Employee Management System

A full-stack web application for managing employee records with CRUD operations, built using Express.js, EJS templating, and Bootstrap for styling.

Features

- **User Sign Up**: Create new user accounts with email and password
- **Admin Authentication**: Secure login system with cookie-based session management
- **Dashboard**: Overview of total employees and recent hires
- **CRUD Operations**:
    - Create new employee records
    - Read and display employee information
    - Update existing employee details
    - Delete employee records
- **RESTful API**: Complete API endpoints for programmatic access
- **Responsive UI**: Bootstrap-styled interface with clean design
- **Testing Suite**: Comprehensive Cypress test cases

Technologies Used

- **Backend**: Node.js, Express.js
- **Templating**: EJS (Embedded JavaScript)
- **Styling**: Bootstrap 5, Bootstrap Icons
- **Data Storage**: JSON file (db.json) using fs module
- **Authentication**: Cookie-based sessions with cookie-parser
- **Testing**: Cypress E2E testing

Installation

1. Clone the repository
2. Install dependencies:

```
npm install
```

Running the Application

Start the server:

```
npm start
```

The application will run on http://localhost:3000

Login Credentials

User Login

- Access at /login
- Use any employee email with their password
- Non-admin users with completed registration → /directory
- Admin users with completed registration → /admin/dashboard
- Users without completed registration → Error message

Admin Login

- Access at /admin/login
- **Username**: Email of any admin user (e.g., michael.chen@company.com)
- **Password**: admin123

Admin users from db.json:

- michael.chen@company.com
- emily.rodriguez@company.com
- amanda.wilson@company.com
- maria.garcia@company.com
- daniel.thompson@company.com

Routes

Web Routes

- GET / - Redirects to /login
- GET /signup - User sign up page
- POST /signup - Create new user account
- GET /login - User login page
- POST /login - Process user login with registration check
- GET /admin/login - Admin login page
- POST /admin/login - Process admin login
- GET /logout - Universal logout (clears all cookies)
- GET /admin/logout - Admin logout (deprecated, use /logout)
- GET /directory - Directory page for non-admin users (requires authentication)
- GET /admin/dashboard - Dashboard for admin users (requires authentication)
- GET /admin/add-employee - Add employee form
- POST /admin/add-employee - Create new employee
- GET /admin/edit-employee/:id - Edit employee form
- POST /admin/edit-employee/:id - Update employee
- POST /admin/delete-employee/:id - Delete employee

API Routes

- GET /api/employees - Get all employees
- GET /api/employees/:id - Get single employee
- POST /api/employees - Create employee
- PUT /api/employees/:id - Update employee
- DELETE /api/employees/:id - Delete employee

Testing

Run Cypress Tests (Headless)

```
npm test
```

Run Cypress Tests (Visual Environment)

```
npm run test:open
```

The Cypress test suite includes:

- Admin authentication tests
- Dashboard display validation
- CRUD operation tests (Create, Read, Update, Delete)
- API endpoint testing
- Form validation tests
- Logout functionality tests

Project Structure

```
├── server.js            # Main Express server file
├── db.json              # JSON database file
├── package.json         # Project dependencies and scripts
├── cypress.config.js    # Cypress configuration
├── views/
│   ├── signup.ejs       # Sign up page
│   ├── login.ejs        # User login page
│   ├── directory.ejs    # Employee directory page
│   └── admin/
│       ├── login.ejs    # Admin login page
│       ├── dashboard.ejs  # Admin dashboard page
│       ├── add-employee.ejs   # Add employee form
│       └── edit-employee.ejs  # Edit employee form
└── cypress/
    └── e2e/
        └── employee-management.cy.js  # Cypress test cases
```

Database Schema

Each employee record contains:

- id (number): Unique identifier
- name (string): Employee name
- designation (string): Job title
- email (string): Email address
- password (string): User password (stored in plain text for demo)
- phone (string): Phone number
- department (string): Department name
- joiningDate (string): Date of joining (YYYY-MM-DD)
- location (object):
    - city (string): City name
    - state (string): State abbreviation
- isAdmin (boolean): Admin privileges flag

- registrationCompleted (boolean): Whether user has completed profile registration

Features in Detail

Sign Up

- Users can create accounts with email (username) and password
- New accounts are created with isAdmin: false by default
- Initial employee record has registrationCompleted: false
- Duplicate email registration is prevented with error message
- After successful signup, redirects to /login

User Login

- Separate login page from admin login (/login vs /admin/login)
- Validates credentials against stored passwords
- **Registration Status Check**:
  - If registrationCompleted: false → Error: "Please contact a system administrator to finish the signup process"
  - If registrationCompleted: true → Proceed to routing
- **Conditional Routing**:
  - Admin users (isAdmin: true) → /admin/dashboard
  - Regular users (isAdmin: false) → /directory
- Sets appropriate cookies for session management

Employee Directory

- Accessible only to logged-in users (admins and regular users)
- Displays complete employee list in a responsive table
- **Table Columns**:
  - ID
  - Name
  - Designation
  - Email (clickable mailto link)
  - Phone
  - Department (displayed as badge)
  - Join Date (formatted)
  - Location (City, State)
- Shows total employee count
- Clean Bootstrap-styled interface with hover effects
- Navigation bar with user name and logout button

Admin Dashboard

- Displays total employee count
- Shows 4 most recently joined employees
- Quick access to Add Employee functionality
- Edit and Delete buttons for each employee card

Employee Cards

Each employee card displays:

- Employee name (as title)
- **Designation**: Job title
- **Email**: Contact email
- **Phone**: Phone number
- **Department**: Department name
- **Join Date**: Formatted joining date
- **Location**: City, State
- Action buttons: Delete and Edit

Form Validation

- All required fields are validated
- Email format validation
- Date picker for joining date
- Dropdown selections for designation and department

Development Notes

- The application uses synchronous file system operations for simplicity
- Authentication is cookie-based with a 1-hour expiration
- For production use, implement proper password hashing and authentication
- Consider using a real database (MongoDB, PostgreSQL, etc.) for production

License

ISC