- **Memory**: Minimum 2GB RAM (4GB recommended)
- **Storage**: 20GB SSD (with growth capacity)
- **Node.js**: Version 16 or higher
- **MongoDB**: Version 5.0 or higher

2. Environment Configuration

```
# Production environment variables
NODE_ENV=production
PORT=80
```

```
MONGODB_URI=mongodb://your-mongo-server:27017/smartpay_hrms
SESSION_SECRET=your-production-secret-key
```

3. Using PM2 for Process Management

```
# Install PM2 globally
npm install -g pm2

# Start application with PM2
pm2 start ecosystem.config.js

# Monitor application
pm2 monit

# Set up auto-restart on server reboot
pm2 startup
pm2 save
```

4. Nginx Configuration

```
server {
    listen 80;
    server_name your-domain.com;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;
    }
}
```

PROF

Docker Deployment

1. Docker Setup

```
FROM node:16-alpine

WORKDIR /app
```

```
COPY package*.json ./
RUN npm ci --only=production

COPY . .

EXPOSE 3000

CMD ["npm", "start"]
```

2. Docker Compose

```
version: '3.8'
services:
  app:
    build: .
    ports:
      - "3000:3000"
    environment:
      - NODE_ENV=production
      - MONGODB_URI=mongodb://mongo:27017/smartpay_hrms
    depends_on:
      - mongo

  mongo:
    image: mongo:5.0
    volumes:
      - mongo_data:/data/db

volumes:
  mongo_data:
```

PROF

Cloud Deployment Options

1. Heroku

```
# Install Heroku CLI
npm install -g heroku

# Login and create app
heroku login
heroku create smartpay-hrms

# Add MongoDB addon
heroku addons:create mongolab:sandbox

# Deploy
git push heroku main
```

2. AWS EC2

- Launch EC2 instance with Ubuntu 20.04
- Install Node.js, MongoDB, and PM2
- Configure security groups for HTTP/HTTPS access
- Set up Load Balancer for high availability

3. Digital Ocean

- Create droplet with pre-configured MEAN stack
- Upload application files
- Configure domain and SSL certificate
- Set up automated backups

🤝 Contributing

We welcome contributions from the community! Please follow these steps:

1. Fork the Repository

```
git clone https://github.com/yourusername/smartpay-hrms.git
cd smartpay-hrms
git remote add upstream https://github.com/original/smartpay-hrms.git
```

2. Create Feature Branch

```
git checkout -b feature/your-feature-name
```

PROF

3. Development Guidelines

- Follow ESLint configuration
- Write comprehensive tests
- Update documentation
- Use conventional commit messages

4. Submit Pull Request

- Ensure all tests pass
- Update CHANGELOG.md
- Provide detailed PR description
- Request code review

Code Style Guidelines

- Use camelCase for variables and functions
- Use PascalCase for classes and constructors
- Include JSDoc comments for functions
- Follow the existing folder structure
- Write meaningful commit messages

📄 License

This project is licensed under the MIT License - see the LICENSE file for details.

📞 Support

For support and questions:

- **Email**: support@smartpay-hrms.com
- **Documentation**: https://docs.smartpay-hrms.com
- **Issues**: GitHub Issues
- **Discussions**: GitHub Discussions

🔗 Useful Links

- Node.js Documentation
- Express.js Guide
- MongoDB Manual
- EJS Documentation
- Bootstrap Components

Built with ❤ by the SmartPay HRMS Team

Last updated: July 2025