README.md 2025-08-10

```
1 / 8
```

Employee Management System Documentation

Table of Contents

Overview

This system provides a comprehensive employee management solution with role-based authentication for administrators and employees. Administrators can perform full CRUD operations on employee records, while employees can view their personal profile information. The system features secure JWT-based authentication with bcrypt password hashing and a modern, responsive UI with glassmorphism design elements.

Tech Stack

- **Frontend**: HTML, CSS, JavaScript
- **Backend**: Node.js with Express
- **Database**: SQLite (local file-based database)
- **Security**: JWT + bcrypt for password hashing
- **UI Framework**: Modern CSS with glassmorphism effects

Features

- **Dual Authentication System**: Separate login for administrators and employees
- **Admin Dashboard**: Complete employee management with CRUD operations
- **Employee Dashboard**: Personal profile view with detailed information
- **Modern UI Design**: Glassmorphism effects with smooth animations
- **Secure Authentication**: JWT tokens with bcrypt password hashing
- **Role-based Access Control**: Different permissions for admin and employee roles
- **Responsive Design**: Works seamlessly on desktop and mobile devices

- **Real-time Form Validation**: Client and server-side input validation
- **SQLite Integration**: Local database with automatic initialization

Setup Instructions

Prerequisites

- **Node.js** (v16+ recommended)
- **npm** (comes with Node.js)
- **Git** (for cloning the repo)

Installation Steps

1. Clone the Repository

```
git clone https://github.com/barath-101/PRODIGY_FS_02.git

cd employee-management
```

2. Install Dependencies

```
npm install
```

3. Environment Configuration

Copy `.env.example` to `.env` and configure:

```
cp .env.example .env
```

Update the configuration in `.env`:

```
# JWT Secret
JWT_SECRET=emp_mgmt_2024_secure_jwt_key_barathisgood_incoding_xyz789

# Server Configuration
PORT=3000
```

4. Database Setup

The SQLite database will be created automatically when you start the server. No manual database setup required!

5. Start the Application

```
npm start
```

- Server runs on **http://localhost:3000**
- Database file `employee_management.db` is created automatically
- Sample admin and employee accounts are created on first run

Usage Guide

Admin Login

1. Navigate to **http://localhost:3000**
2. Select **"Administrator"** role
3. Use default credentials:
    - **Email**: `admin@company.com`
    - **Password**: `admin123`
4. Click **"Login"**
5. Access the admin dashboard with full employee management capabilities

Employee Login

1. Navigate to **http://localhost:3000**
2. Select **"Employee"** role
3. Use default credentials:
    - **Email**: `barathg.work@gmail.com`
    - **Password**: `employee123`
4. Click **"Login"**
5. View your personal profile dashboard

Admin Dashboard Features

1. **View All Employees**: See complete employee list in a modern table
2. **Add New Employee**: Click "Add New Employee" to create new accounts
3. **Edit Employee**: Use "Edit" button to modify employee details
4. **Delete Employee**: Remove employees with "Delete" button
5. **Search & Filter**: Find specific employees quickly
6. **Logout**: Secure session termination

Employee Dashboard Features

1. **Profile Overview**: View personal information and employment details
2. **Contact Information**: See email and phone details
3. **Employment Details**: Check position, department, and salary information
4. **Hire Date**: View employment start date
5. **Status**: Check current employment status

API Endpoints

User Authentication

URL: `/api/login`
Method: POST
Request Body:

```
{
  "email": "admin@company.com",
  "password": "admin123",
  "userType": "admin"
}
```

Response: JWT token and user information

Get All Employees (Admin Only)

URL: `/api/employees`
Method: GET
Headers: `Authorization: Bearer <jwt_token>`
Response: Array of employee objects

Get Single Employee

URL: `/api/employees/:id`
Method: GET
Headers: `Authorization: Bearer <jwt_token>`
Response: Employee information as JSON

Create New Employee (Admin Only)

URL: `/api/employees`
Method: POST
Headers: `Authorization: Bearer <jwt_token>`
Request Body:

```
{
  "firstName": "John",
  "lastName": "Doe",
  "email": "john.doe@company.com",
  "phone": "+1-555-0123",
  "position": "Software Engineer",
  "department": "Engineering",
  "salary": 75000,
  "password": "securepassword"
}
```

Response: Created employee object

Update Employee (Admin Only)

URL: `/api/employees/:id`
Method: PUT
Headers: `Authorization: Bearer <jwt_token>`
Request Body: Employee fields to update
Response: Updated employee object

Delete Employee (Admin Only)

URL: `/api/employees/:id`
Method: DELETE
Headers: `Authorization: Bearer <jwt_token>`
Response: Success confirmation message

Security Implementation

- **Password Hashing**: All passwords are hashed using bcrypt with 10 salt rounds
- **JWT Authentication**: Secure token-based authentication with 24-hour expiration
- **Role-based Access**: Strict separation between admin and employee permissions
- **Input Validation**: Comprehensive client and server-side validation
- **CORS Protection**: API configured with CORS for secure cross-origin requests
- **Environment Variables**: Sensitive configuration stored in environment files
- **SQL Injection Prevention**: Parameterized queries protect against SQL injection

Troubleshooting

- **Server Won't Start**: Check if port 3000 is available or change PORT in .env
- **Login Failures**: Verify email and password match default credentials
- **Database Issues**: Delete `employee_management.db` file to reset database
- **Permission Errors**: Ensure proper role selection during login
- **UI Not Loading**: Clear browser cache and refresh the page
- **API Errors**: Check browser console for detailed error messages

Project Structure

```
employee-management/
├── public/                    # Frontend files
│   ├── login.html             # Login page with role selection
│   ├── admin.html             # Admin dashboard with employee management
│   ├── employee.html          # Employee profile dashboard
│   ├── styles.css             # Modern CSS with glassmorphism effects
│   ├── login.js               # Login authentication logic
│   ├── admin.js               # Admin dashboard functionality
│   └── employee.js            # Employee dashboard logic
├── server.js                  # Express server with API endpoints
├── package.json               # Dependencies and scripts
```

```
├── .env.example              # Environment variables template
├── .env                      # Environment configuration (created by user)
├── employee_management.db    # SQLite database (auto-created)
└── README.md                 # Project documentation
```

This structure separates frontend and backend logic, with the public folder containing all client-side files and server.js handling API endpoints and database interactions. The SQLite database file is automatically created and managed by the application.

Future Enhancements

Short-term Improvements

- **Email Verification**: Add email verification step during employee creation
- **Password Reset**: Implement "Forgot Password" functionality for employees
- **Profile Pictures**: Allow employees to upload and manage profile images
- **Advanced Search**: Add filtering by department, position, and status

Mid-term Goals

- **Bulk Operations**: Import/export employee data via CSV files
- **Reporting Dashboard**: Generate employee reports and analytics
- **Audit Logging**: Track all admin actions for compliance
- **Department Management**: Add department creation and management features

Long-term Vision

- **Multi-tenant Support**: Support multiple organizations in one system
- **Advanced Permissions**: Granular role-based permissions system
- **API Rate Limiting**: Implement rate limiting for enhanced security
- **Real-time Notifications**: Live updates for employee changes

License

LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

For more information, please refer to https://unlicense.org