

Web Technologies Lab

Subject Code: MCAL14

A Practical Journal Submitted in Fulfillment of
the Degree of

MASTER

In

COMPUTERAPPLICATION

Year 2023-2024

By

Mr. KAMBLE TEJAS GUNAJI ANITA

(Application Id: - 74610)

Semester- 1

Under the Guidance of

Prof. Dr. Sujatha Iyer



Institute of Distance and Open Learning
Vidya Nagari, Kalina, Santacruz East – 400098.
University of Mumbai

PCP Center

[Satish Pradhan Dyanasadhana College, Thane]



**Institute of Distance and Open Learning,
Vidyanagari, Kalina, Santacruz (E) -400098**

CERTIFICATE

This to certify that, **Mr. KAMBLE TEJAS GUNAJI ANITA** appearing **Master in Computer Application (Semester I) Application ID: 74610** has satisfactory completed the prescribed practical of **MCAL13-Advanced Database Management System Lab** as laid down by the University of Mumbai for the academic year 2023-24

Teacher in charge

Examiners

Coordinator
IDOL, MCA
University of Mumbai

Date: -

Place: -

Index

Practical No	Practical	Page No.	Sign.
1	What is node.js?	1	
2	To demonstrate the use of REPL Terminal in node.js	4	
3	To demonstrate the use of Standard callback pattern in node.js	7	
4	To demonstrate the event emitter pattern in node.js	9	
5	What is angular.js?	11	
6	Write a simple program for multiplication in AngularJS	14	
7	Write a program to display your name with welcome note: HELLO in AngularJS	15	
8	Create simple User Registration form in AngularJS	17	

Practical No. 1

Aim: What is node js?

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Features of Node.js:

Following are some of the important features that make Node.js the first choice of software architects.

- **Asynchronous and Event Driven** – All APIs of Node.js library is asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
- **Very Fast** – Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- **Single Threaded but Highly Scalable** – Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
- **No Buffering** – Node.js applications never buffer any data. These applications simply output the data in chunks.
- **License** – Node.js is released under the MIT license

Node.js Process Model:

The Node.js process model differs from traditional web servers in that Node.js runs in a single process with requests being processed on a single thread. One advantage of this is that Node.js requires far fewer resources. When a request comes in, it will be placed in an event queue. Node.js uses an event loop to listen for events to be raised for an asynchronous job. The event loop continuously runs, receiving requests from the event queue.

There are two scenarios that will occur depending on the nature of the request. If the request is non-blocking, it does not involve any long running processes or data requests, the response will be immediately prepared and then sent back to the client. In the event the request is blocking, requiring I/O operations, the request will be sent to a worker thread pool. The request will have an associated call-back function that will fire when the request is finished, and the worker thread can send the request to the event loop to be sent back to the client.

Traditional Web Server Model:

The traditional web server model consists of a pool of threads which may process requests. Each time a new request comes in, it is assigned to a different thread in the pool. In the event a request is received, and a thread is not available, the request will have to wait until a previous request finishes, a response is returned, and the thread is returned to the thread pool. In this way, the web server model is synchronous, or blocking.

How to Install Node.js and NPM on Windows:

In a web browser, navigate to <https://nodejs.org/en/download/>. Click the Windows Installer button to download the latest default version. At the time this article was written, version 10.16.0-x64 was the latest version. The Node.js installer includes the NPM package manager.

Downloads

Latest LTS Version: 18.15.0 (includes npm 9.5.0)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS
Recommended For Most Users


Windows Installer
node-v18.15.0-x64.msi

Current
Latest Features


macOS Installer
node-v18.15.0.pkg


Source Code
node-v18.15.0.tar.gz

Windows Installer (.msi)
Windows Binary (.zip)
macOS Installer (.pkg)
macOS Binary (.tar.gz)
Linux Binaries (x64)
Linux Binaries (ARM)
Source Code

32-bit	64-bit
32-bit	64-bit
64-bit / ARM64	
64-bit	ARM64
64-bit	
ARMv7	ARMv8
node-v18.15.0.tar.gz	

Step 2: Install Node.js and NPM from Browser

1. Once the installer finishes downloading, launch it. Open the downloads link in your browser and click the file. Or browse to the location where you have saved the file and double-click it to launch.
2. The system will ask if you want to run the software – click Run.
3. You will be welcomed to the Node.js Setup Wizard – click Next.
4. On the next screen, review the license agreement. Click Next if you agree to the terms and install the software.
5. The installer will prompt you for the installation location. Leave the default location, unless you have a specific need to install it somewhere else – then click Next.
6. The wizard will let you select components to include or remove from the installation. Again, unless you have a specific need, accept the defaults by clicking Next.

7. Finally, click the Install button to run the installer. When it finishes, click Finish.

Step 3: Verify Installation:

Open a command prompt (or PowerShell) and enter the following:

`node -v`

The system should display the Node.js version installed on your system. You can do the same for NPM: `npm -v`

Practical No. 2

Aim: To demonstrate the use of REPL Terminal in node.js.

REPL stands for Read Eval Print Loop and it represents a computer environment like a Windows console or Unix/Linux shell where a command is entered and the system responds with an output in an interactive mode. Node.js or **Node** comes bundled with a REPL environment. It performs the following tasks –

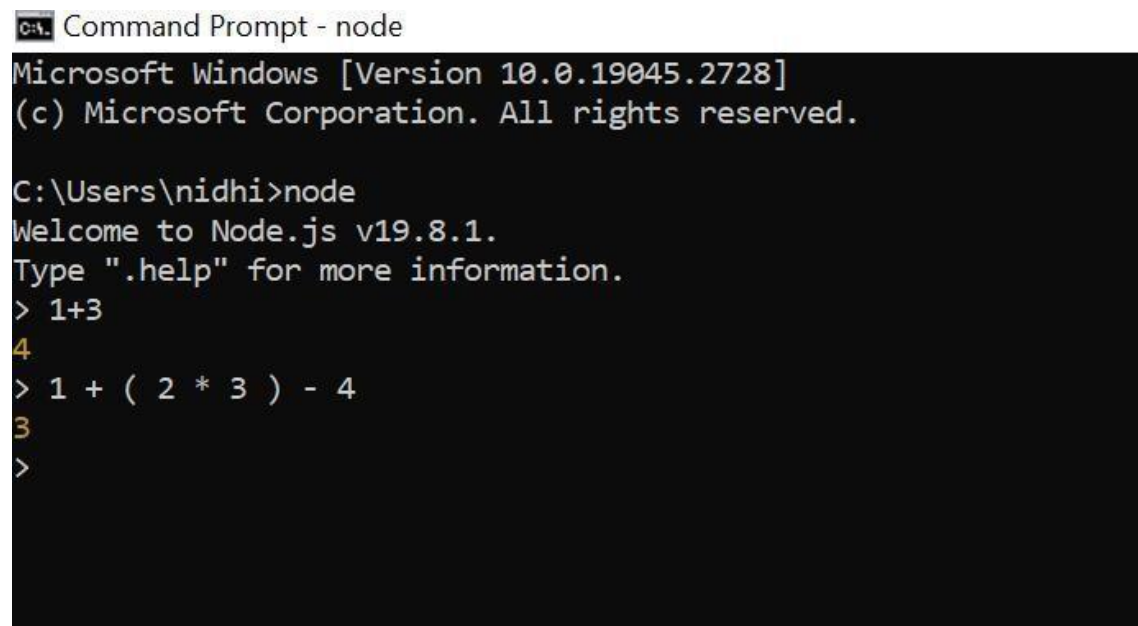
- **Read** – Reads user's input, parses the input into JavaScript data-structure, and stores in memory.
- **Eval** – Takes and evaluates the data structure.
- **Print** – Prints the result.
- **Loop** – Loops the above command until the user presses **ctrl-c** twice.

The REPL feature of Node is very useful in experimenting with Node.js codes and to debug JavaScript codes.

Online REPL Terminal

To simplify your learning, we have set up an easy to use Node.js REPL environment online, where you can practice Node.js syntax – [Launch Node.js REPL Terminal](#)

Simple Expression



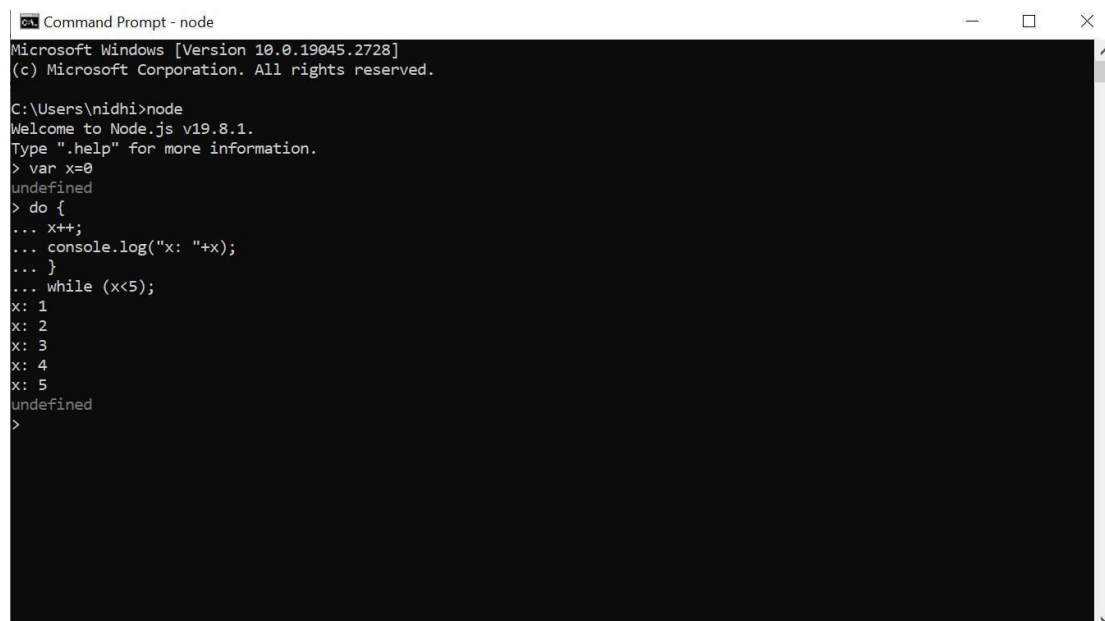
```
C:\> Command Prompt - node
Microsoft Windows [Version 10.0.19045.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nidhi> node
Welcome to Node.js v19.8.1.
Type ".help" for more information.
> 1+3
4
> 1 + ( 2 * 3 ) - 4
3
>
```

Use Variables

```
> x=10
10
> var y=10
undefined
> x+y
20
> console.log("Hello World")
Hello World
undefined
>
```

Multiline Expression



The screenshot shows a Windows Command Prompt window titled "Command Prompt - node". The window displays the Node.js v19.8.1 welcome message and the user's input of a multiline expression. The expression is a `do-while` loop that increments a variable `x` from 0 to 5 and logs its value. The output shows the values 1 through 5 being printed on separate lines, followed by `undefined` and a new prompt `>`.

```
Microsoft Windows [Version 10.0.19045.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nidhi>node
Welcome to Node.js v19.8.1.
Type ".help" for more information.
> var x=0
undefined
> do {
... x++;
... console.log("x: "+x);
... }
... while (x<5);
x: 1
x: 2
x: 3
x: 4
x: 5
undefined
>
```

Underscore Variable

Web technologies lab

```
> var x=10
undefined
> var y=20
undefined
> x+y
30
> var sum=_
undefined
> console.log(sum)
30
undefined
>
```

Practical No. 3

Aim: To demonstrate the use of standard callback pattern in node.js.

Blocking Code Example

Create a text file named **input.txt** with the following content –

```
Tutorials Point is giving self-learning content  
to teach the world in simple and easy way!!!!
```

Create a js file named **main.js** with the following code –

```
var fs = require("fs");  
var data = fs.readFileSync('input.txt');  
  
console.log(data.toString());
```

Now run the main.js to see the result –

```
C:\Users\nidhi\callback>node main.js  
Tutorials Point is giving self learning content  
to teach the world in simple and easy way!!!!  
Program Ended  
  
C:\Users\nidhi\callback>
```

Non-Blocking Code Example

Create a text file named input2.txt with the following content.

Hey There I'm using Node JS to demonstrate Non-Blocking Code Example Update main2.js to have the following code –

```
var fs = require("fs");  
fs.readFile('input2.txt', function (err, data) { if (err) return console.error(err);  
console.log(data.toString());  
});  
console.log("Program Ended");
```

```
C:\Users\nidhi\callback>node main2.js  
Program Ended  
Hey There I m using Node JS to demonstrate Non-Blocking Code Example  
  
C:\Users\nidhi\callback>
```

Practical No. 4

Aim: To demonstrate the event emitter pattern in node.js

When an EventEmitter instance faces any error, it emits an 'error' event. When a new listener is added, 'newListener' event is fired and when a listener is removed, 'removeListener' event is fired.

Event Emitter provides multiple properties like on and emit. on property is used to bind a function with the event and emit is used to fire an event.

Create a js file named main3.js with the following Node.js code –

```
var events = require('events');
var EventEmitter = new events.EventEmitter();
// listener #1
var listner1 = function listner1() { console.log('listner1 executed.')}
}
// listener #2
var listner2 = function listner2() { console.log('listner2 executed.')}
}
// Bind the connection event with the listner1 function
eventEmitter.addListener('connection', listner1);
// Bind the connection event with the listner2 function
eventEmitter.on('connection', listner2);
```

Aim: To demonstrate the event emitter pattern in node.js

When an EventEmitter instance faces any error, it emits an 'error' event. When a new listener is added, 'newListener' event is fired and when a listener is removed, 'removeListener' event is fired.

EventEmitter provides multiple properties like on and emit. on property is used to bind a function with the event and emit is used to fire an event.

Create a js file named main3.js with the following Node.js code –

```
var events = require('events');
var EventEmitter = new events.EventEmitter();
// listener #1
var listner1 = function listner1() { console.log('listner1 executed.')}
// listener #2
var listner2 = function listner2() { console.log('listner2 executed.')}
// Bind the connection event with the listner1 function
eventEmitter.addListener('connection', listner1);
// Bind the connection event with the listner2 function
eventEmitter.on('connection', listner2);
```

```
C:\Users\nidhi\callback>node main3.js
2 Listener(s) listening to connection event
listner1 executed.
listner2 executed.
listner1 will not listen now.
listner2 executed.
1 Listener(s) listening to connection event
Program Ended.

C:\Users\nidhi\callback>
```

Practical No. 5

AngularJS is a structural framework for dynamic web applications. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application

components clearly and succinctly. Its data binding and dependency injection eliminates much

of the code you currently have to write. And it all happens within the browser, making it an

Aim: What is Angular JS?

ideal partner with any server technology.

Core Features

The core features of AngularJS are as follows –

Data-binding – It is the automatic synchronization of data between model and view components.

Scope – These are objects that refer to the model. They act as a glue between controller and view.

Controller – These are JavaScript functions bound to a particular scope.

Services – AngularJS comes with several built-in services such as \$http to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.

Filters – These select a subset of items from an array and returns a new array.

Directives – Directives are markers on DOM elements such as elements, attributes, css, and more. These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives such as ngBind, ngModel, etc.

Templates – These are the rendered view with information from the controller and model.

These can be a single file (such as index.html) or multiple views in one page using *partials*.

Routing – It is concept of switching views.

Model View Whatever – MVW is a design pattern for dividing an application into different parts called Model, View, and Controller, each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model- View-ViewModel). The Angular JS team refers it humorously as Model View Whatever.

Deep Linking – Deep linking allows to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.

Dependency Injection – AngularJS has a built-in dependency injection subsystem that helps the developer to create, understand, and test the applications easily.

SETUP ANGULARJS DEVELOPMENT ENVIRONMENT:

The following tools are needed to setup a development environment for AngularJS: □ AngularJS Library

- Editor/IDE
- Browser
- Web server

AngularJS Library:

To download AngularJS library, go to angularjs.org -> click download button, which will open the following popup.

Download AngularJS

Branch

1.8.x (latest)1.2.x (legacy)

Build

MinifiedUncompressedZip

CDN

<https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js>

Bower

bower install angular#1.8.2

npm

npm install angular@1.8.2

Extras

[Browse additional modules](#)

Previous Versions

 Download

Download AngularJS Library

Select the required version from the popup and click on download button in the popup.

CDN: AngularJS library can be included from CDN url -

<https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js>

Editor:

AngularJS is eventually HTML and JavaScript code.

So, install any good editor/IDE as per your choice.

The following editors are recommended:

- Aptana Studio 3
- Ultra Edit
- Eclipse
- Visual Studio

Online Editor:

The following online editors can be used for learning purpose.

- plnkr.co
- jsbin.com

Web server:

Use any web server such as IIS, apache etc., locally for development purpose.

Browser:

Install any browser of your choice as AngularJS supports cross-browser compatibility. However, it is recommended to use Google Chrome while developing an application.

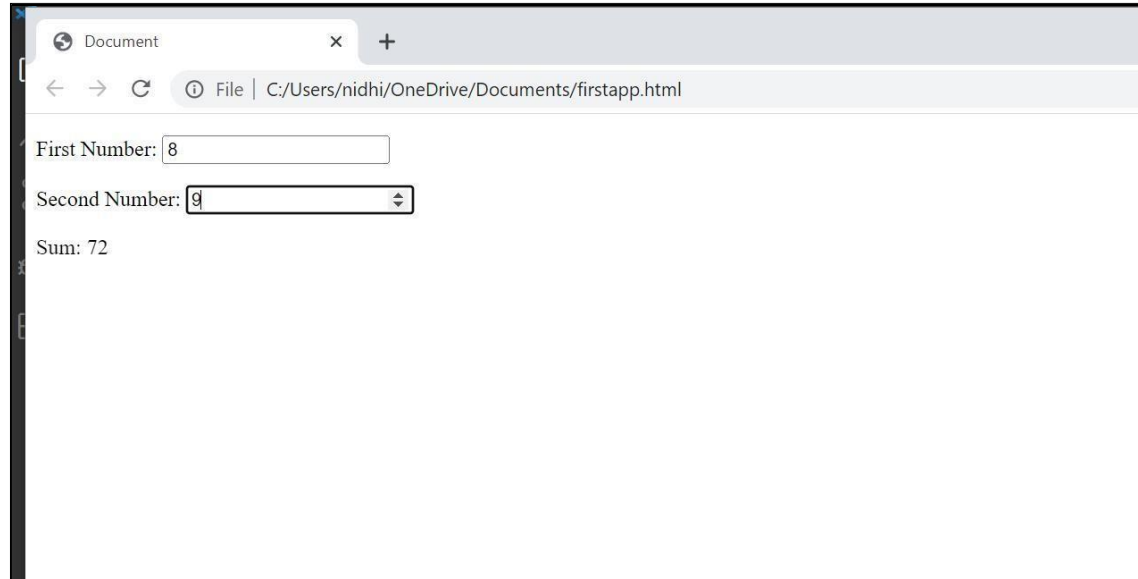
Angular Seed:

Use Angular seed project to quickly get started on AngularJS application. The Angular-seed is an application skeleton for a typical AngularJS web application. It can be used to quickly bootstrap your angular webapp projects and development environment for your project. Download angular-seed from GitHub.

Practical No. 6

Aim: Write a Simple program for multiplication in AngularJS.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
init="num1=0" />
</p>
<p>Second Number:
<input type="number" ng-model="num2" ng-init="num2=0" />
</p>
<p>Sum: {{ num1 * num2 }}</p>
</div>
</body>
</html>
```



Practical No. 7

Aim: Write a program to display your name with Welcome note Hello in Angular JS.

Program:

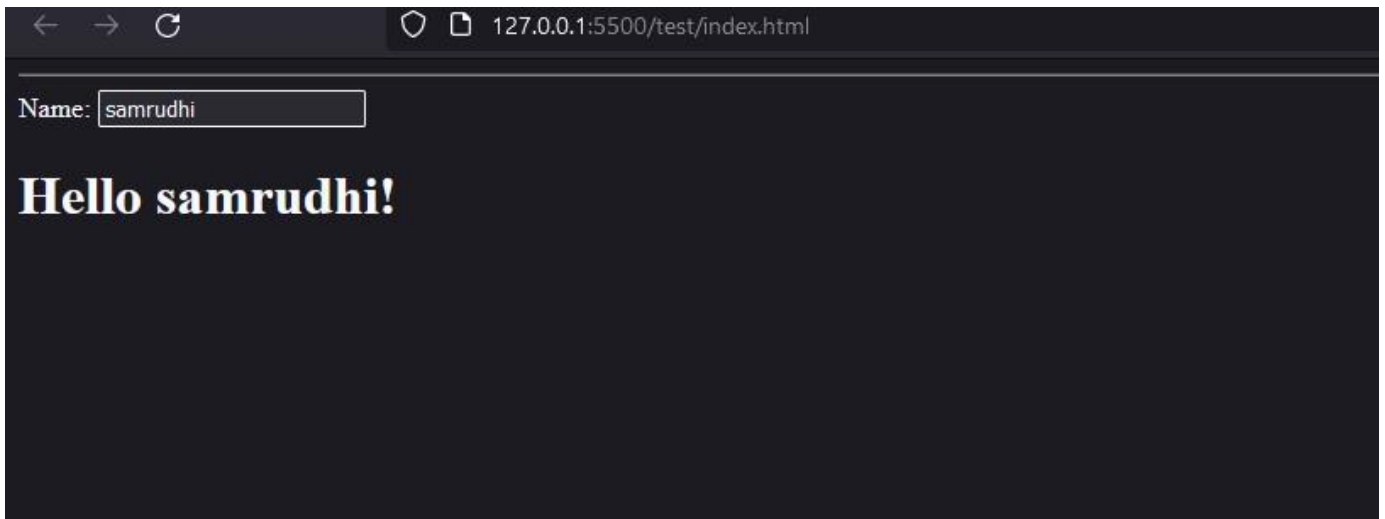
testAngularJS.js

```
<html>
<head>
<title>AngularJS First Application</title>
</head>
<body>
<h1>Sample Application</h1>
<div ng-app = "">
<p>Enter your Name: <input type = "text" ng-model = "name"></p>
<p>Hello <span ng-bind = "name"></span>!!</p>
</div>
<script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js" ">
</script>
</body>
</html>
```

index.html

```
<hr><!doctype html>
<html ng-app>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js" ">
</script>
</head>
<body>
<div>
<label>Name:</label>
<input type="text" ng-model="yourName" placeholder="Enter a name
here">
<h1>Hello {{yourName}}!!</h1>
</div>
</body>
</html>
```

OUTPUT:



Practical No. 8

Aim: Create Simple User Registration Form in AngularJS.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>HTML Starter</title>

<meta name="viewport" content="width=device-width, initial-scale=1">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.26/angular.min.js"></script>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
<script src="app.js"></script>
</head>
<body>
<div ng-app = "myApp" class = "container" style="width:550px">
<div style="text-align:center;color:blue">
<h3><b>User Registraion Form</b></h3>
</div>
<div ng-controller = "ContactController">
<div align="right">
<a href="#" ng-click="searchUser()">{{title}}</a>
</div>
<form role = "form" class="well" ng-hide="ifSearchUser">
<div class = "form-group">
<label for = "name"> Name: </label>
<input type = "text" id = "name" class = "form-control" placeholder = "Enter Name " ng-model =
"newcontact.name">
</div>
<div class = "form-group">
<label for = "email"> Email: </label>
<input type = "email" id = "email" class = "form-control" placeholder = "Enter Email " ng-model =
"newcontact.email">
</div>
<div class = "form-group">
<label for = "password"> Password: </label>
<input type = "password" id = "password" class = "form-control" placeholder = "Enter Password " ng-model =
"newcontact.password">
</div>
<div class = "form-group">
<label for = "phone"> Phone: </label>
<input type = "text" id = "phone" class = "form-control" placeholder = "Enter Phone " ng-model =
"newcontact.phone">
</div>
<br>
<input type="hidden" ng-model="newcontact.id">
<input type="button" class="btn btn-primary" ng-click="saveContact()" class="btn btn-primary" value =
"Submit">
</form>
<div><h4><b>Registered Users</b></h4>
<table ng-if="contacts.length" class = "table table-striped table- bordered table-hover">
<thead>
```

```
<tr class = "info">
<th>Name</th>
<th>Email</th>
<th>Phone</th>
<th ng-if="!ifSearchUser">Action</th>
</tr>
</thead>
<tbody>

<tr ng-repeat = "contact in contacts">
<td>{{ contact.name }}</td>
<td>{{ contact.email }}</td>
<td>{{ contact.phone }}</td>

<td ng-if="!ifSearchUser">
<a href="#" ng-click="edit(contact.id)" role = "button" class = "btn btn-info">edit</a> &nbsp;
<a href="#" ng-click="delete(contact.id)" role = "button" class = "btn btn-danger">delete</a>
</td>
</tr>
</tbody>
</table>
<div ng-hide="contacts.length > 0">No Users Found</div>

</div>
</div>
</div>
</body>
</html>
```

Web technologies lab

App.js

```
var myApp = angular.module("myApp", []); myApp.service("ContactService",  
function(){ var uid = 1;  
var contacts =  
[ { 'id' : 0,  
'name' : 'Steve John', 'email' : 'john@gmail.com', 'password': 'John123', 'phone' : '911-91-199-999' }];  
  
// Save Service for saving new contact and saving existing edited contact. this.save = function(contact)  
{  
if(contact.id == null)  
{  
contact.id = uid++; contacts.push(contact);  
}  
else  
{  
for(var i in contacts)  
{  
if(contacts[i].id == contact.id)  
{  
contacts[i] = contact;  
}  
}  
}  
};
```

// serach for a contact

Web

```
this.get = function(id)
{
for(var i in contacts ) {

if( contacts[i].id == id)
{
return contacts[i];
}
}
};

//Delete a contact this.delete = function(id)
{
for(var i in contacts) {

if(contacts[i].id == id)
{
contacts.splice(i,1);
} }
};

//Show all contacts this.list = function() {

});

return contacts;
}      ;

////Controller area .....

myApp.controller("ContactController" , function($scope , ContactService){ console.clear();

$scope.ifSearchUser = false;
$scope.title ="List of Users";

$scope.contacts = ContactService.list();

$scope.saveContact = function()
{
console.log($scope.newcontact);
if($scope.newcontact == null || $scope.newcontact == angular.undefined) return;
ContactService.save($scope.newcontact);
$scope.newcontact = {};
};

$scope.delete = function(id)
{
```

Web technologies lab

```
id)

ContactService.delete(id); if($scope.newcontact != angular.undefined
&& $scope.newcontact.id ==

{
$scope.newcontact = {};
}

};

$scope.edit = function(id)
{
$scope.newcontact = angular.copy(ContactService.get(id));
};
$scope.searchUser = function(){ if($scope.title == "List of Users"){
$scope.ifSearchUser=true;
$scope.title = "Back";
}
else
{
$scope.ifSearchUser = false;

$scope.title = "List of Users";
}
};
});
```


Web technologies lab

HTML Starter x +

File | C:/Users/nidhi/OneDrive/Documents/Index.html#

User Registraion Form

List of Users

Name:

Email:

Password:

Phone:

Registered Users

Name	Email	Phone	Action
Ankit	rg27101972@gmail.com	7387522338	<input type="button" value="edit"/> <input type="button" value="delete"/>