JSPM'S

Jayawantrao Sawant College of Engineering, Hadapsar.

Department of Information Technology

Lab Manual

Lab Practice II [TE IT] 2019

Web Application Development [WAD]

314458

Experiment Level Outcome
ELO

Experiment level outcome	CO mapping	PO mapping
ELO1: Develop Static and Dynamic responsive website using technologies HTML, CSS, Bootstrap and AJAX	CO314453.1 CO314453.2	PO1,PO2,PO3, PO8,PO9,PO10
ELO2: Create Version Control Environment.	CO314453.2	PO1,PO2,PO3, PO8,PO9,PO10
ELO3: Develop an application using front end and backend technologies.	CO314453.3	PO1,PO2,PO3, PO8,PO9,PO10
ELO4: Develop mobile website using JQuery Mobile.	CO314453.4	PO1,PO2,PO3, PO8,PO9,PO10
ELO5: Deploy web application on cloud using AWS.	CO314453.5	PO1,PO2,PO3, PO8,PO9,PO10

Experiments List

Sr. No	Title	CO mapping	PO mapping
1	Create a responsive web page which shows the ecommerce/college/exam admin dashboard with sidebar and statistics in cards using HTML, CSS and Bootstrap.	CO314453.1 CO314453.2	PO1,PO2,PO3,PO8,PO9,PO10,PSO1
2	Write a JavaScript Program to get the user registration data and push to array/local storage with AJAX POST method and data list in new page.	CO314453.2	PO1,PO2,PO3, PO8,PO9,PO10, PSO1
3	Create version control account on GitHub and using Git commands to create repository and push your code to GitHub	CO314453.2	PO1,PO2,PO3, PO8,PO9,PO10, PSO1
4	Create an Angular application which will do following actions: Register User, Login User, Show User Data on Profile Component	CO314453.3	PO1,PO2,PO3, PO8,PO9,PO10, PSO1
5	**5-A** Create a Node.JS Application which serves a static website. **5-B** Create four API using Node.JS, ExpressJS and MongoDB for CURD Operations	CO314453.3	PO1,PO2,PO3, PO8,PO9,PO10, PSO1
6	Create a simple Mobile Website using jQuery Mobile.	CO314453.4	PO1,PO2,PO3, PO8,PO9,PO10, PSO1
7	Deploy/Host Your web application on AWS VPC or AWS Elastic Beanstalk	CO314453.5	PO1,PO2,PO3, PO8,PO9,PO10, PSO1
8	**Mini Project** **Develop a web application using full stack development**	All CO	PO1,PO2,PO3, PO8,PO9,PO10, PSO1

Assignment No.1

Title: Create a responsive web page using HTML, CSS, Bootstrap

Problem Statement: Create a responsive web page which shows the ecommerce/college/exam admin dashboard with sidebar and statistics in cards using HTML, CSS and Bootstrap.

Theory:

I.<u>**HTML**</u>

•HTML stands for Hyper Text Markup Language

•HTML is the standard markup language for creating Web pages

•HTML describes the structure of a Web page

•HTML consists of a series of elements

•HTML elements tell the browser how to display the content

•HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc

A simple HTML Document

```
<! DOCTYPE html>

<html>

<head>

<title>Page Title</title>

</head>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>

</html>
```

- The <!DOCTYPE html> declaration defines that this document is an HTML5 document
- The <html> element is the root element of an HTML page
- The <head> element contains meta information about the HTML page
- The <title> element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
- The <body> element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.

- The <h1> element defines a large heading
- The <p> element defines a paragraph

Different HTML Tags used

1. <a> tag-Hyperlink

The <a> or anchor tag is used to add a hyperlink.

Attributes

i) href – used to mention the link

e.g. <a href="https:// jspmjscoe.edu.in" target="_self">CLICK ME</a>

Open in same window

e.g. <a href="https:// jspmjscoe.edu.in" target="_blank">CLICK ME</a>

Open in different window

2. **HTML Unordered Lists**

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

```
e.g. <ul>

  <li>Coffee</li>

  <li>Tea</li>

  <li>Milk</li>

</ul>
```

Example –Disc style

```
<ul style="list-style-type:disc;">

  <li>Coffee</li>

  <li>Tea</li>

  <li>Milk</li>

</ul>
```

3. HTML Ordered Lists

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag

Example

```
<ol>

  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

The type attribute of the <ol> tag, defines the type of the list item marker:

Type	Description
type="1"	The list items will be numbered with numbers (default)
type="A"	The list items will be numbered with uppercase letters
type="a"	The list items will be numbered with lowercase letters
type="I"	The list items will be numbered with uppercase roman numbers
type="i"	The list items will be numbered with lowercase roman numbers

II. CSS- Cascading Style Sheet

•CSS stands for Cascading Style Sheets

•CSS describes how HTML elements are to be displayed on screen, paper, or in other media

•CSS saves a lot of work. It can control the layout of multiple web pages all at once

•External style sheets are stored in CSS files

•CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

Simple CSS example

```
<!DOCTYPE html>
<html>
<head>
<style> body {
```

```
background-color: lightblue;
}
h1 {
color: white;
text-align: center;
}
p {
font-family: verdana; font-size: 20px;
}
</style>
</head>
<body>
<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

1. **CSS Navigation Bar**

Examples-

Vertical

- Home
- News
- Contact
- About

Horizontal

- Home
- News
- Contact
- About

- Having easy-to-use navigation is important for any web site.
- With CSS you can transform boring HTML menus into good-looking navigation bars.

Navigation Bar = List of Links

Example code - Vertical

```
<ul>

  <li><a href="default.asp">Home</a></li>

  <li><a href="news.asp">News</a></li>

  <li><a href="contact.asp">Contact</a></li>

  <li><a href="about.asp">About</a></li>

</ul>

ul {

  list-style-type: none;

  margin: 0;

  padding: 0;

  width: 60px;

}

li a {

  display: block;

}
```

Link to refer : https://www.w3schools.com/css/css_navbar_vertical.asp

Example1 –Horizontal

```
li {
  display: inline;

}
```

Example 2

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
}

li {
  float: left;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

/* Change the link color to #111 (black) on hover */
li a:hover {
  background-color: #111;
}
```

Link for reference: https://www.w3schools.com/css/css_navbar_horizontal.asp

III. Bootstrap

□ Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites. It solves many problems which we had once, one of which is the cross-browser compatibility issue. Nowadays, the websites are perfect for all the browsers (IE, Firefox, and Chrome) and for all sizes of screens (Desktop, Tablets, Phablets, and Phones).

□ **Why Bootstrap?**

- Faster and Easier Web Development.
- It creates Platform-independent web pages.
- It creates Responsive Web-pages.
- It is designed to be responsive to mobile devices too.
- It is Free! Available on www.getbootstrap.com

Bootstarp Example

```
<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">
```

```
<meta name="viewport" content="width=device-width, initial- scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jq uery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bo otstrap.min.js"></script>

</head>

<body>

<div class="container">

<h1>My First Bootstrap Page</h1>

<p>This is some text.</p>

</div>

</body>

</html>
```

- **Ways to add Bootstrap**

1. Include Bootstrap from a CDN

```
<!-- Latest compiled and minified CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">

<!-- Latest compiled JavaScript -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
```

2. Download Bootstrap from getbootstrap.com

Link to refer: https://www.w3schools.com/bootstrap5/bootstrap_get_started.php

Responsive web design

- Responsive web design (RWD) is a web development approach that creates dynamic changes to the appearance of a website, depending on the screen size and orientation of the device being used to view it.
- Responsive web design is about creating web pages that look good on all devices!
- A responsive web design will automatically adjust for different screen sizes and viewports.
- Responsive Web Design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it look good on all devices (desktops, tablets, and phones)

1. Bootstrap Container

- The .container class provides a responsive fixed width container
- The .container-fluid class provides a full width container, spanning the entire width of the viewport

1. Fixed Container

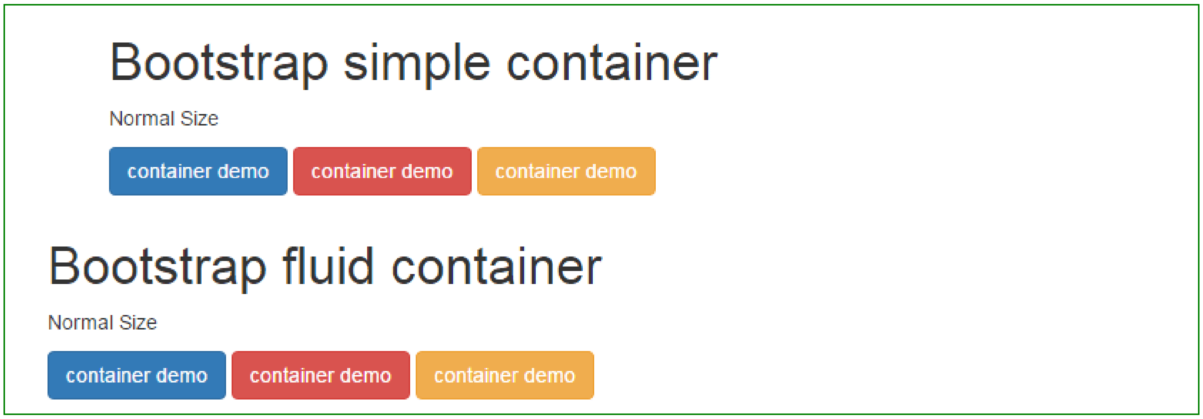
Use the .container class to create a responsive, fixed-width container.
Note that its width (max-width) will change on different screen sizes:

```
<div class="container">
  <h1>My First Bootstrap Page</h1>
  <p>This is some text.</p>
</div>
```

2. Fluid Container

Use the .container-fluid class to create a full width container that will always span the entire width of the screen (width is always 100%):

```
  <div class="container-fluid">
  <h1>My First Bootstrap Page</h1>
  <p>This is some text.</p>
</div>
```



Bootstrap simple container

Normal Size

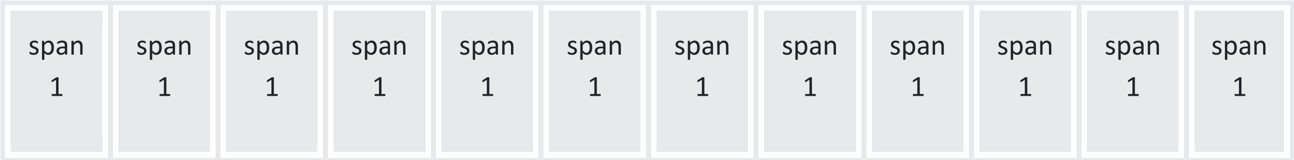
Bootstrap fluid container

Normal Size

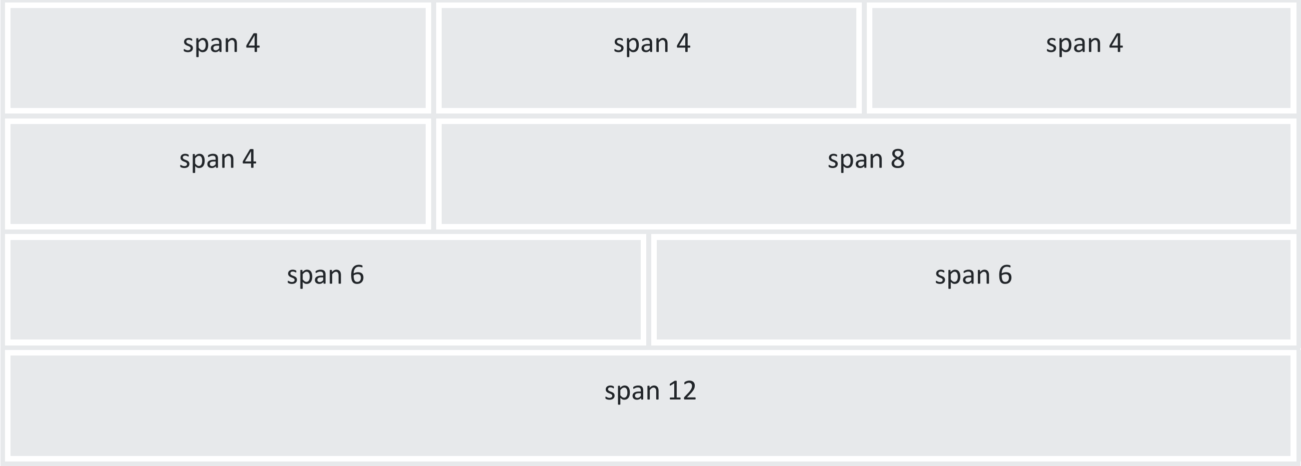
2. Bootstrap Grid system

Bootstrap's grid system is built with flexbox and allows up to 12 columns across the page.

If you do not want to use all 12 columns individually, you can group the columns together to create wider columns:



span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1



span 4 span 4 span 4

span 4 span 8

span 6 span 6

span 12

The grid system is responsive, and the columns will re-arrange automatically depending on the screen size.

Make sure that the sum adds up to 12 or fewer (it is not required that you use all 12 available columns).

Grid Classes

The Bootstrap 5 grid system has six classes:

- `.col-` (extra small devices - screen width less than 576px)
- `.col-sm-` (small devices - screen width equal to or greater than 576px)
- `.col-md-` (medium devices - screen width equal to or greater than 768px)
- `.col-lg-` (large devices - screen width equal to or greater than 992px)
- `.col-xl-` (xlarge devices - screen width equal to or greater than 1200px)
- `.col-xxl-` (xxlarge devices - screen width equal to or greater than 1400px)

The classes above can be combined to create more dynamic and flexible layouts.

Tip: Each class scales up, so if you want to set the same widths for `sm` and `md`, you only need to specify `sm`.

Basic Structure of a Bootstrap 5 Grid

The following is a basic structure of a Bootstrap 5 grid:

```
<!-- Control the column width, and how they should appear on different devices -->
<div class="row">
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
</div>
<div class="row">
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
</div>

<!-- Or let Bootstrap automatically handle the layout -->
<div class="row">
  <div class="col"></div>
  <div class="col"></div>
  <div class="col"></div>
</div>
```

First example: create a row (`<div class="row">`). Then, add the desired number of columns (tags with appropriate `.col-*-*` classes). The first star (*) represents the responsiveness: sm, md, lg, xl or xxl, while the second star represents a number, which should add up to 12 for each row.

Second example: instead of adding a number to each `col`, let bootstrap handle the layout, to create equal width columns: two `"col"` elements = 50% width to each col, while three cols = 33.33% width to each col. Four cols = 25% width, etc. You can also use `.col-sm|md|lg|xl|xxl` to make the columns responsive.

Link for Reference:

1. **https://www.w3schools.com/bootstrap5/bootstrap_containers.php**

2. **https://www.jquery-az.com/bootstrap-container-and-container-fluid-what-is-the-difference/**

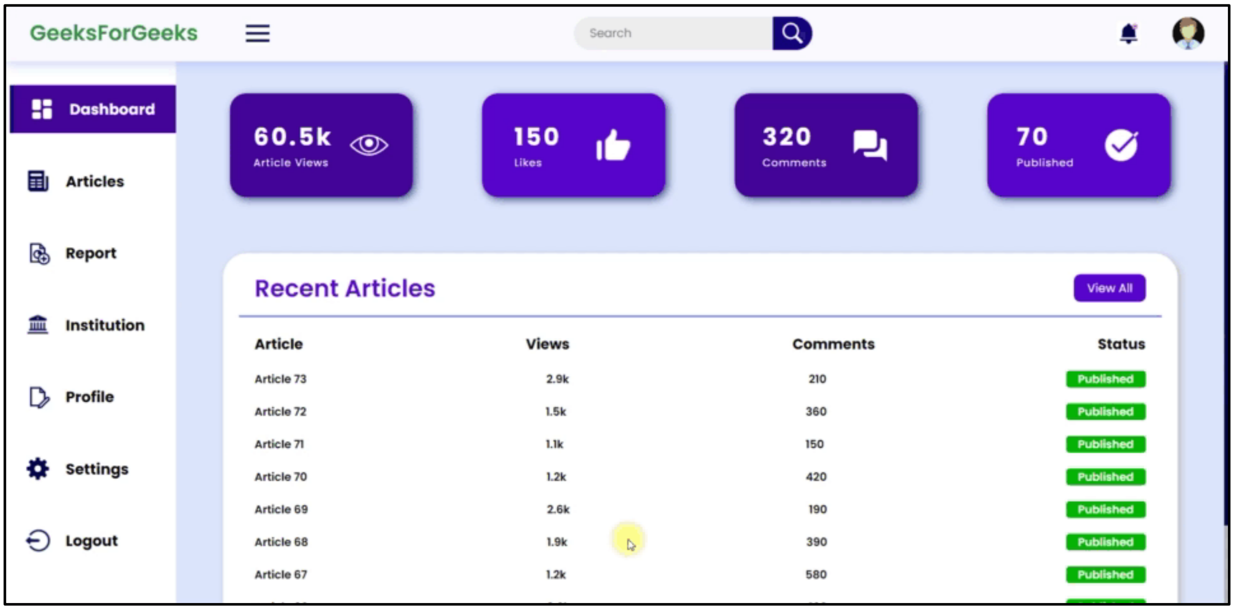
3. **https://getbootstrap.com/docs/5.0/layout/containers/**

Problem Statement: Create a responsive web page which shows the commerce/college/exam admin dashboard with sidebar and statistics in cards using HTML, CSS and Bootstrap.

Steps for Implementation:

1. Create workspace in Visual studio code
2. Use basic HTML tags
3. Use “CSS Navbar” for creation of dashboard
4. A navigation bar with user settings;
5. A sidebar with navigation items;
6. A section to show the title, tagline, and breadcrumbs for the current page;
7. The main content area with a couple of widget cards;
8. Last of but not least footer.
9. Use bootstrap container, grid system.
10. Run using live webserver

Expected Output



Recent Articles

Article	Views	Comments	Status
Article 73	2.9k	210	Published
Article 72	1.5k	360	Published
Article 71	1.1k	150	Published
Article 70	1.2k	420	Published
Article 69	2.6k	190	Published
Article 68	1.9k	390	Published
Article 67	1.2k	580	Published

Example of admin Dashboard

Conclusion: Responsive web page design is implemented by creating admin Dashboard with side bars using HTML,CSS and Bootstrap successfully.

Assignment No.2

Title: AJAX Communication

Problem Statement: Write a JavaScript Program to get the user registration data and push to array/local storage with AJAX POST method and data list in new page.

Theory:

I. AJAX

□AJAX stands for Asynchronous JavaScript and XML.AJAX is a new technique for creat- ing better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

□Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.

□Conventional web applications transmit information to and from the sever using synchronous re- quests. It means you fill out a form, hit submit, and get directed to a new page with new infor- mation from the server.

□With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the re- sults, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server. XML is commonly used as the format for receiving server da- ta, although any format, including plain text, can be used. AJAX is a web browser technology in- dependent of web server software.

□A user can continue to use the application while the client program requests information from the server in the background.

□Intuitive and natural user interaction.

□Clicking is not required; mouse movement is a sufficient event trigger.

□Data-driven as opposed to page-driven.

AJAX is based on the following open standards –

•Browser-based presentation using HTML and Cascading Style Sheets (CSS).

•Data is stored in XML format and fetched from the server.

•Behind-the-scenes data fetches using XMLHttpRequest objects in the browser.

•JavaScript to make everything happen

AJAX cannot work independently. It is used in combination with other technologies to create interactive webpages.

•XMLHttpRequest

– JavaScript object that performs asynchronous interaction with the server.

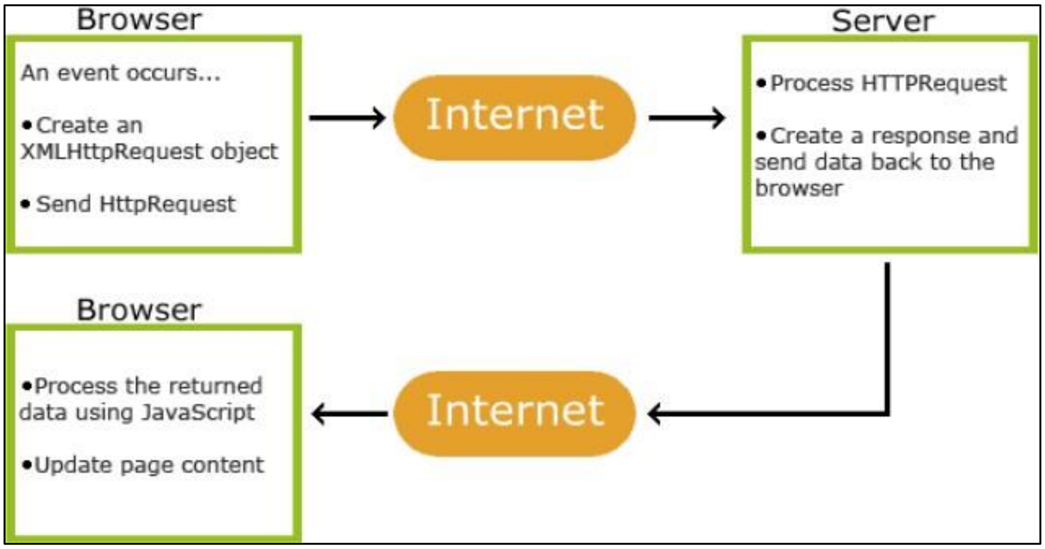

Browser
An event occurs...
•Create an XMLHttpRequest object
•Send HttpRequest
Internet
Server
•Process HTTPRequest
•Create a response and send data back to the browser
Browser
•Process the returned data using JavaScript
•Update page content
Internet

Fig 1. How AJAX works

AJAX – Events : onreadystatechange Event Properties

Property	**Description**
onReadyStateChange	It is called whenever readystate attribute changes. It must not be used with synchronous requests.
readyState	Represents the state of the request. It ranges from 0 to 4. • 0: request not initialized (open() is not called.) • 1: server connection established (open is called but send() is not called.) • 2: request received (send() is called, and headers and status are available.) • 3: processing request (Downloading data; responseText holds the data.) • 4: request finished and response is ready (The operation is completed fully.)
status	**200: "OK"** **403: "Forbidden" 404: "Page not found"**

XMLHttpRequest object properties

Property	**Description**
readyState	An integer from 0. . .4. (0 means the call is uninitialized, 4 means that the call is complete.)
onreadystatechange	Determines the function called when the objects readyState changes.
responseText	Data returned from the server as a text string (read-only).
responseXML	Data returned from the server as an XML document object (read-only).
status	HTTP status code returned by the server
statusText	HTTP status phrase returned by the server

XMLHttpRequest object methods

Method	**Description**
open('method', 'URL', asyn)	Specifies the HTTP method to be used (GET or POST as a string, the target URL, and whether or not the request should be handled asynchronously (asyn should be true or false, if omitted, true is assumed).
send(content)	Sends the data for a POST request and starts the request, if GET is used you hould call send(null).
setRequestHeader('x','y')	Sets a parameter and value pair x=y and assigns it to the header to be sent with the request.
getAllResponseHeaders()	Returns all headers as a string.
getResponseHeader(x)	Returns header x as a string.
abort()	Stops the current operation.

Sample code :

ajaxcommunication.html

```
<html>
<body>
<div id="xyz">
        Hello Friends <br> Welcome to
        Pune!!!!!<br>
<button type="button" onclick="load()">Submit
</button>
</div>
<script>
        function load(){
           var req=new XMLHttpRequest()
           req.onreadystatechange=function() {
              if(this.readyState == 4 && this.status ==
                 200){ document.getElementById("xyz").innerHTML=this.responseText
              }
           }
           req.open('GET','data.txt',true)req.send()
                        }
                </script>

</body>
</html>
```

Data.txt

```
I am enjoying learning JavaScript!!!!!!
```

II. JavaScript

- Loosely typed scripting language.
- JavaScript function is called when an event occurs in a page.
- Glue for the whole AJAX operation.

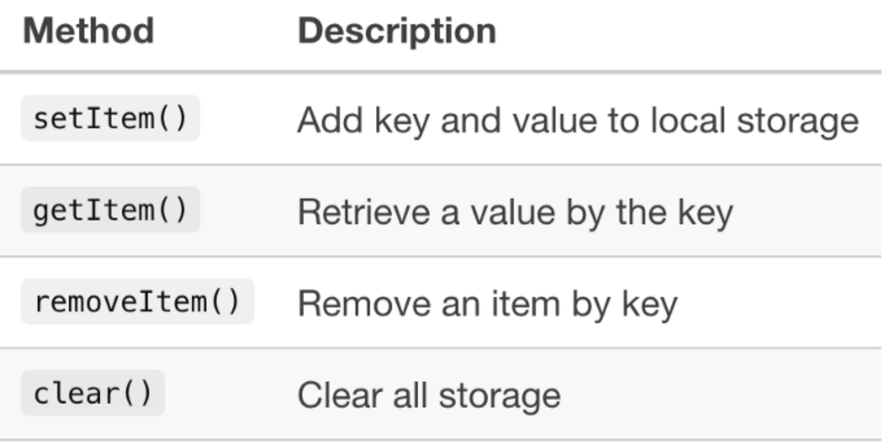
How To Use Local Storage With JavaScript

- Local storage is data stored locally in a client’s browser. It is also pure JavaScript.
- There are two types of storage you can use:

Local storage: Data that persists even if you refresh the page or close the browser.

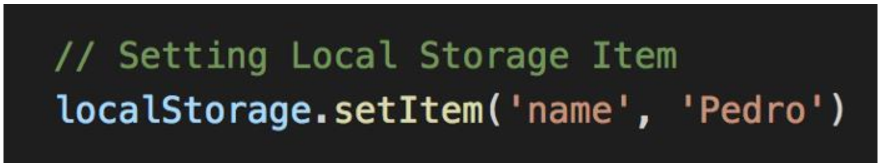
Session storage: Data that will get cleared when the browser is closed.

Methods for Local Storage



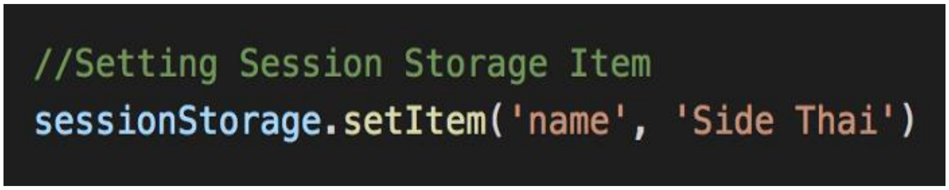
Method	Description
`setItem()`	Add key and value to local storage
`getItem()`	Retrieve a value by the key
`removeItem()`	Remove an item by key
`clear()`	Clear all storage

Setting data into local storage:



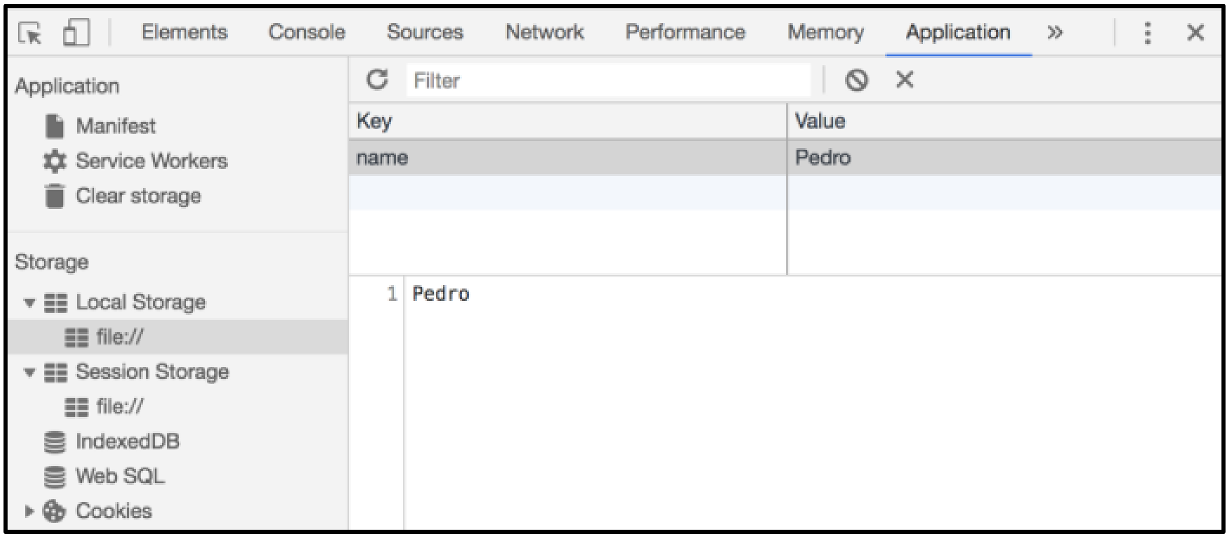
```
// Setting Local Storage Item
localStorage.setItem('name', 'Pedro')
```

Setting data into session storage:



```
//Setting Session Storage Item
sessionStorage.setItem('name', 'Side Thai')
```

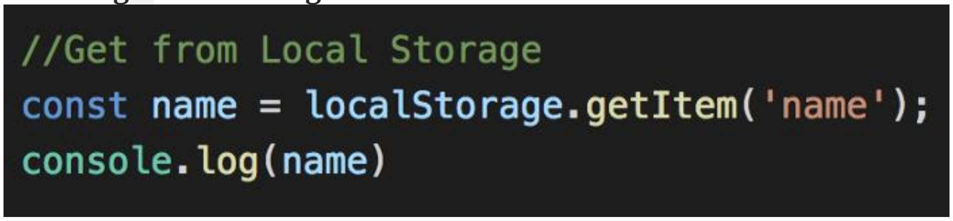
If you go into your console in your browser (CMD+Option+J) and go into the Application tab you can see your data.



Key	Value
name	Pedro

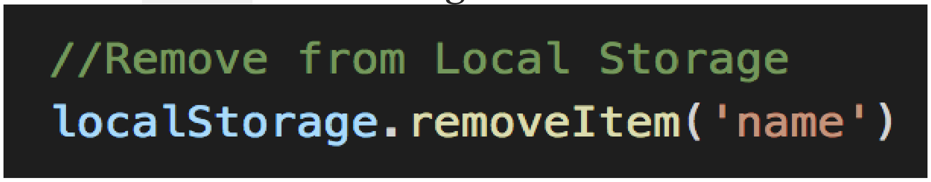
```
1 Pedro
```

We can get from storage:



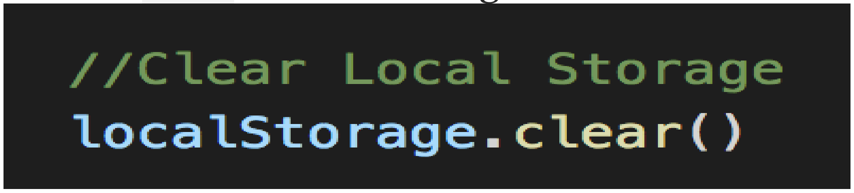
```
//Get from Local Storage
const name = localStorage.getItem('name');
console.log(name)
```

We can `remove` from storage:



```
//Remove from Local Storage
localStorage.removeItem('name')
```

We can `clear` all from storage:



```
//Clear Local Storage
localStorage.clear()
```

Problem Statement: Write a JavaScript Program to get the user registration data and push to array/local storage with AJAX POST method and data list in new page.

<u>Steps for implementation of assignment</u>

1. Create user registration form [Student/employee/patient] using Bootstrap. Try to use different form controls for taking input. [index.html]

2. Create "Add data" button for a form.

3. Write a jQuery code on button event for storing form data in to local storage using AJAX ["storeData.js" file in script tag]

4. Write a jQuery code for getting a data from local storage and display the same in "dis- playData.html" file. Use "DisplayData.js script file for writing jQuery code. Embed in script tag

Link for reference: https://betterprogramming.pub/how-to-use-local-storage-with-javascript-9598834c8b72

Conclusion: registration form data is successfully stored in local storage and retrieved; and displayed in new web page using AJAX

Assignment 3

Title: Create version control account on GitHub and using Git commands to create repository and push your code to GitHub.

Theory:

What is Git?

It is a popular version control system. It was created by Linus Torvalds in 2005, and has been hen.

What is version control?

Version control allows you to keep track of your work and helps you to easily explore the changes you have made, be it data, coding scripts, notes, etc. You are probably already doing some type of version control, if you save multiple files, such as Dissertation_script_25thFeb.R, Dissertation_script_26thFeb.R, etc. This approach will leave you with tens or hundreds of similar files, making it rather cumbersome to directly compare different versions, and is not easy to share among collaborators. With version control software such as Git, version control is much smoother and easier to implement. Using an online platform like Github to store your files means that you have an online back up of your work, which is beneficial for both you and your collaborators.

1. It is used for:

- Tracking code changes
- Tracking who made changes
- Coding collaboration

2. What does Git do?

- Manage projects with Repositories
- Clone a project to work on a local copy
- Control and track changes with Staging and Committing
- Branch and Merge to allow for work on different parts and versions of a project
- Pull the latest version of the project to a local copy
- Push local updates to the main project

3. Working with Git

- Initialize Git on a folder, making it a Repository
- Git now creates a hidden folder to keep track of changes in that folder
- When a file is changed, added or deleted, it is considered modified
- You select the modified files you want to Stage
- The Staged files are Committed, which prompts Git to store a permanent snapshot of the files
- Git allows you to see the full history of every commit.
- You can revert back to any previous commit.

- Git does not store a separate copy of every file in every commit, but keeps track of changes madein each commit!

4. Why Git?

- Over 70% of developers use Git!
- Developers can work together from anywhere in the world.
- Developers can see the full history of the project.
- Developers can revert to earlier versions of a project.

5. What is GitHub

- Git is not the same as GitHub.
- GitHub makes tools that use Git.
- GitHub is the largest host of source code in the world, and has been owned by Microsoft since2018.

Steps for Implementation to Push and PULL version control repository to GitHub

Step No	**Command**	**Description**
1	Git Installation	Download Git from the website: https://www.git-scm.com/
2	Command line >git –version	If Git is installed, it should show something like gitversion X.Y
3	git config --global user.name "w3schools-test" git config --global user.email "test@w3schools.com"	Configure Git Change the user name and e-mail address to your own
4	mkdir myprojectcd myproject	Creating Git Folder
5	git init	Initialize Git Initialized empty Git repository in /Users/user/myproject/.git/
6	git status	To check the status
7	gitadd index.html	Add file to staging environment
8	gitadd --all	add all files in the current directory to the Staging Environment:
9	git commit -m "First releaseof Hello World!"	The committ command performs a commit, and the -m "message" adds a message.
10	git commit -a -m "Updated index.html with a new line"	Skips staging environment
11	git log	To view the history of commits for a repository, you can use the log command
12	git *command* -help	See all the available options for the specific command

13	git help --all	See all possible commands
14	git commit -help	See help for specific command
15	git branch hello-world-images	a branch is a new/separate version of the main repository. This command creates a new branchhello-world-images
16	git checkout hello-world-images	checkout is the command used to check out/ move to a branch
17	git checkout master	Used to switch between branches
18	https://github.com/	Create a new account on github
19		Create a Repository on GitHub
20	git remote add origin https://github.com/w3schools-test/hello-world.git	Push Local Repository to GitHub
21	git push --set-upstream originmaster	push master branch to the origin url,
22		go back into GitHub and see that the repository has been updated:
23	git fetch origin	fetch gets all the change history of a tracked branch/repo
24	git merge origin/master	merge combines the current branch, with a specified branch.
25	git pull origin	pull is a combination of fetch and merge It is used to pull all changes from a remote repository into the branch you are working on.

##Create a new repository on the command line

```
touch README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin git@github.com:alexpchin/<reponame>.git

git push -u origin master
```

##Push an existing repository from the command line

```
git remote add origin git@github.com:alexpchin/<reponame>.git
git push -u origin master
```

Video Link for reference: https://www.youtube.com/watch?v=7JeL-3sk3F4

Conclusion: Version control account on GitHub is created and using Git commands repository is created successfully.

Assignment 4

Title: Develop web application using front end as Angular

Problem Statement: Create an Angular application which will do following actions: Register User, Login User,Show User Data on Profile Component

Theory:

I. Angular

- Angular was introduced to create Single Page applications. This framework brings structure and consistency to web applications and provides excellent scalability and maintainability.
- Angular is an open-source, JavaScript framework wholly written in TypeScript. It uses HTML's syntax to express your application's components clearly.
- Angular is an open-source JavaScript MVC framework for the web applications.
- It extends the HTML and makes it dynamic.
- A component-based framework for building scalable web applications
- A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, and more.
- A suite of developer tools to help you develop, build, test, and update your code

Difference between Angular JS and Angular

Feature	AngularJS	Angular
Language	JavaScript	TypeScript
Architecture	Supports Model-View-Controller design	Uses components and directives
Mobile support	Not supported by mobile browsers	Supports all popular mobile browsers
Dependency Injection	Doesn't support	Supports
Routing	@routeProvider is used to provide routing information	@Route configuration is used to define routing information

Management	Difficult to manage with an increase in source code size	Better structured, easy to create and manage bigger applications

II. Angular Components

Components are the building blocks that compose an application. A component includes a TypeScript class with a @Component() decorator, an HTML template, and styles. The @Component() decorator specifies the following Angular-specific information:

- A CSS selector that defines how the component is used in a template. HTML elements in your template that match this selector become instances of the component.
- An HTML template that instructs Angular how to render the component
- An optional set of CSS styles that define the appearance of the template's HTML elements

Minimal code for component

```
import { Component } from '@angular/core';

@Component({

  selector: 'hello-world',

  template: `

    <h2>Hello World</h2>

    <p>This is my first component!</p>

`

})

export class HelloWorldComponent {

  // The code in this class drives the component's behavior.

}
```

III. How to install Angular on your local system? Setup

- **Requirement**

1. Node.js- Angular requires an **active LTS or maintenance LTS version** of Node.js.

2. npm package manager- Angular, the Angular CLI, and Angular applications depend on npm packages for many features and functions

IV. Install the Angular CLI

1. Open terminal window and type following command

```
npm install -g @angular/cli
```

V. Create a workspace and initial application

1. Run the CLI command ng new and provide the name my-app, as shown here:
   ng new my-app
2. The ng new command prompts you for information about features to include in the initial app. Accept the defaults by pressing the Enter or Return key.
3. The CLI creates a new workspace and a simple Welcome app, ready to run.

VI. Run the application

The Angular CLI includes a server, for you to build and serve your app locally.

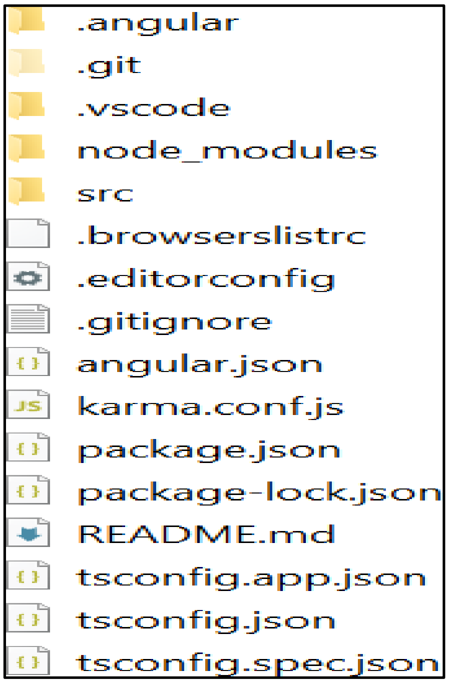
1. Navigate to the workspace folder, such as my-app.
2. Run the following command:
   my-app

ng serve –open

The ng serve command launches the server, watches your files, and rebuilds the app as you make changes to those files.

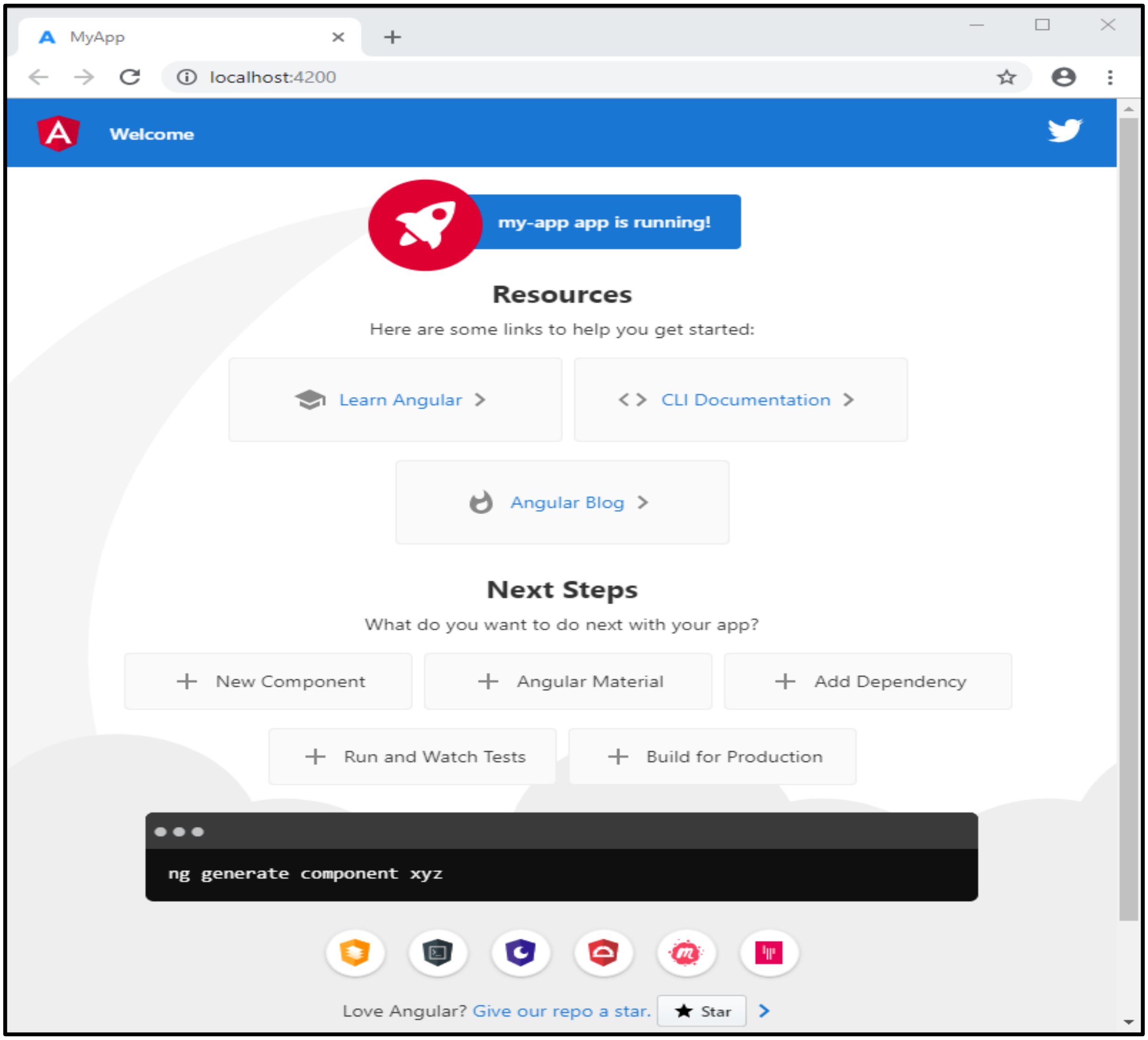
The --open (or just -o) option automatically opens your browser to http://localhost:4200/.

The project will be created as directory structure below –



```
.angular
.git
.vscode
node_modules
src
.browserslistrc
.editorconfig
.gitignore
angular.json
karma.conf.js
package.json
package-lock.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
```

If your installation and setup was successful, you should see a page similar to the following.



my-app app is running!

Resources

Here are some links to help you get started:

Next Steps

What do you want to do next with your app?

```
ng generate component xyz
```

Love Angular? Give our repo a star.

Steps for Implementation

1. **Open folder src/app**

Modify app.module.ts for form application

import { NgModule } from'@angular/core';

import { BrowserModule } from'@angular/platform-browser';

import { AppRoutingModule } from'./app-routing.module'; import { AppComponent } from'./app.component';

import {FormsModule} from'@angular/forms'

@NgModule({ declarations: [AppComponent

],

```
imports: [ BrowserModule, AppRoutingModule, FormsModule,

],

providers: [],

bootstrap: [AppComponent]

})

exportclassAppModule { }
```

2. **Open app.component.html**

Write html code for form (representative code is mentioned here, modify for multiple inputs)

```
<h1>Simple Form</h1>

<form#simpleForm = "ngForm"(ngSubmit) = "getValues(simpleForm.value)">

<inputtype ="text"ngModelname = "user"placeholder = "Enter Name">

<br><br>

<inputtype ="text"ngModelname = "age"placeholder = "Enter age">

 <br><br>

<inputtype ="text"ngModelname = "city"placeholder = "Enter city">

<br><br>

<button>Get user value</button>

</form>
```

3. **Make changes in app.component.ts**

```
import { Component } from'@angular/core';

@Component({ selector:'app-root',

templateUrl:'./app.component.html', styleUrls: ['./app.component.css']

})

exportclassAppComponent { title = 'AngProj1'; getValues(val:any)

{

console.log(val);

}

}
```

Here getValue() function which is called in form file is defined. You can check inputted values through form in console.

4. Build application

1. Use Angular CLI command ng serve -o to build an application. The -o indicates to open it automatically in the default browser.

2. Use NPM command ‘npm start’ to build an application http://localhost:4200 to see the application home page.

3. Open the terminal in VS Code from menu Terminal -> New Terminal, and type ng serve -o command and press enter,

You can send the form contents from console to other page. On the basis of above implementation, you can design login user, show user data.

Conclusion: Angular application is developed successfully.

Assignment 5

5-A

Title: - Create a Node.JS Application.

Problem Statement: Create a Node.JS Application which serves a static website.

Theory:

Installation: Node.js (site – Node.js), Express.Js(installed through cmd)

➢ Node.js overview

In basic terms, Node is an open source cross-platform library for server-side programming that permits clients to develop web applications rapidly. With Node, we can execute JavaScript applications or network applications. Its basic modules are engraved in JavaScript.

It is generally utilized for server applications in real-time. Node.js permits JavaScript to execute locally on a machine or a server.

Node.js gives numerous systems to utilize. One of such structures is Express.js. It is more valuable and mainstream than the different structures of Node.js.

➢ Features of Node.js

•Versatility: Node is incredibly adaptable as the server reacts in a non-blocking way.

•Zero Buffering: Applications yield the measurements in enormous pieces. This gives the advantage of 'No buffering' to developers.

•Network: Node.js upholds an open-source community. This is the main explanation that numerous glorious modules have been added to Node.js applications over time.

•Occasion driven Input and output: APIs of Node.js are non-blocking, meaning that the server won't wait for the arrival of information from an API. Rather, it will move to another API.

➢ Advantages of Node.js

•Easy to learn: NodeJs quite simple for developers to utilize and learn. Learning Node.js is less difficult than React.

•Better Performance: Node.js takes the code of JavaScript via Google's V8 JavaScript engine. The main advantage of this process is that it complies with the JavaScript code directly into the machine code

•Freedom: Node.js offers a lot of freedom when it comes to development. There are generally less constraints with Node.

•Extended support for tools: Another advantage of Node.js is that developers have more community support.

•Extensible: Node.js is known to be quite extensible. You can utilize JSON to give the degree to trade of information between the web server and the client.

•**Scalability:** Node.js makes it simple to scale applications in horizontal as well as vertical directions. The applications can be scaled even by the option of extra hubs to the current framework.

➢ Limitations of Node.js

•Programming interface isn’t steady: The Application Programming Interface (API) of Node can be challenging to work with. It changes regularly and doesn’t remain stable.

•No strong library support system: JavaScript does not hold a strong library system. This limits the developers to implement even common programming tasks using Node.js.

•Programming model is not synchronous: Many developers find this programming model tougher in comparison to linear blocking I/O programming.

➢ Express.js

"Express is a fast, opinionated minimalist web framework for Node.js" - official web site: Expressjs.com

Express.js is a web application framework for Node.js. It provides various features that make web application development fast and easy which otherwise takes more time using only Node.js.

Express.js is based on the Node.js middleware module called connect which in turn uses http module. So, any middleware which is based on connect will also work with Express.js.

Advantages of Express.js

1. Makes Node.js web application development fast and easy.

2. Easy to configure and customize.

3. Allows you to define routes of your application based on HTTP methods and URLs.

4. Includes various middleware modules which you can use to perform additional tasks on request and response.

5. Easy to integrate with different template engines like Jade, Vash, EJS etc.

6. Allows you to define an error handling middleware.

7. Easy to serve static files and resources of your application.

8. Allows you to create REST API server.

9. Easy to connect with databases such as MongoDB, Redis, MySQL

Steps for implementation

1 .Install Node.js

2. Setting up express.js

3. Structuring files

4. Creating your express server

5. Servicing your static files

6. Building your web page
7. Running your project

Open Node.js command terminal &run the following in your terminal-

<u>Setting up Express.JS</u>

1. Create a new directory for your project - mkdir your-project-name

2. Change into your new directory - cd your-project-name

3. Initialize a new Node project with defaults. This will set a package.json file to access your dependencies: npm init -y

4. Create your entry file, index.js. This is where you will store your Express server: if you are working on Linux, you can run : touch index.js. if you are working on windows, you can edit in VSCode

5. Install Express as a dependency : npm install express –save

6. Edit package.json. Within your package.json, update your start script to include node and your index.js file.

Let express-static-file-tutorial is your project name Package.json

```
{

"name": "express-static-file-tutorial", "version": "1.0.0",

"description": "",

"main": "index.js", "scripts": {

"start": "node index.js" // change start value as node index.js

},                          // This will allow you to use the npm start command in your terminal to launch

"keywords": [],        your express server "author": "Paul Halliday",

"license": "MIT"

}
```

Structuring Your Files

To store your files on the client-side, create a public directory and include an index.html file express-static-file-tutorial

```
|- index.js

|- public
|- index.html
```

Creating Your Express Server

Edit index.js file

Index.js

```
const express = require('express'); const app = express();

const PORT = 3000;

app.use(express.static('public')); // represents application is serving static webpage in public directory

app.get('/', (req, res) => { res.send('Hello World!');

});

app.listen(PORT, () => console.log(`Server listening on port: ${PORT}`));
```

First of all, import the Express.js module.

In the above example, we imported Express.js module using require() function. The express module returns a function. This function returns an object which can be used to configure Express application (app in the above example).

The app object includes methods for routing HTTP requests, configuring middleware, rendering HTML views and registering a template engine.

The app.listen() function creates the Node.js web server at the specified host and port. It is identical to Node's http.Server.listen() method. Instead of Get(), post(), put() and delete() methods can be used.

Building Your Web Page – Client side

Navigate to your index.html file in the public directory. Populate the file with body and image elements:

[label index.html]

```
<html>

<head>

<title>Hello World!</title>

</head>

<body>

<h1>Hello, World!</h1>

<img src="shark.png" alt="shark"> //download & store image in public directory

</body>

</html>
```

(Instead of building Hello world application, building applications like student’s Registration form/main page of website is recommended)

Running Your Project

In your terminal, launch your Express project

```
npm start
```

It will display

```
Server listening on port : 3000
```

Open your web browser, and navigate to http://localhost:3000.

Conclusion: Node.js application is created successfully.

Part 5- B

Title: Create four API using Node.JS, ExpressJS and MongoDB for CURD Operations

Problem Statement: Create four API using Node.JS, ExpressJS and MongoDB for CURD Operations

- Node.js and MongoDB Connectivity

- Node.js can be used in database applications. One of the most popular NoSQL database is MongoDB.

- Express is one of the most popular web frameworks for Node.js that supports routing, middleware, view system…
- Mongoose is a promise-based Node.js ODM for MongoDB that provides a straight-forward, schema-based solution to model our application data along with built-in type casting, validation, query building, business logic hooks…

➢ **MongoDB**

We can download a free MongoDB database at https://www.mongodb.com.

Or get started right away with a MongoDB cloud service at https://www.mongodb.com/cloud/atlas.

1. **Install MongoDB Driver**

Access a MongoDB database with Node.js.

To download and install the official MongoDB driver, open the Command Terminal and execute the following:

Download and install mongodb package:

```
C:\Users\Your Name>npm install mongodb
```

Now you have downloaded and installed a mongodb database driver.

Node.js can use this module to manipulate MongoDB databases:

```
var mongo = require('mongodb');
```

2. **Creating a Database**

To create a database in MongoDB, start by creating a MongoClient object, then specify a connection URL with the correct ip address and the name of the database you want to create.

MongoDB will create the database if it does not exist, and make a connection to it.

Example

Create a database called "mydb":

```
var MongoClient = require('mongodb').MongoClient;

var url = "mongodb://localhost:27017/mydb";
```

```
MongoClient.connect(url, function(err, db) {

  if (err) throw err;
  console.log("Database created!");
  db.close();
});
```

Save the code above in a file called "demo_create_mongo_db.js" and run the file:

Run "demo_create_mongo_db.js"

```
C:\Users\Your Name>node demo_create_mongo_db.js
```

Which will give you this result:

```
Database created!
```

3. Collection

➢ Creating a Collection

A collection in MongoDB is the same as a table in MySQL

To create a collection in MongoDB, use the createCollection() method:

Example

Create a collection called "customers":

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  dbo.createCollection("customers", function(err, res) {
    if (err) throw err;
    console.log("Collection created!");
    db.close();
  });
});
```

Save the code above in a file called "demo_mongodb_createcollection.js" and run the file:

Run "demo_mongodb_createcollection.js"

```
C:\Users\Your Name>node demo_mongodb_createcollection.js
```

Which will give you this result:

```
Collection created!
```

4. Insert Into Collection

To insert a record, or document as it is called in MongoDB, into a collection, we use the insertOne() method.

A document in MongoDB is the same as a record in MySQL

The first parameter of the insertOne() method is an object containing the name(s) and value(s) of each field in the document you want to insert.

It also takes a callback function where you can work with any errors, or the result of the insertion:

Example

Insert a document in the "customers" collection:

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var myobj = { name: "Company Inc", address: "Highway 37" };
  dbo.collection("customers").insertOne(myobj, function(err, res) {
    if (err) throw err;
    console.log("1 document inserted");
    db.close();
  });
});
```

Save the code above in a file called "demo_mongodb_insert.js" and run the file:

Run "demo_mongodb_insert.js"

```
C:\Users\Your Name>node demo_mongodb_insert.js
```

Which will give you this result:

```
1 document inserted
```

Insert Multiple Documents

To insert multiple documents into a collection in MongoDB, we use the insertMany() method.

The first parameter of the insertMany() method is an array of objects, containing the data you want to insert.

➢ Step by step procedure to build Node.js Restful API for CRUD operations using Express, Mongoose with MongoDB database.

Step1: First, Install MongoDB on machine

Step2: **Node.js MongoDB Rest CRUD API overview**

We will build Rest Apis that can create, retrieve, update, delete

First, we start with an Express web server. Next, we add configuration for MongoDB database, create **Tutorial model with Mongoose**, write the controller. Then we define routes for handling all CRUD operations (including custom finder).

The following table shows overview of the Rest APIs that will be exported:

Methods	Urls	Actions
GET	api/tutorials	get all Tutorials
GET	api/tutorials/:id	get Tutorial by id
POST	api/tutorials	add new Tutorial
PUT	api/tutorials/:id	update Tutorial by id
DELETE	api/tutorials/:id	remove Tutorial by id
DELETE	api/tutorials	remove all Tutorials
GET	api/tutorials/published	find all published Tutorials
GET	api/tutorials?title=[kw]	find all Tutorials which title contains 'kw'

Step 3: Create Node.js App

First, we create a folder:

```
$ mkdir nodejs-express-mongodb
$ cd nodejs-express-mongodb
```

Next, we initialize the Node.js App with a *package.json* file:

```
npm init

name: (nodejs-express-mongodb)
version: (1.0.0)
description: Node.js Restful CRUD API with Node.js, Express and MongoDB
```

```
entry point: (index.js) server.js
test command:
git repository:
keywords: nodejs, express, mongodb, rest, api
author: bezkoder
license: (ISC)

Is this ok? (yes) yes
```

We need to install necessary modules: express, mongoose and cors.
Run the command:

```
npm install express mongoose cors --save
```

The *package.json* file should look like this:

```
{
  "name": "node-express-mongodb",
  "version": "1.0.0",
  "description": "Node.js Restful CRUD API with Node.js, Express and MongoDB",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "nodejs",
    "express",
    "rest",
    "api",
    "mongodb"
  ],
  "author": "bezkoder",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.18.1",
    "mongoose": "^6.4.2"
  }
}
```

Step4: Setup Express web server

In the root folder, let’s create a new **server.js** file:

```
const express = require("express");
const cors = require("cors");

const app = express();

var corsOptions = {
  origin: "http://localhost:8081"
};

app.use(cors(corsOptions));

// parse requests of content-type - application/json
app.use(express.json());

// parse requests of content-type - application/x-www-form-urlencoded
app.use(express.urlencoded({ extended: true }));

// simple route
app.get("/", (req, res) => {
  res.json({ message: "Welcome to bezkoder application." });
});

// set port, listen for requests
const PORT = process.env.PORT || 8080;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}.`);
});
```

Step 5: import express and cors modules:

Express is for building the Rest apis

cors provides Express middleware to enable CORS with various options.

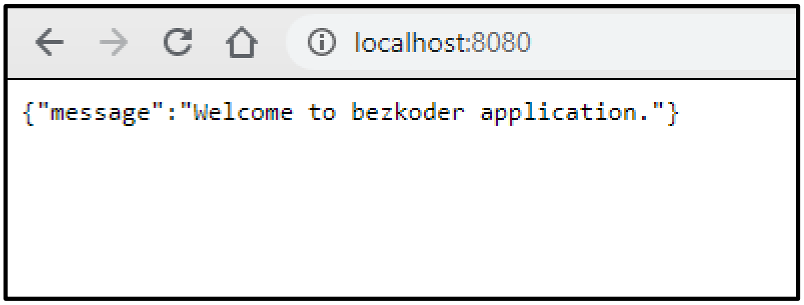
– create an Express app, then add body-parser (json and urlencoded) and cors middlewares using app.use() method. Notice that we set origin: http://localhost:8081.

– define a GET route which is simple for test.

– listen on port 8080 for incoming requests.

Now let’s run the app with command: node server.js.

Open your browser with url http://localhost:8080/, you will see:



```
{"message":"Welcome to bezkoder application."}
```

Step6: Configure MongoDB database & Mongoose

In the app folder, we create a separate “config” folder for configuration with **db.config.js** file like this:

```
module.exports = {
  url: "mongodb://localhost:27017/bezkoder_db"
};
```

Step 7: Define Mongoose

We’re gonna define Mongoose model (tutorial.model.js) also in app/models folder in the next step.

Now create app/models/index.js with the following code:

```
const dbConfig = require("../config/db.config.js");
const mongoose = require("mongoose");
mongoose.Promise = global.Promise;
const db = {};
db.mongoose = mongoose;
db.url = dbConfig.url;
db.tutorials = require("./tutorial.model.js")(mongoose);
module.exports = db;
```

Don’t forget to call connect() method in server.js:

```
...
const app = express();
app.use(...);
const db = require("./app/models");
db.mongoose
  .connect(db.url, {
    useNewUrlParser: true,
    useUnifiedTopology: true
  })
  .then(() => {
    console.log("Connected to the database!");
  })
  .catch(err => {
    console.log("Cannot connect to the database!", err);
    process.exit();
  });
```

Define the Mongoose Model

In models folder, create tutorial.model.js file like this:

```
module.exports = mongoose => {
  const Tutorial = mongoose.model(
    "tutorial",
    mongoose.Schema(
      {
        title: String,
        description: String,
        published: Boolean
      },
      { timestamps: true }
    )
  );

  return Tutorial;
};
```

Step

Step8: Create the Controller

Inside app/controllers folder, let’s create tutorial.controller.js with these CRUD functions:

create

findAll

findOne

update

delete

deleteAll

findAllPublished

```
const db = require("../models");
const Tutorial = db.tutorials;
// Create and Save a new Tutorial
exports.create = (req, res) => {
  };
// Retrieve all Tutorials from the database.
exports.findAll = (req, res) => {
  };
// Find a single Tutorial with an id
exports.findOne = (req, res) => {
  };
// Update a Tutorial by the id in the request
exports.update = (req, res) => {
};
// Delete a Tutorial with the specified id in the request
exports.delete = (req, res) => {

};
// Delete all Tutorials from the database.
exports.deleteAll = (req, res) => {
};
// Find all published Tutorials
```

```
exports.findAllPublished = (req, res) => {

  };
```

Reference: **https://www.bezkoder.com/node-express-mongodb-crud-rest-api/**

Conclusion: **API using Node.JS, ExpressJS and MongoDB for CURD Operations is created and implemented successfully**

Assignment 6

Title: Create a simple Mobile Website using jQuery Mobile.

Theory:

➢ <u>jQuery Mobile</u>

JQuery Mobile is a user interface framework, built on jQuery Core and used for developing responsive websites or applications that are accessible on mobile, tablet, and desktop devices. It uses features of both jQuery and jQueryUI to provide API features for mobile web applications. This tutorial will teach you the basics of jQuery Mobile framework. We will also discuss some detailed concepts related to jQuery Mobile.

➢ Why Use jQuery Mobile?

•It creates web applications that it will work the same way on the mobile, tablet, and desktop de- vices.

• It is compatible with other frameworks such as PhoneGap, Whitelight, etc.

• It provides a set of touch-friendly form inputs and UI widgets.

➢ Features of jQuery Mobile

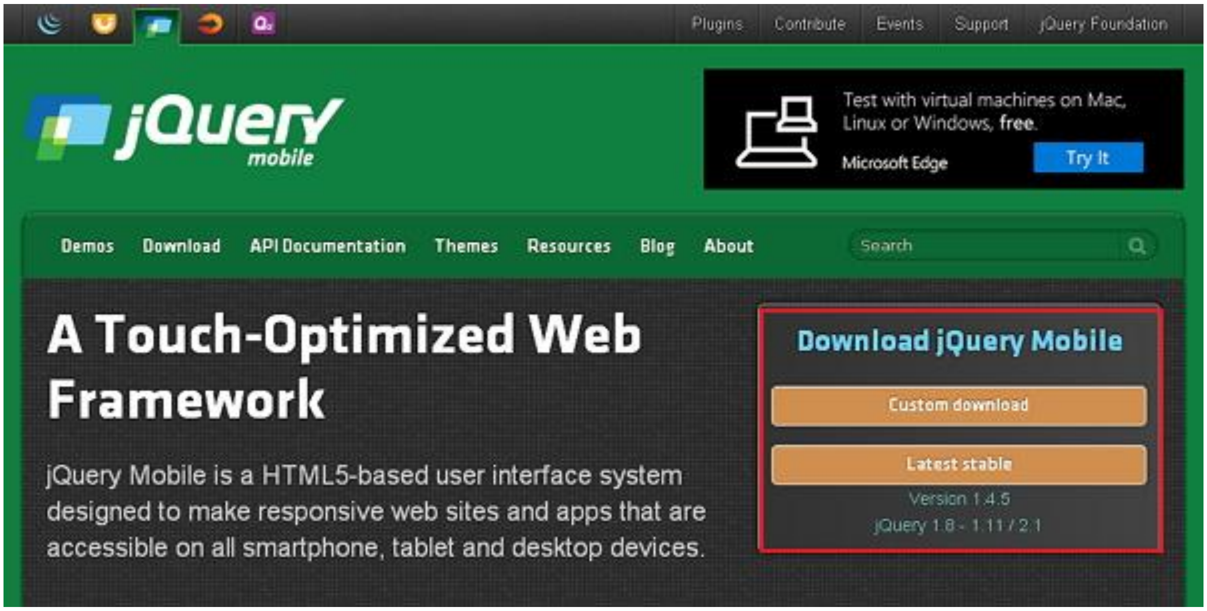
• It is built on jQuery Core and "write less, do more" UI framework.

•It is an open source framework, and cross-platform as well as cross-browser compatible.

• It is written in JavaScript and uses features of both jQuery and jQuery UI for building mobile- friendly sites.

Download jQuery Mobile

When you open the link https://jquerymobile.com/ you will see there are two options to download jQuery mobile library.



Test with virtual machines on Mac, Linux or Windows, **free**.

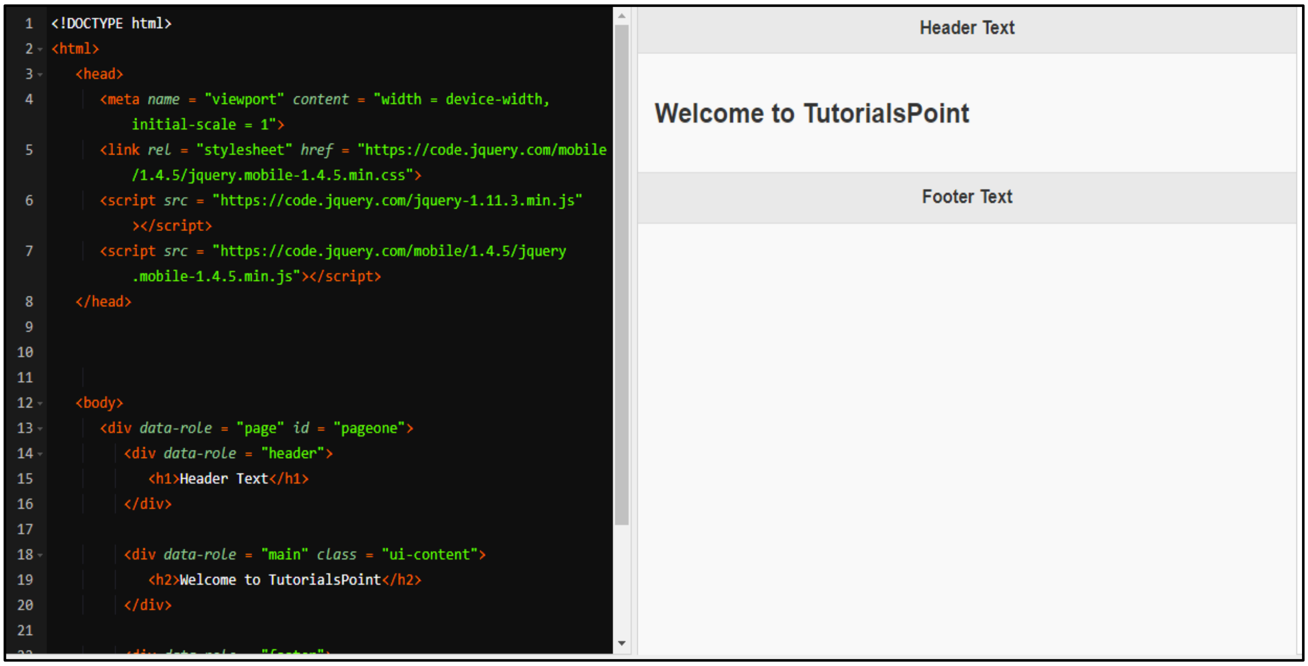
A Touch-Optimized Web Framework

jQuery Mobile is a HTML5-based user interface system designed to make responsive web sites and apps that are accessible on all smartphone, tablet and desktop devices.

- **Custom Download** − Click this button to download a customized version of library.
- **Latest Stable** − Click this button to get the stable and latest version of jQuery mobile library.

Example Code

```
<!DOCTYPE html>
<html>
  <head>
    <meta name = "viewport" content = "width = device-width, initial-scale = 1">
    <link rel = "stylesheet" href = "https://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.css">
    <script src = "https://code.jquery.com/jquery-1.11.3.min.js"></script>
    <script src = "https://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.js"></script>
  </head>


  <body>
    <div data-role = "page" id = "pageone">
      <div data-role = "header">
        <h1>Header Text</h1>
      </div>

      <div data-role = "main" class = "ui-content">
        <h2>Welcome to TutorialsPoint</h2>
      </div>

      <div data-role = "footer">
        <h1>Footer Text</h1>
      </div>
    </div>
  </body>
</html>
```



```
<!DOCTYPE html>
<html>
  <head>
    <meta name = "viewport" content = "width = device-width,
        initial-scale = 1">
    <link rel = "stylesheet" href = "https://code.jquery.com/mobile
        /1.4.5/jquery.mobile-1.4.5.min.css">
    <script src = "https://code.jquery.com/jquery-1.11.3.min.js"
        ></script>
    <script src = "https://code.jquery.com/mobile/1.4.5/jquery
        .mobile-1.4.5.min.js"></script>
  </head>



  <body>
    <div data-role = "page" id = "pageone">
      <div data-role = "header">
        <h1>Header Text</h1>
      </div>

      <div data-role = "main" class = "ui-content">
        <h2>Welcome to TutorialsPoint</h2>
      </div>

      <div data-role = "footer">
```

Header Text

Welcome to TutorialsPoint

Footer Text

<u>Above Code run and expected output</u>

- **Details of the above code are –**

1. **This code is specified inside the head element.**

```
<meta name = "viewport" content = "width = device-width, initial-scale = 1">
```

- **The viewport is used to specify (by the browser) to display the page zoom level and dimension.**
- **content = "width = device-width" is used to set the pixel width of the page or screen device.**
- **initial-scale = 1 sets the initial zoom level, when the page is loaded for the first time.**

- **Include the following CDNs**

```
<link rel = "stylesheet" href = "https://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.css">
<script src = "https://code.jquery.com/jquery-1.11.3.min.js"></script>
<script src = "https://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.js"></script>
```

2. **Content inside the <body> tag is a page displayed in the browser.**

```
<div data-role = "page">

  ...

</div>
```

data-role = "header" creates the header at the top of the page.

data-role = "main" is used to define the content of the page.

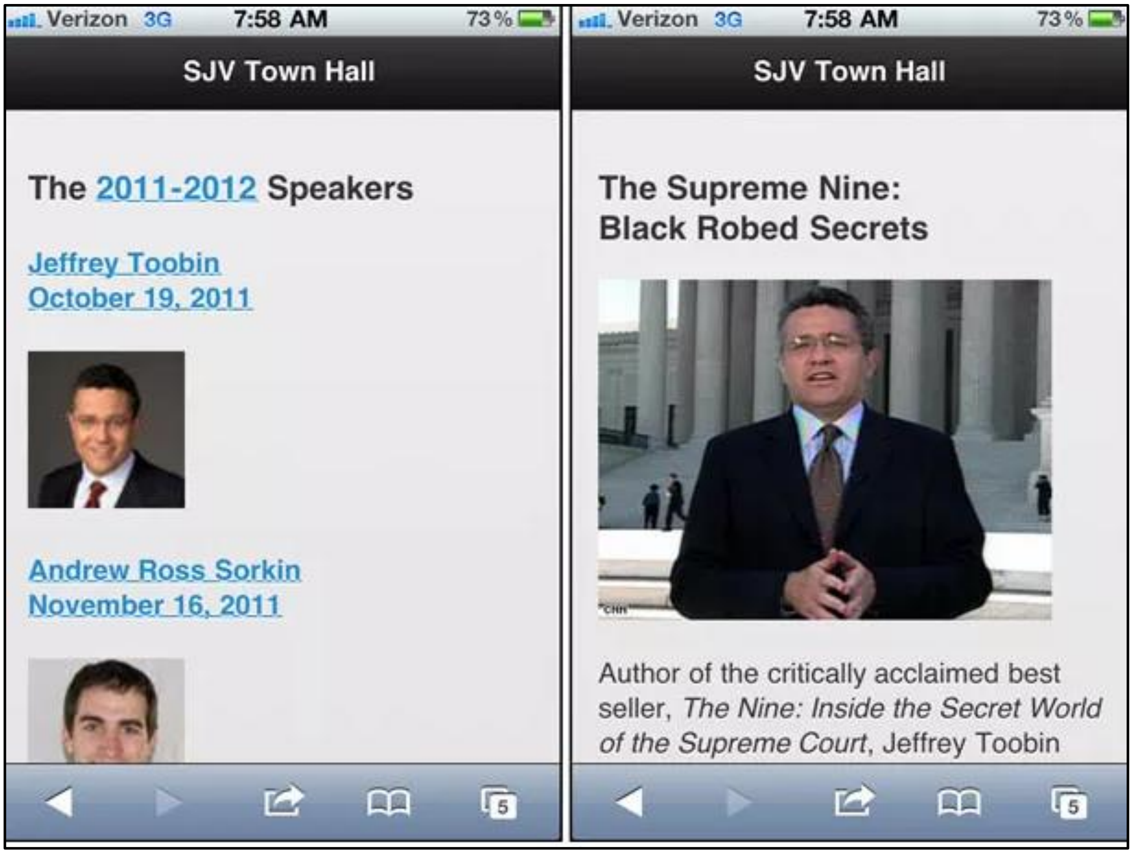
data-role = "footer" creates the footer at the bottom of the page.

class = "ui-content" includes padding and margin inside the page content.

- **The HTML for the two pages in the body of one HTML file:**

```
<div data-role="page">

<header data-role="header"><h1>SJV Town Hall</h1></header>

<section data-role="content">

<h3>The 2011-2012 Speakers</h3>

<a href="#toobin">

<h4>Jeffrey Toobin<br>October 19, 2011</h4>

<img src="images/toobin75.jpg" alt="Jeffrey Toobin"></a>

<!-- THE ELEMENTS FOR THE REST OF THE SPEAKERS -->

</section>
```

```
<footer data-role="footer"><h4>&copy; 2011</h4></footer>

</div>

<div data-role="page" id="toobin">

<header data-role="header"><h1>SJV Town Hall</h1></header>

<section data-role="content">

<h3>The Supreme Nine:<br>Black Robed Secrets</h3>

<img src="images/toobin_court.cnn.jpg" alt="Jeffrey Toobin">

<p>Author of the critically acclaimed best seller, <i>The Nine:

<!-- THE COPY CONTINUES -->

</section>

<footer data-role="footer"><h4>&copy; 2011</h4></footer>

</div>
```



The 2011-2012 Speakers

Jeffrey Toobin
October 19, 2011

Andrew Ross Sorkin
November 16, 2011

The Supreme Nine:
Black Robed Secrets

Author of the critically acclaimed best seller, *The Nine: Inside the Secret World of the Supreme Court*, Jeffrey Toobin

Above Code run and expected output for multiple pages

➢ The transitions that can be used

slide	The next page slides in from right to left.
slideup	The next page slides in from bottom to top.
slidedown	The next page slides in from top to bottom.
pop	The next page fades in from the middle of the screen.
fade	The next page fades into view.
flip	The next page flips from back to front similar to a playing card being flipped over. This transition isn’t supported on some device.

HTML that opens the page as a dialog box with the “pop” transition:

```
<a href="#toobin" data-rel="dialog" data-transition="pop">
```

HTML that opens the page with the “fade” transition:

```
<a href="#toobin" data-transition="fade">
```

➢ Create buttons

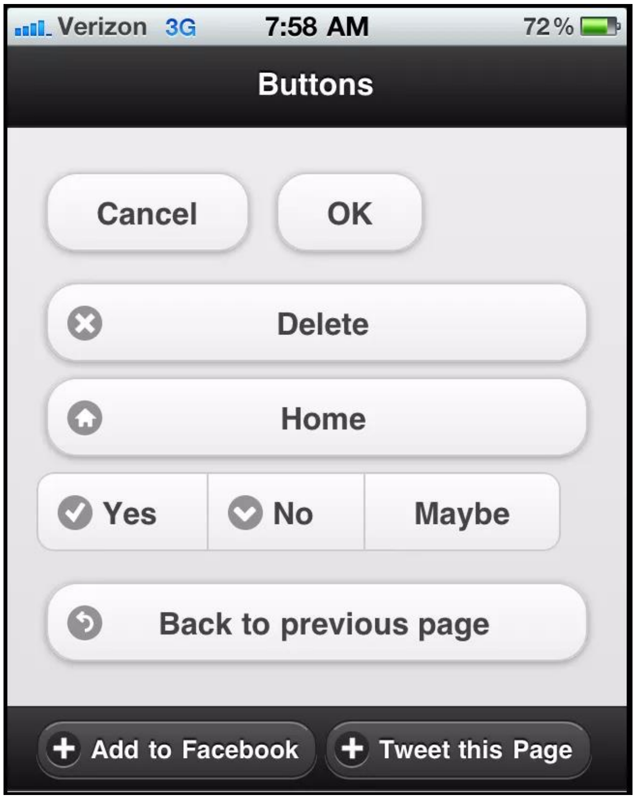
The HTML for the buttons in the section:

```
<!-- For inline buttons, set the data-line attribute to true -->

<a href="#" data-role="button" data-inline="true">Cancel</a>

<a href="#" data-role="button" data-inline="true">OK</a>

<!-- To add an icon to a button, use the data-icon attribute -->

<a href="#" data-role="button" data-icon="delete">Delete</a>

<a href="#" data-role="button" data-icon="home">Home</a>

<!-- To group buttons, use a div element with the attributes that follow -->

<div data-role="controlgroup" data-type="horizontal">

<a href="#" data-role="button" data-icon="check">Yes</a>
```

```
<a href="#" data-role="button" data-icon="arrow-d">No</a>
<a href="#" data-role="button">Maybe</a>
</div>
<!-- To code a Back button, set the data-rel attribute to back -->
<a href="#" data-role="button" dat-rel="back" data-icon="back">
Back to previous page</a>
```

The HTML for the buttons in the footer:

```
<footer data-role="footer" class="ui-bar">
<a href="http://www.facebook.com" data-role="button"
data-icon="plus">Add to Facebook</a>
<a href="http://www.twitter.com" data-role="button"
data-icon="plus">Tweet this Page</a>
</footer>
```

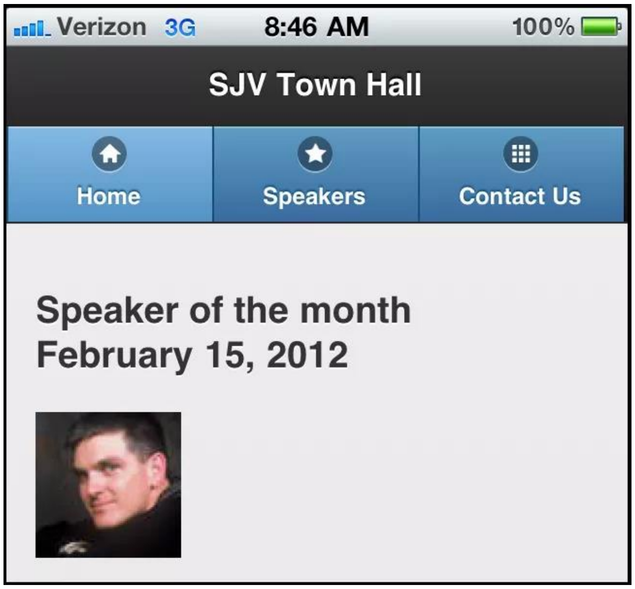


Above Code run and expected output for buttons

- **To create a navigation bar**

The HTML for the navigation bar

```
<header data-role="header">

<h1>SJV Town Hall</h1>

<div data-role="navbar">

<ul>

<li><a href="#home" class="ui-btn-active"

data-icon="home" data-theme="b">Home</a></li>

<li><a href="#speakers"

data-icon="star" data-theme="b">Speakers</a></li>

<li><a href="#contactus

data-icon="grid" data-theme="b">Contact Us</a></li>

</ul>

</div>

</header>
```



Speaker of the month
February 15, 2012

Above Code run and expected output for navigation bar

Conclusion:

Assignment No. 7

Title: Deploy web application on Cloud

Problem Statement: Deploy/Host Your web application on AWS VPC or AWS Elastic Beanstalk

Theory:

- **What is Cloud Computing?**

Cloud computing is a term referred to storing and accessing data over the internet. It doesn't store any data on the hard disk of your personal computer. In cloud computing, you can access data from a remote server.

- **What is AWS?**

The full form of AWS is Amazon Web Services. It is a platform that offers flexible, reliable, scalable, easy-to-use and, cost-effective cloud computing solutions.

AWS is a comprehensive, easy to use computing platform offered Amazon. The platform is developed with a combination of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offerings.

It is a secure cloud services platform, offering compute power, database storage, content delivery and other functionality to help businesses scale and grow.

In simple words AWS allows you to do the following things- Running web and application servers in the cloud to host dynamic websites.

- **History of AWS**

- 2002- AWS services launched
- 2006- Launched its cloud products
- 2012- Holds first customer event
- 2015- Reveals revenues achieved of $4.6 billion
- 2016- Surpassed $10 billon revenue target
- 2016- Release snowball and snowmobile
- 2019- Offers nearly 100 cloud services
- 2021- AWS comprises over 200 products and services

- **Important AWS Services**

Amazon Web Services offers a wide range of different business purpose global cloud-based products. The products include storage, databases, analytics, networking, mobile, development tools, enterprise applications, with a pay-as-you-go pricing model.

- **Applications of AWS services**

Amazon Web services are widely used for various computing purposes like:

- Web site hosting
- Application hosting/SaaS hosting
- Media Sharing (Image/ Video)
- Mobile and Social Applications
- Content delivery and Media Distribution

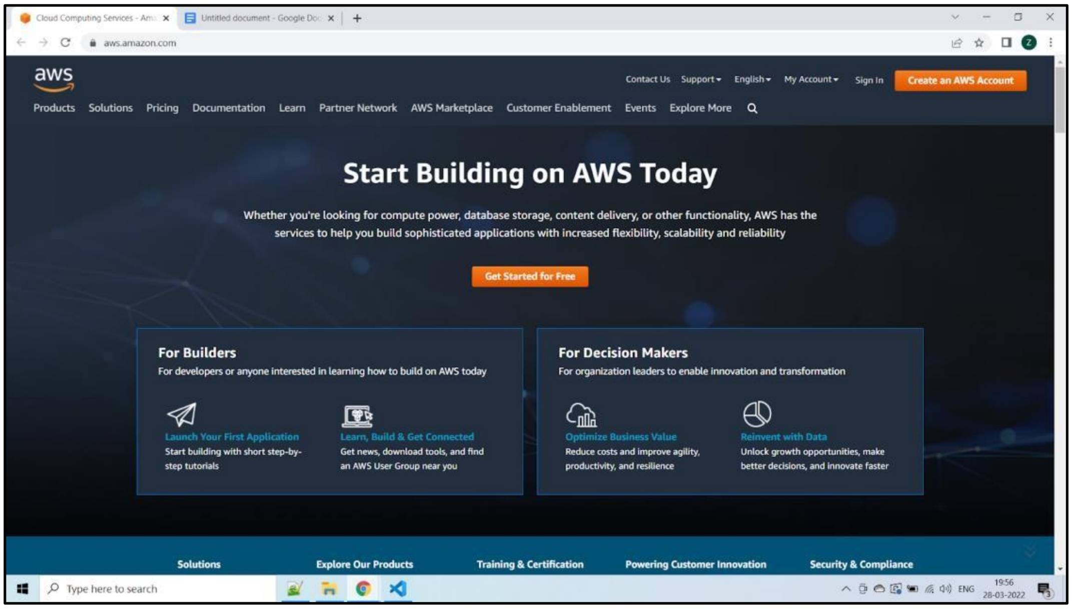
- Storage, backup, and disaster recovery
- Development and test environments
- Academic Computing
- Search Engines
- Social Networking

1. Creating an AWS Account is the first step you need to take in order to learn Amazon Web Services.

Steps to follow are as follows :

Step 1 – Visiting the Signup Page Go to https://aws.amazon.com

You should see something like below:



Start Building on AWS Today

Whether you're looking for compute power, database storage, content delivery, or other functionality, AWS has the services to help you build sophisticated applications with increased flexibility, scalability and reliability

For Builders

For developers or anyone interested in learning how to build on AWS today

Launch Your First Application

Start building with short step-by-step tutorials

Learn, Build & Get Connected

Get news, download tools, and find an AWS User Group near you

For Decision Makers

For organization leaders to enable innovation and transformation

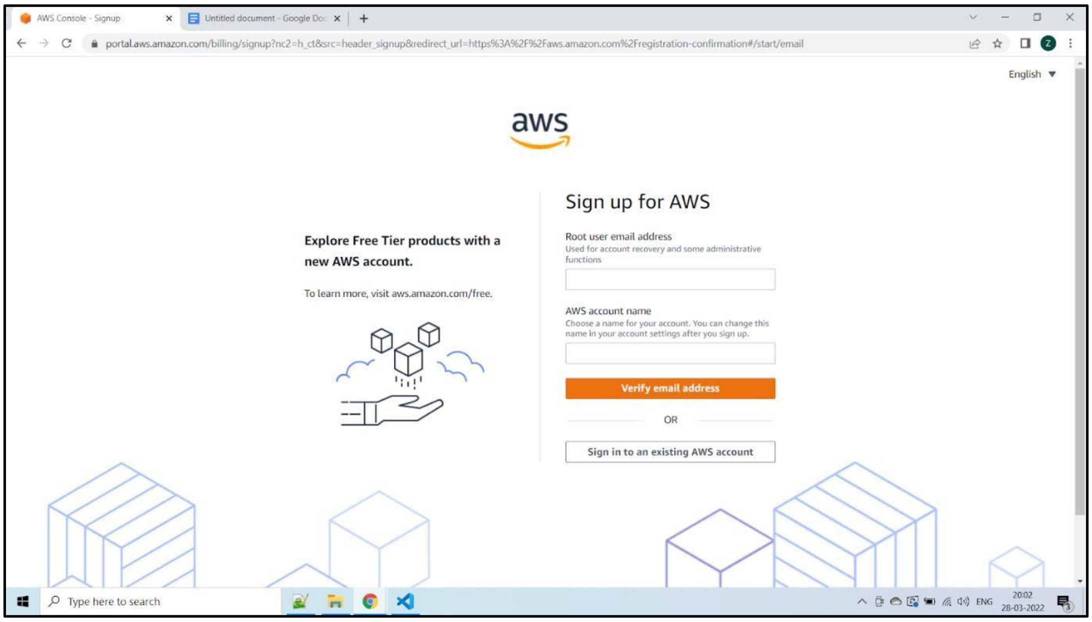
Optimize Business Value

Reduce costs and improve agility, productivity, and resilience

Reinvent with Data

Unlock growth opportunities, make better decisions, and innovate faster

In order to continue, click the Complete Sign Up button in the middle of the screen or on the top right corner of the screen. You will see the below screen.


AWS Console - Signup
Untitled document - Google Doc
portal.aws.amazon.com/billing/signup?nc2=h_ct&src=header_signup&redirect_url=https%3A%2F%2Faws.amazon.com%2Fregistration-confirmation#/start/email
English
Type here to search
ENG
20:02
28-03-2022

Explore Free Tier products with a new AWS account.

To learn more, visit aws.amazon.com/free.

Sign up for AWS

Root user email address
Used for account recovery and some administrative functions

AWS account name
Choose a name for your account. You can change this name in your account settings after you sign up.

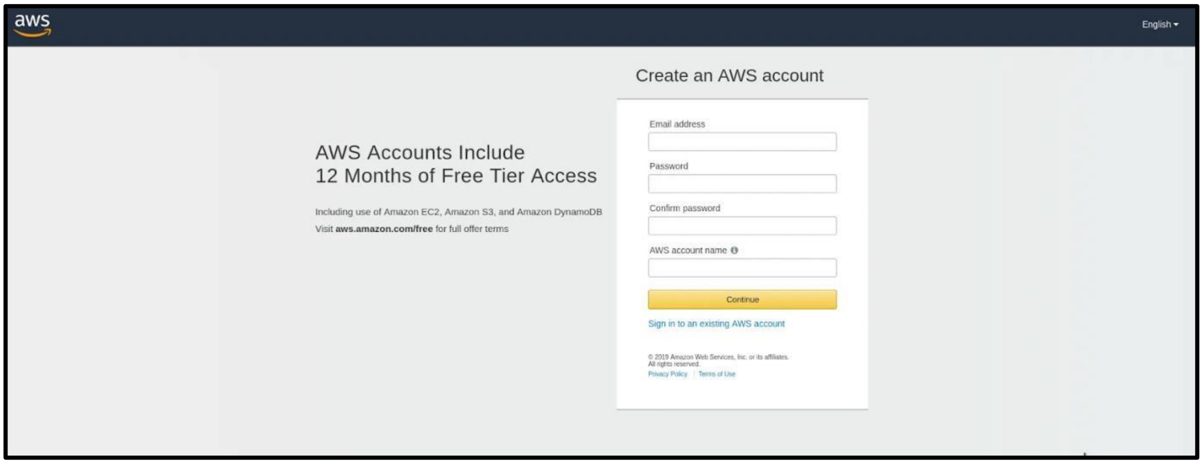
Verify email address

OR

Sign in to an existing AWS account

Step 2 – Entering User Details

After you have chosen to Create a new AWS account, you will see the below screen asking for few details.



Create an AWS account

AWS Accounts Include
12 Months of Free Tier Access

Including use of Amazon EC2, Amazon S3, and Amazon DynamoDB
Visit **aws.amazon.com/free** for full offer terms

Email address

Password

Confirm password

AWS account name

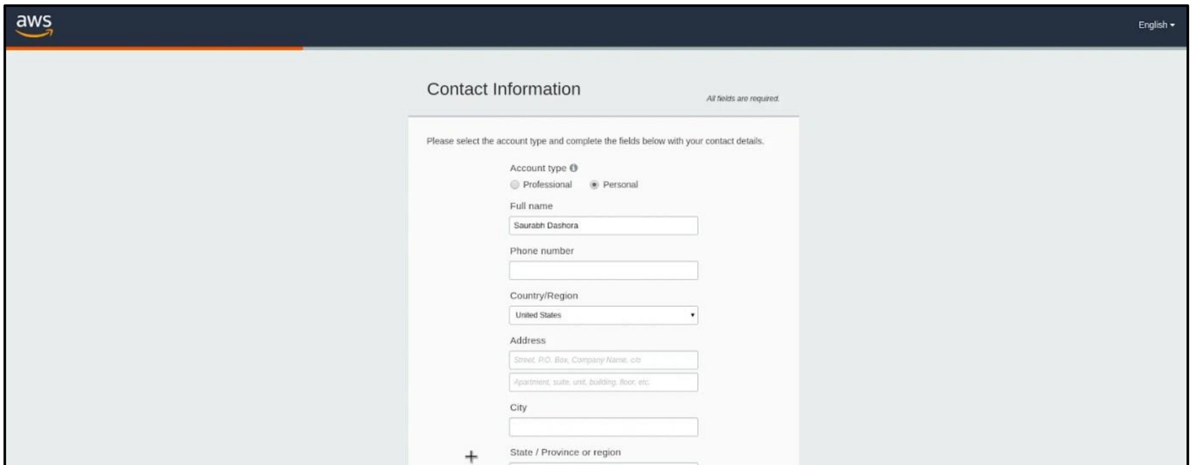
Continue

Sign in to an existing AWS account

© 2019 Amazon Web Services, Inc. or its affiliates.
All rights reserved.
Privacy Policy | Terms of Use

You can fill up the details as per your requirements and click **Continue**.

Next you will be asked to fill up your contact details such contact number, country, address and so on. You should fill them up properly because your contact number is important for further steps.



Contact Information

All fields are required.

Please select the account type and complete the fields below with your contact details.

Account type
○ Professional ● Personal

Full name
Saurabh Dashora

Phone number

Country/Region
United States

Address
Street, P.O. Box, Company Name, c/o
Apartment, suite, unit, building, floor, etc.

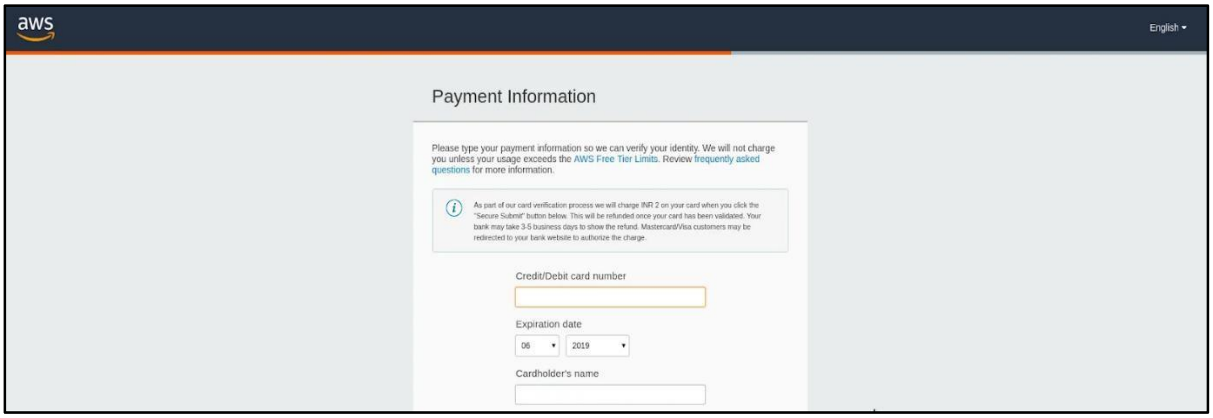
City

State / Province or region

After filling up the details, click on the **Create Account and Continue** button at the bottom of the form.

Step 3 – Filling up the Credit Card details

For **Creating an AWS Account**, you need to enter your **Credit Card** details



Payment Information

Please type your payment information so we can verify your identity. We will not charge you unless your usage exceeds the AWS Free Tier Limits. Review frequently asked questions for more information.

As part of our card verification process we will charge INR 2 on your card when you click the "Secure Submit" button below. This will be refunded once your card has been validated. Your bank may take 3-5 business days to show the refund. Mastercard/Visa customers may be redirected to your bank website to authorize the charge.

Credit/Debit card number

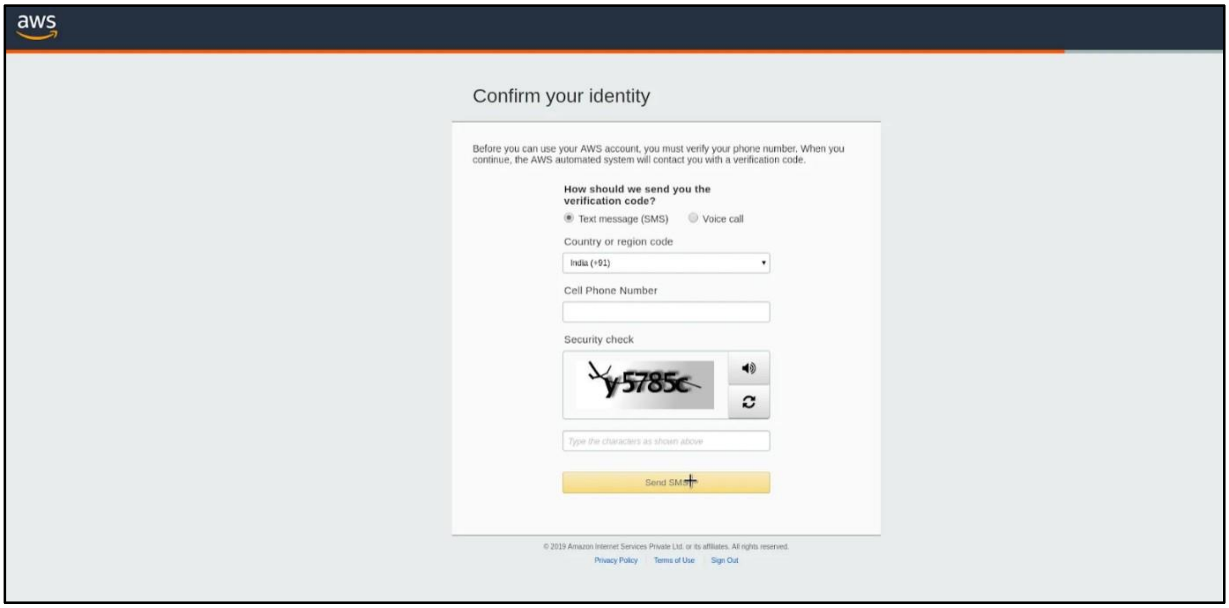
Expiration date: 06 ▾ 2019 ▾

Cardholder's name

After entering the details, click on **Secure Submit** button. It might take a while to process the request depending on your bank/credit card company servers.

Step 4 – Identity Confirmation

Once the credit card details are confirmed, you will need to complete the **Identity Confirmation** step. You will see the below screen:



Confirm your identity

Before you can use your AWS account, you must verify your phone number. When you continue, the AWS automated system will contact you with a verification code.

How should we send you the verification code?

● Text message (SMS) ○ Voice call

Country or region code

India (+91)

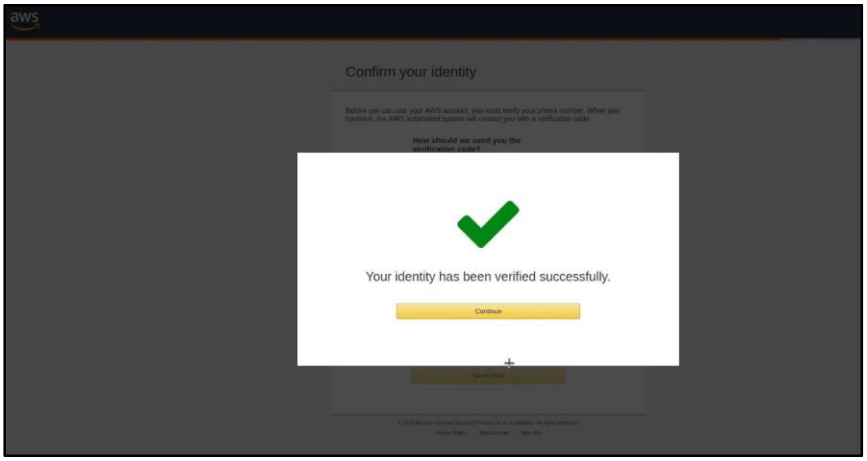
Cell Phone Number

Security check

Type the characters as shown above

© 2019 Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

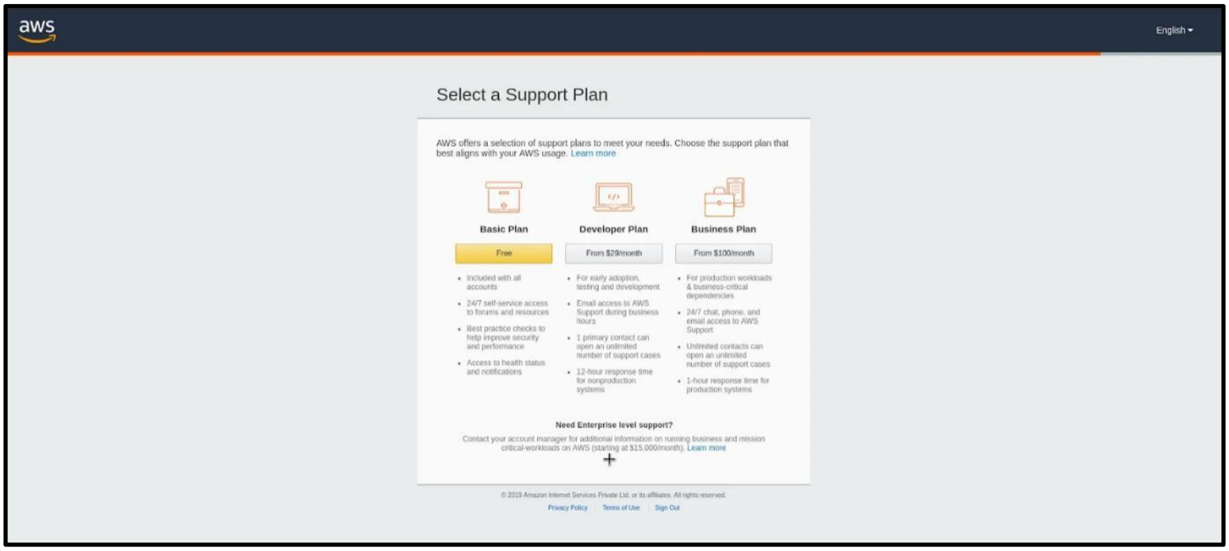
Once you have verified successfully, you should see a screen like below:



Confirm your identity

Your identity has been verified successfully.

Step 5 – Selecting a Support Plan



Select a Support Plan

AWS offers a selection of support plans to meet your needs. Choose the support plan that best aligns with your AWS usage. Learn more

Basic Plan

- Included with all accounts
- 24/7 self-service access to forums and resources
- Best practice checks to help improve security and performance
- Access to health status and notifications

Developer Plan

- For early adoption, testing and development
- Email access to AWS Support during business hours
- 1 primary contact can open an unlimited number of support cases
- 12-hour response time for nonproduction systems

Business Plan

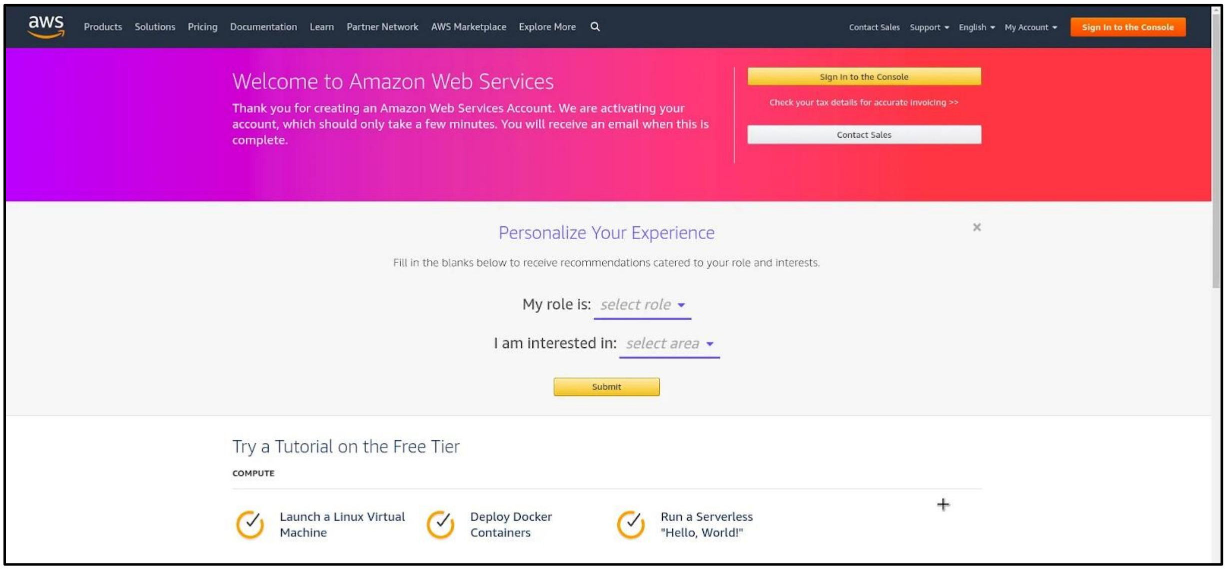
- For production workloads & business-critical dependencies
- 24/7 chat, phone, and email access to AWS Support
- Unlimited contacts can open an unlimited number of support cases
- 1-hour response time for production systems

Need Enterprise level support?

Contact your account manager for additional information on running business and mission critical-workloads on AWS (starting at $15,000/month). Learn more

© 2019 Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.

Go for **Basic Plan**. It is Free of cost and great for learning purposes. The other plans are **Developer Plan** and a **Business Plan**. But both of them are paid options.



Welcome to Amazon Web Services

Thank you for creating an Amazon Web Services Account. We are activating your account, which should only take a few minutes. You will receive an email when this is complete.

Personalize Your Experience

Fill in the blanks below to receive recommendations catered to your role and interests.

My role is: select role

I am interested in: select area

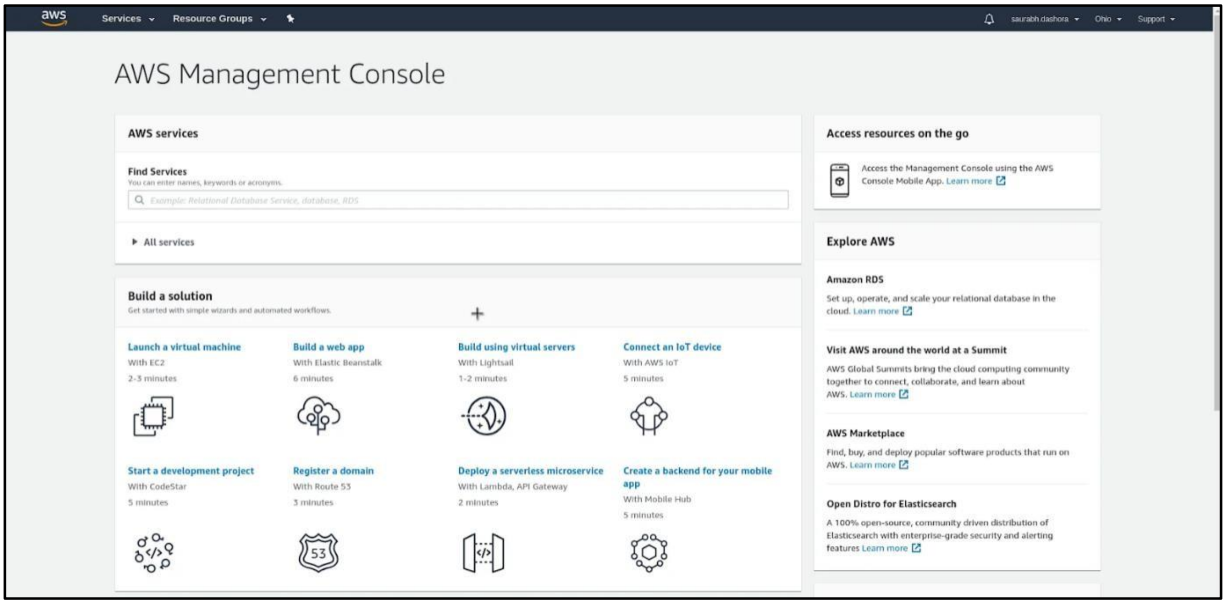
Try a Tutorial on the Free Tier

Launch a Linux Virtual Machine

Deploy Docker Containers

Run a Serverless "Hello, World!"

Finally, after logging in, you should be able to see the AWS Management Console as below:



AWS Management Console

AWS services

Find Services
You can enter names, keywords or acronyms.

Example: Relational Database Service, database, RDS

▸ All services

Build a solution
Get started with simple wizards and automated workflows.

Launch a virtual machine
With EC2
2-3 minutes

Build a web app
With Elastic Beanstalk
6 minutes

Build using virtual servers
With Lightsail
1-2 minutes

Connect an IoT device
With AWS IoT
5 minutes

Start a development project
With CodeStar
5 minutes

Register a domain
With Route 53
3 minutes

Deploy a serverless microservice
With Lambda, API Gateway
2 minutes

Create a backend for your mobile app
With Mobile Hub
5 minutes

Access resources on the go

Access the Management Console using the AWS Console Mobile App. Learn more

Explore AWS

Amazon RDS
Set up, operate, and scale your relational database in the cloud. Learn more

Visit AWS around the world at a Summit
AWS Global Summits bring the cloud computing community together to connect, collaborate, and learn about AWS. Learn more

AWS Marketplace
Find, buy, and deploy popular software products that run on AWS. Learn more

Open Distro for Elasticsearch
A 100% open-source, community driven distribution of Elasticsearch with enterprise-grade security and alerting features Learn more

If you have reached this far, you have successfully finished **Creating an AWS Account**.

Deployment Steps:

Step 1: Launch a Windows Server Amazon EC2 instance.

Step 2: Configure your source content to deploy to the Windows Server Amazon EC2 instance.Step 3: Upload your "hello, world!”...

Step 4: Deploy your Hello World application.

Step 5: Update and redeploy your "hello,
world!" ...

Step 6: Clean up your "hello, world!"

Conclusion: Web application deployment is done successfully.