The application project is to create a new Employee application for our HR department. With the recent directive from the VP of IT, all new applications will be done in Angular.

The data for the application already exists and is provided to all applications via a JSON service running in Azure.

Here is a rendering of the user interface for the application:

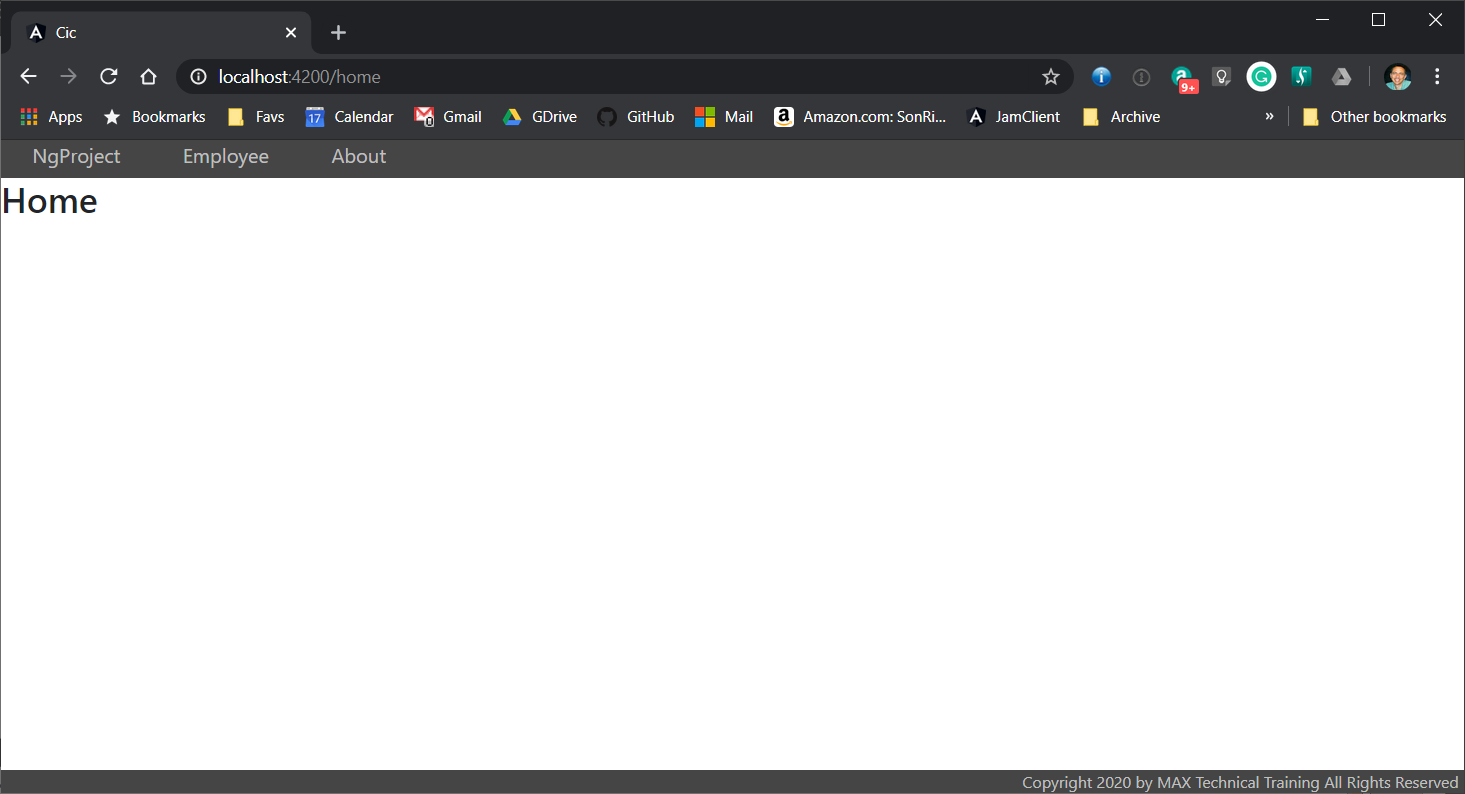


Figure 1 - Home page

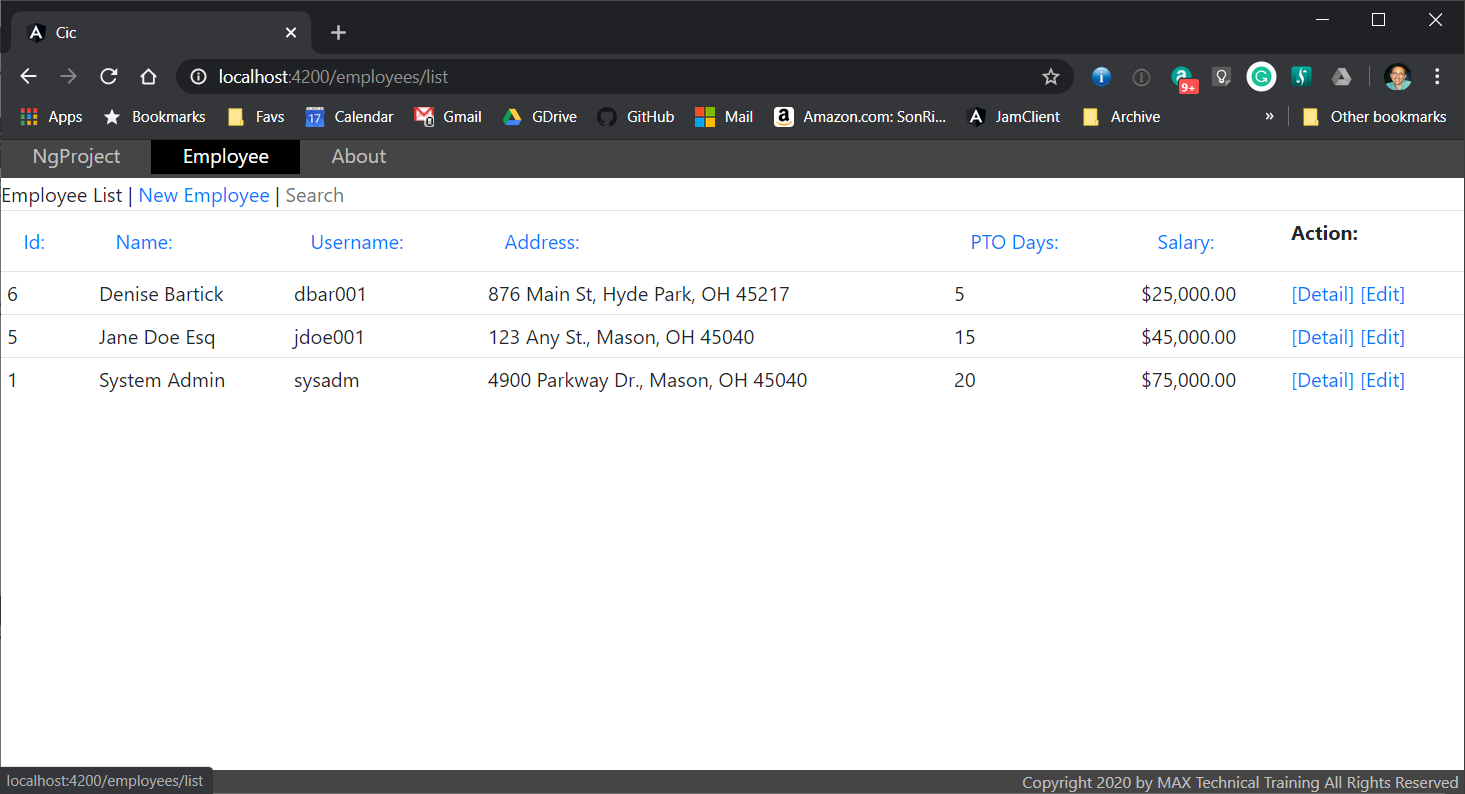


Figure 2 - Employee list page

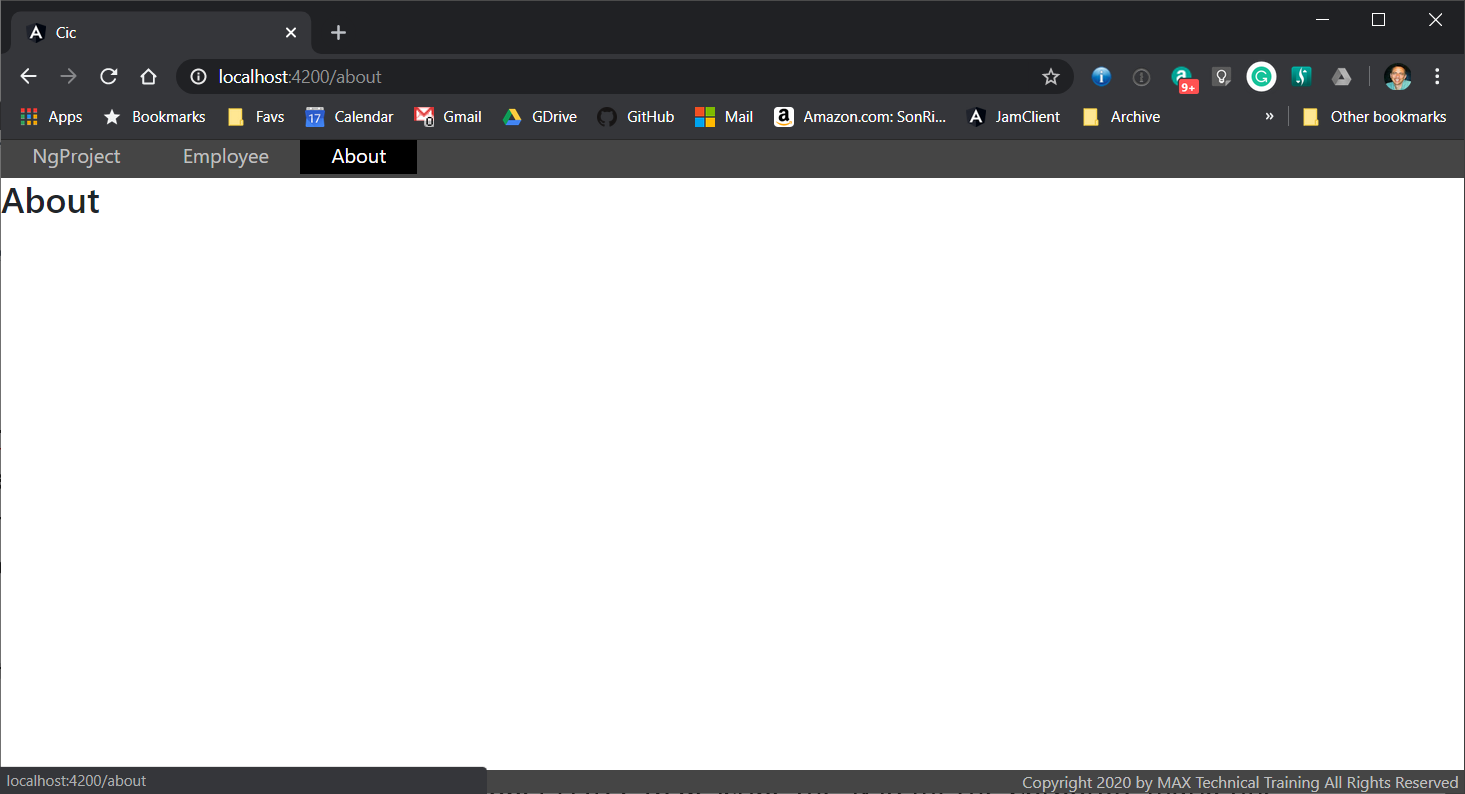


Figure 3 - About page

## Menu

The top line with the dark background is a menu with only three options:

* **NgProject** – navigates to the home page
* **Employee** – navigates to the set of pages that display and maintain the employee information
* **About** – navigates to the about page where the author can show profile information about themselves, so users know who to contact

The construction of the menu should permit adding new menu items very easy. While not a requirement today, some individual menu items may need to be rendered differently from others.

## Home page

The home page is just static text at this time. The developer is free to describe the system. The Corporate marketing group is working on the design of the home page. It can be left blank.

## Employee List page

The Employee list page is the heart of the system at this time. It displays all the employees contained in the system in a scrollable list. Details about this and other employee pages and their requirements are found later in this document.

## About page

The about page is just static text at this time. The developer is free include the developer profile on this page. It is recommended that the developer include some simple profile information.

## Footer

There should be a display of copyright information displayed at the bottom of every page. Is should say, "Copyright 2020 by CinFin All Rights Reserved"

## Data Source

The data source for this application is provided by a JSON service hosted in Microsoft's Azure Cloud. All the APIs needed already exist in the service. The developer will be required to make asynchronous calls to the service. All response data from the service will be JSON data. The service is found at the following url:

https://ngjsonservice20191211114126.azurewebsites.net

This service provides the following APIs:

* GET /api/employees – returns all employees
* GET /api/employees/5 – returns single employee with primary key 5 or not found
* POST /api/employees – inserts new employee; pass employee in request body
* PUT /api/employees/5 – updates employee with primary key 5; pass employee in request body
* DELETE /api/employees/5 – deletes employee with primary key 5; no data in request body

## Employee data

The following table provides details on the employee data:

|  |  |
| --- | --- |
| id | Number; Not Null; Primary Key; Generated |
| username | String; Max 30; Not Null |
| password | String; Max 30; Not Null |
| name | String; Max 50; Not Null |
| address | String; Max 30; Not Null |
| city | String; Max 30; Not Null |
| state | String; Max 2; Not Null |
| zip | String; Max 10; Not Null |
| isActive | Boolean; Not Null; Default true |
| salary | Integer; Not Null |
| ptoDays | Integer; Not Null |

## User Interface

This section defines the user interface for all the employee components.

### Employee List

The employee list component is illustrated in figure 2 above.

It displays the general menu at the top of the component as all components do.

Under the menu, there is a static page title (Employee list). The page title will be different on every component.

Next, there is a link to the Employee Add component.

Next, there is a search box. The search filters the displayed employees in the list by to only employees with data that contains the string entered in the search box. The filtering occurs with every character entered in the search box.

Each column list heading except Action is a link clickable by the user. When a column header is clicked, the displayed employee data is sorted in ascending sequence by that column. If the column header is clicked again, the same column is sorted in descending sequence. Clicked a third time and the sort order reverts to ascending sequence etc.

Each column displays the property of the employee as it exists in the instance. There are special considerations for the address and salary columns

The address column displays a concatenation of the address, city, state, and zip properties.

The salary column displays the number as currency in the US with decimals even though the data is integer.

### Actions

The Action column displays two links for every row. The Detail link navigates to the Employee Detail component to display the selected employee is read only mode. The Edit link navigates to the Employee Edit component allowing modification of the selected employee.

### Employee Detail

The employee detail page displays a single employee. No changes are allowed. The particular employee to display is provided in the route by including the employee's primary key.

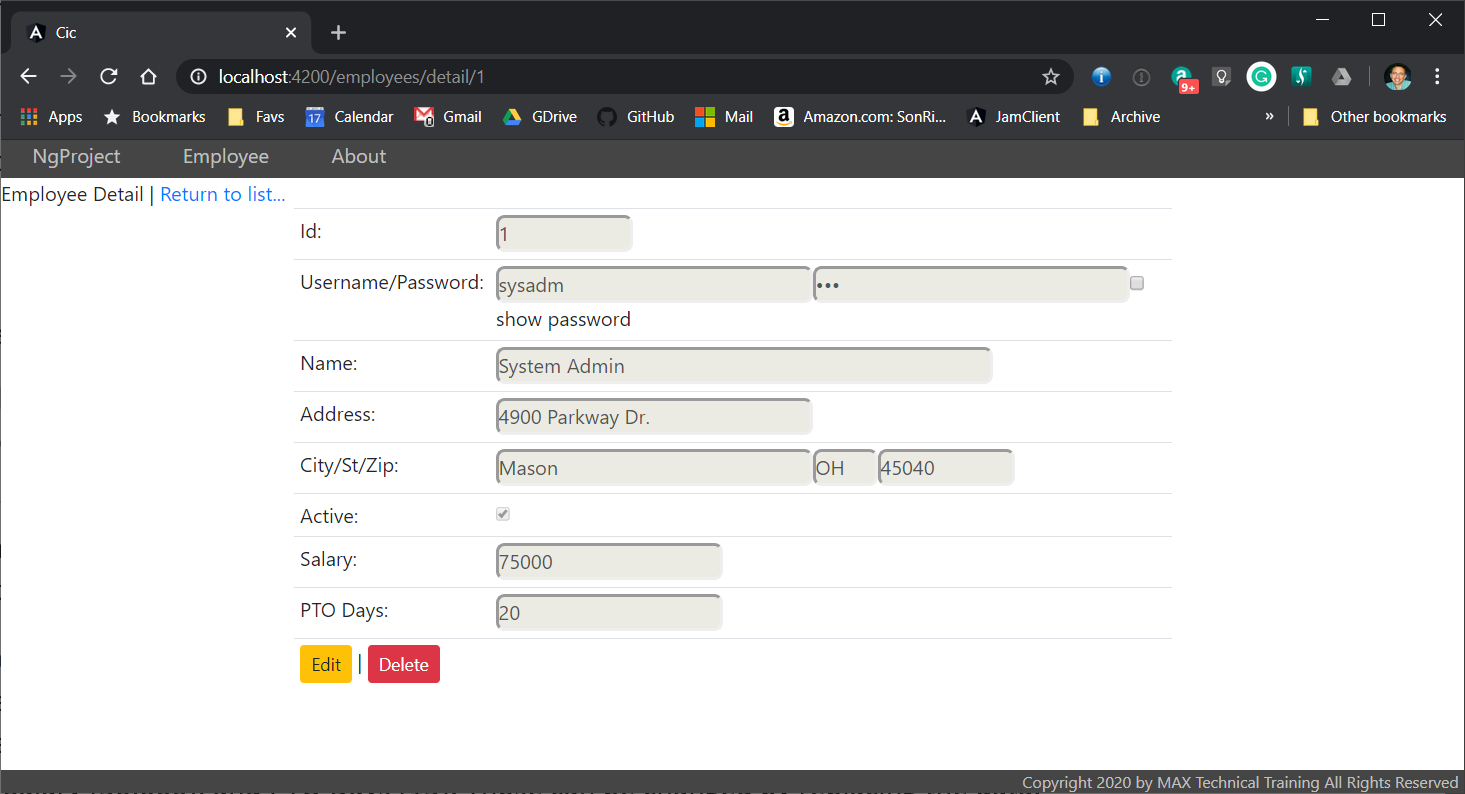


Figure - Employee Detail Page

The HTML is built using strings for tags and <input> controls for binding the data. The each <input> control includes the *disabled* attribute to prevent changing the data.

The password field is displayed as masked (with asterisks) unless the user clicks the *show password* checkbox which causes the password to display in clear text.

At the bottom of the employee detail are two buttons: Edit and Delete. Clicking Edit navigates to the employee edit component for the same employee. Clicking Delete displays a Verify Delete button which will delete the employee when clicked. After edit or delete, the user is returned to the employee list.

### Employee Create

The employee create page adds new employees. The user interface appears like the employee detail except that all <input> tags are enabled.

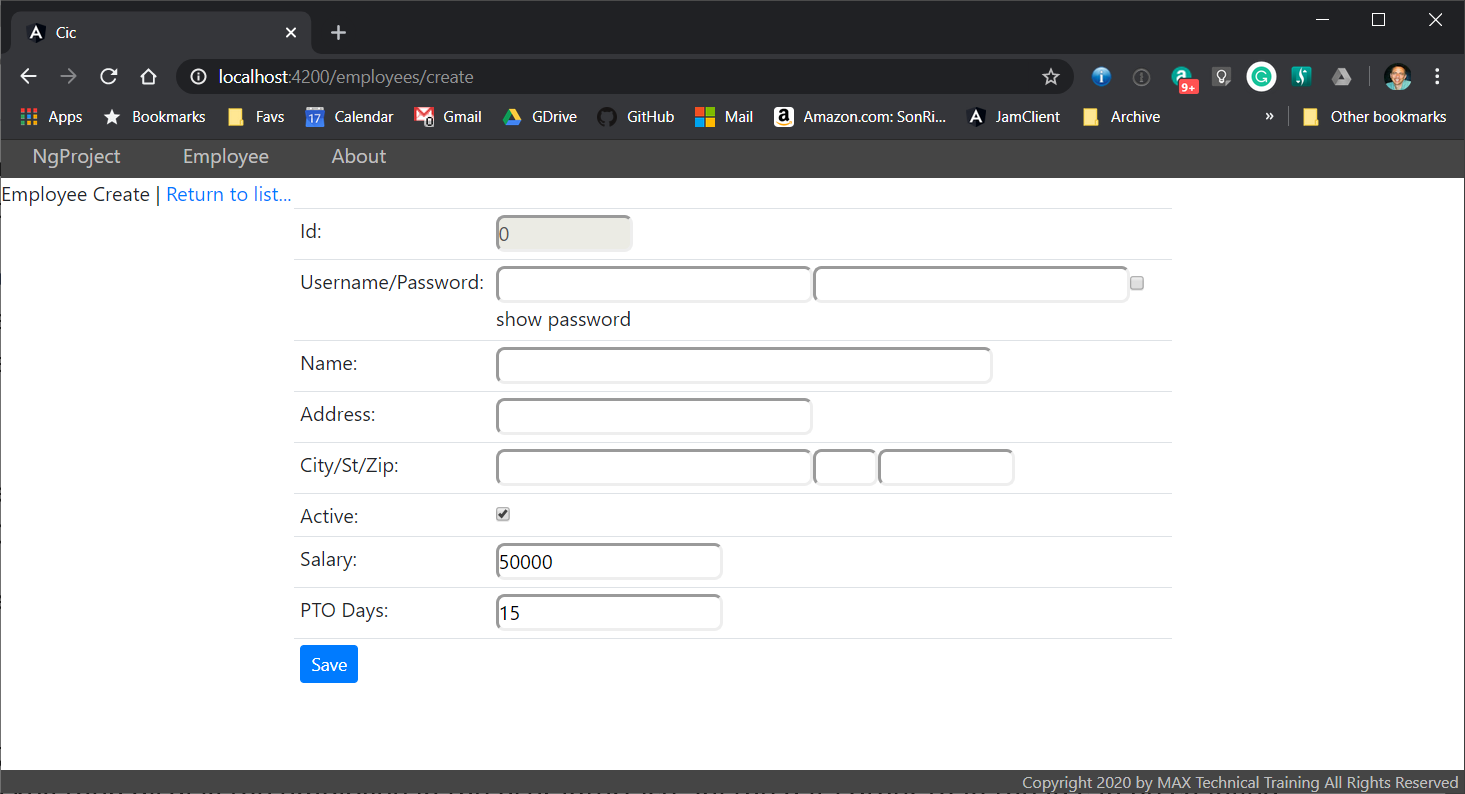


Figure - Employee Create Page

Most fields will be empty when creating a new employee, but some will have default values like Active (true), Salary (50,000), and PTO Days (15). Those can be changed by replacing the data.

When all the data is entered, clicking the Save button will add the employee and return to the employee list.

### Employee Edit

The employee edit page allows changing most of the data about an employee. The particular employee to change is passed by placing the Id of the employee at the end of the route.

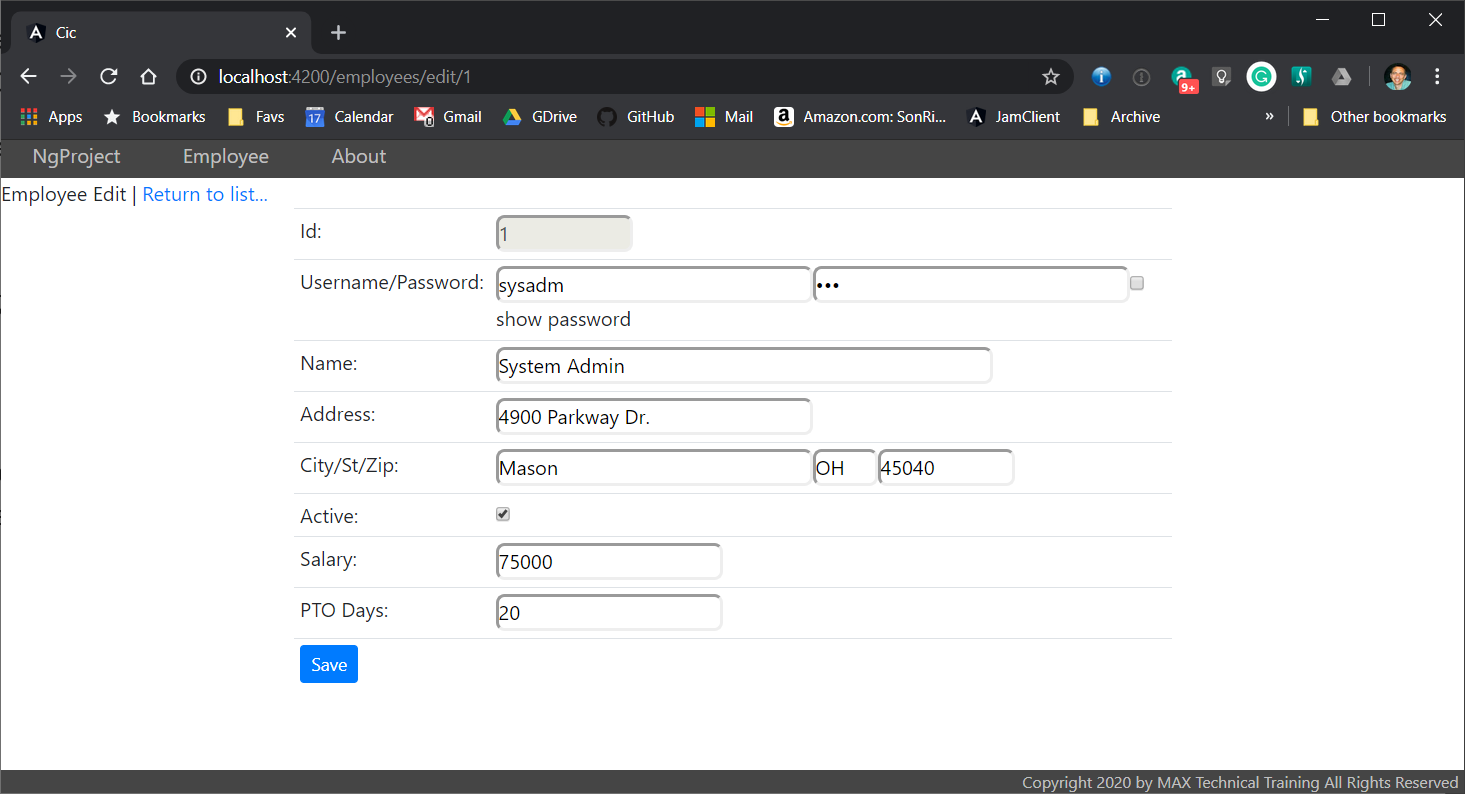


Figure - Employee Edit Page

The employee edit will read for the employee with that passed in primary key then display the employee in the user interface. All the data other than the Id can be changed. When done, clicking the Save button will update the employee data and navigate to the employee list page.