

COURSE OBJECTIVES:

1. To understand basic concepts of web programming and scripting languages.
2. To learn Version Control Environment.
3. To learn frontend technologies and back-end technologies.
4. To understand mobile web development.
5. To comprehend web application deployment.

COURSE OUTCOMES:

On completion of the course, students will be able to –

CO1: Develop Static and Dynamic responsive website using technologies HTML, CSS, Bootstrap and AJAX.

CO2: Create Version Control Environment.

CO3: Develop an application using frontend and backend technologies.

CO4: Develop mobile website using JQuery Mobile.

CO5: Deploy web application on cloud using AWS.

Assignment 1:

a. 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. HTML

- 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 contents such as "this is a heading", "this is a paragraph", "this is a link", etc

A simple HTML Document

```
<!DOCTYPEhtml>
<html>
<head>
<title>PageTitle</title>
</head>
<body>

<h1>MyFirst Heading</h1>
<p>Myfirstparagraph.</p>

</body>
</html>
```

- The `<!DOCTYPEhtml>` 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

II. CSS

- 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

```
<!DOCTYPEhtml>
<html>
<head>
<style>
body {
  background-color:lightblue;
}

h1 {
  color:white;
  text-align: center;
}

p {
  font-family:verdana;
  font-size: 20px;
}
</style>
</head>
<body>

<h1>MyFirstCSSExample</h1>
<p>Thisisaparagraph.</p>

</body>
</html>
```

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
- Simple 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/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</head>
<body>

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

</body>
</html>
```

Bootstrap CDN

```
<!--LatestcompiledandminifiedCSS-->
<linkrel="stylesheet"href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstra
p.min.css">

<!--jQuerylibrary-->
<scriptsrc="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<!--LatestcompiledJavaScript -->
<scriptsrc="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script
>
```

b. 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.

AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and JavaScript.

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 server using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.

With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, 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 data, although any format, including plain text, can be used. AJAX is a web browser technology independent 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.

- JavaScript
 - Loosely typed scripting language.
 - JavaScript function is called when an event occurs in a page.
 - Glue for the whole AJAX operation.
- DOM
 - API for accessing and manipulating structured documents.
 - Represents the structure of XML and HTML documents.
- CSS
 - Allows for a clear separation of the presentation style from the content and may be changed programmatically by JavaScript
- XMLHttpRequest
 - JavaScript object that performs asynchronous interaction with the server.

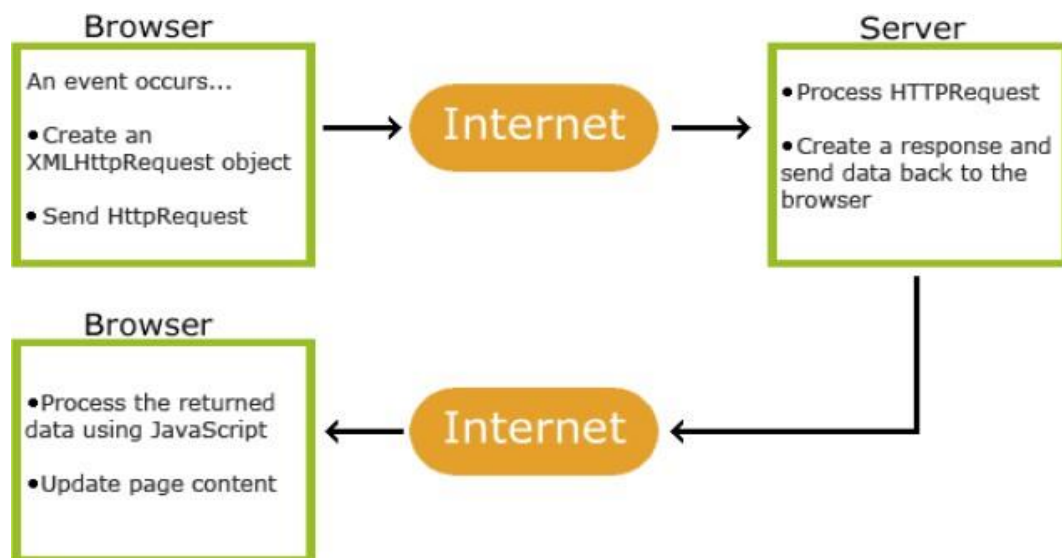


Fig1. How AJAX works

AJAX–Events: onreadystatechangeEvent Properties

Property	Description
onReadyStateChange	It is called whenever readyState attribute changes. It must not be used with synchronous requests.

readyState	<p>represents the state of the request. It ranges from 0 to 4.</p> <ul style="list-style-type: none"> 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	<p>200: "OK" 403: "Forbidden" 404: "Page not found"</p>

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 object's 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

<u>Method</u>	<u>Description</u>
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 should 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 Fri
      ends<br>
      WelcometoPune!!!!<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!!!!!!
```

Conclusion:

Assignment 2:

- a. Create version control account on GitHub and using Git commands to create repository and push your code to GitHub.
- b. Create Docker Container Environment (NVIDIA Docker or any other).
- c. Create an Angular application which will do following actions: Register User, Login User, Show User Data on Profile Component

Theory:

Part a.

1. What is Git?

Git is a popular version control system. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.

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

Subject: LP-II (Web Application Development)

- 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 made in each commit!

4. Why Git?

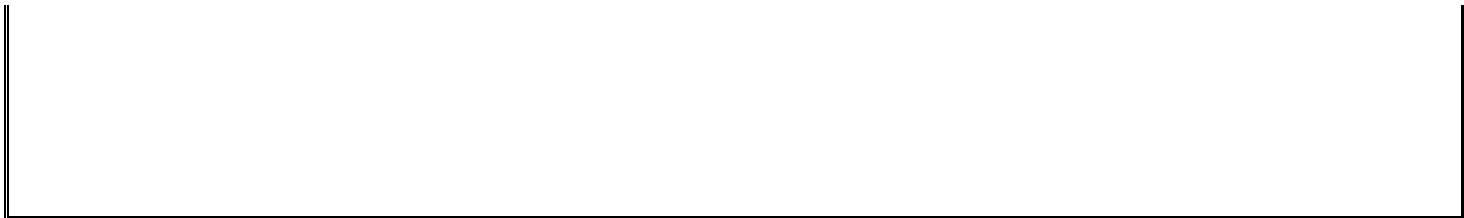
- Over 70% of developers use Git!
- Developers can work together from anywhere in the world.
- Developers can see the full history of the project.
- Developers can revert to earlier versions of a project.

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 since 2018.

6. Steps 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	Commandline > git --version	If Git is installed, it should show something like git version X.Y
3	git config --global user.name "w3schools-test" git config --global user.email "test@w3schools.com"	Configure Git Change the username and email address to your own
4	mkdir myproject cd myproject	Creating Git Folder
5	git init	Initialize Git



		Initialized empty Git repository in /Users/user/myproject/.git/
6	<code>git status</code>	To check the status
7	<code>git add index.html</code>	Add file to staging environment
8	<code>git add --all</code>	add all files in the current directory to the Staging Environment:
9	<code>git commit -m "First release of Hello World!"</code>	The commit command performs a commit, and the -m "message" adds a message.
10	<code>git commit -a -m "Updated index.html with a new line"</code>	Skips staging environment
11	<code>git log</code>	To view the history of commits for a repository, you can use the log command
12	<code>git command -help</code>	See all the available options for the specific command
13	<code>git help --all</code>	See all possible commands
14	<code>git commit -help</code>	See help for specific command
15	<code>git branch hello-world-images</code>	a branch is a new/separate version of the main repository. This command creates a new branch hello-world-images
16	<code>git checkout hello-world-images</code>	checkout is the command used to checkout/move to a branch
17	<code>git checkout master</code>	Used to switch between branches
18	https://github.com/	Create a new account on github
19		Create a Repository on GitHub
20	<code>git remote add origin https://github.com/w3schools-test/hello-world.git</code>	Push Local Repository to GitHub
21	<code>git push --set-upstream origin master</code>	push master branch to the origin url,

22		go back into GitHub and see that the repository has been updated:
23	<code>git fetch origin</code>	fetch gets all the change history of a tracked branch/repo
24	<code>git merge origin/master</code>	merge combines the current branch, with a specified branch.
25	<code>git pull origin</code>	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.

Partc:

Useful link - <https://www.tutorialsteacher.com/angular/install-angular>

1. Angular requires a current, active LTS (long term support) or maintenance LTS version of Node.js and NPM.

install node.js <https://nodejs.org/>

It will automatically install NPM - node package manager

2. Install Angular CLI

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

To Create Angular 2 Application Angular CLI is required

3. To create new project

















through CLI go to folder of the new project Give

command as -

`ng new project-name` press

ENTER

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

Open foldersrc/app

Modifyapp.module.tsforformapplication

```
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]
})
export class AppModule { }
```

Open app.component.html

Writehtmlcodeforform(representative codeismentioned here,modifyformultipleinputs)

```
<h1>Simple Form</h1>
<form #simpleForm="ngForm"(ngSubmit)="getValues(simpleForm.value)">
  <input type="text" ngModel name="user" placeholder="Enter Name">
  <br><br>
  <input type="text" ngModel name="age" placeholder="Enter age">
  <br><br>
  <input type="text" ngModel name="city" placeholder="Enter city">
  <br><br>
  <button>Get user value</button>
</form>
```

Make changes in app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'AngProj1';
  getValues(val: any) {
    console.log(val);
  }
}
```

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

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.

Assignment 3:

Create a Node.js Application which serves a static website.

- a. Create four APIs using Node.js, Express.js and MongoDB for CRUD Operations on assignment 2.C.

Theory:

Part a:

Create a Node.js Application which serves a static website.

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

Theory :

Node.js overview

In basic terms, Node.js is an open-source cross-platform library for server-side programming that permits clients to develop web applications rapidly. With Node.js, 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.js 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:** Node.js is 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.js.
- **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 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. In asynchronous programming, the codes become clumsier, and developers have to depend on the nes

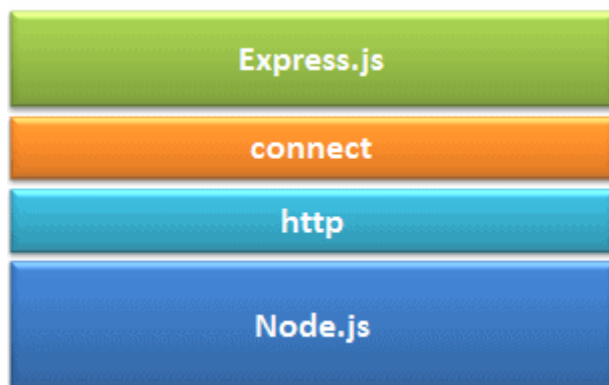
Express.js

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

[Expressjs.com](https://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:

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 webpage
7. Running your project

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

Setting up express.js

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": [],
  "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—clientside

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>
     // 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

Assignment 4:

a. Create a simple Mobile Website using jQuery Mobile.
b. Deploy/Host Your web application on AWS VPC or AWS Elastic Beanstalk. Mini Project
Develop a web application using full stack development technologies in any of the following domains:

1. Social Media
2. e-commerce
3. Restaurant
4. Medical
5. Finance
6. Education
7. Any other

Theory:

Part a:

jQuery Mobile

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 will work the same way on the mobile, tablet, and desktop devices.
- 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 "writeless, domore" 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.



Click the *Stable* button, which leads directly to a ZIP file containing the CSS and jQuery files, for the latest version of jQuery mobile library. Extract the ZIP file contents to a jQuery mobile directory.

This version contains all files including all dependencies, a large collection of demos, and even the library's unit test suite. This version is helpful to getting started.

Conclusion:

Part b:

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 billion revenue target
- 2016-Releases 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:

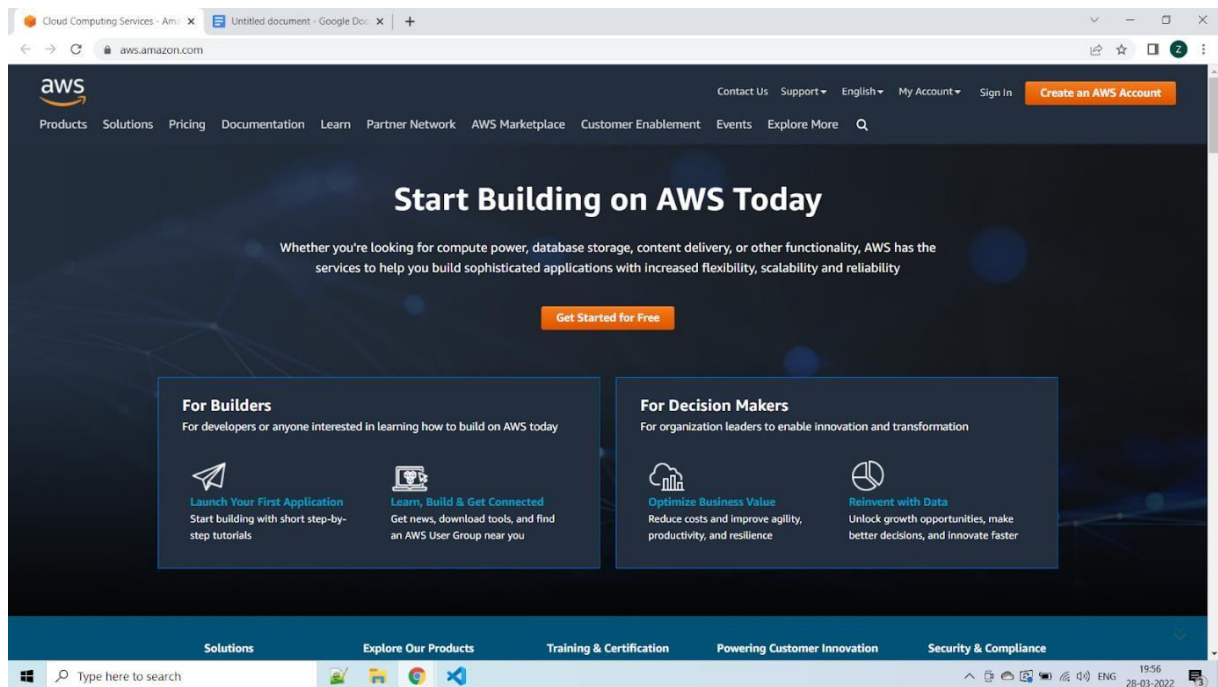
- Website hosting
- Application hosting/SaaS hosting
- Media Sharing (Image/ Video)
- Mobile and Social Applications
- Content delivery and Media Distribution
- Storage, backup, and disaster recovery
- Development and test environments
- Academic Computing
- Search Engines
- Social Networking

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

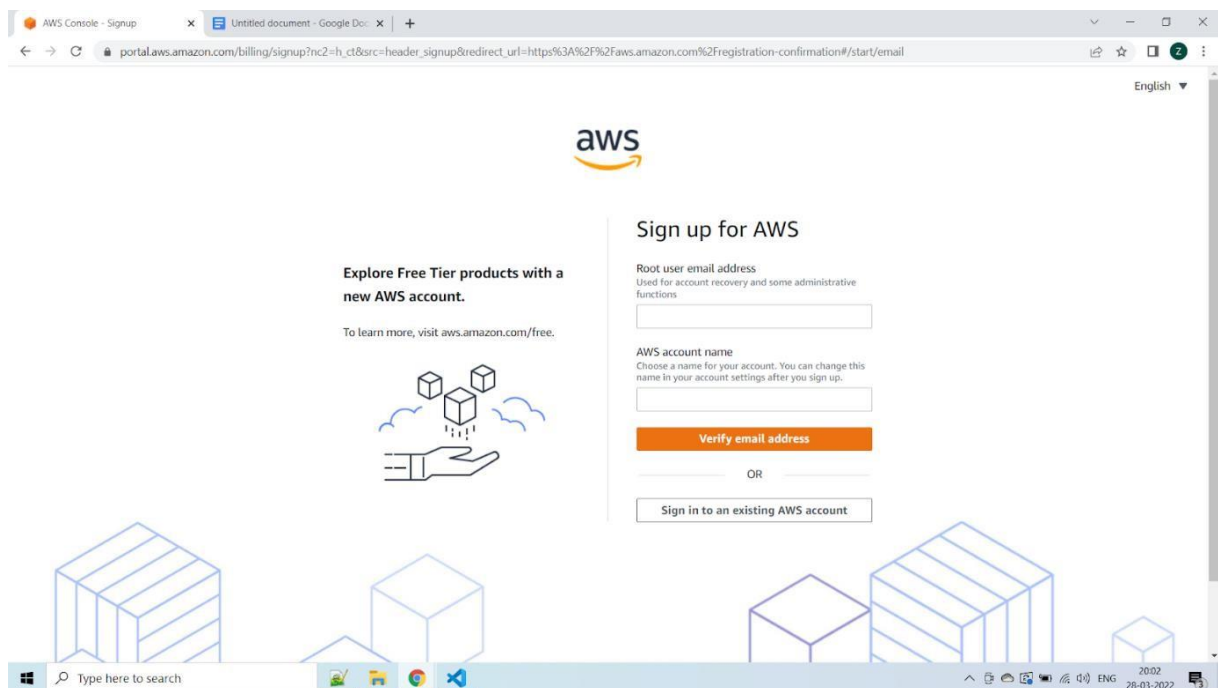
Steps to follow are as follows :

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

You should see something like below:

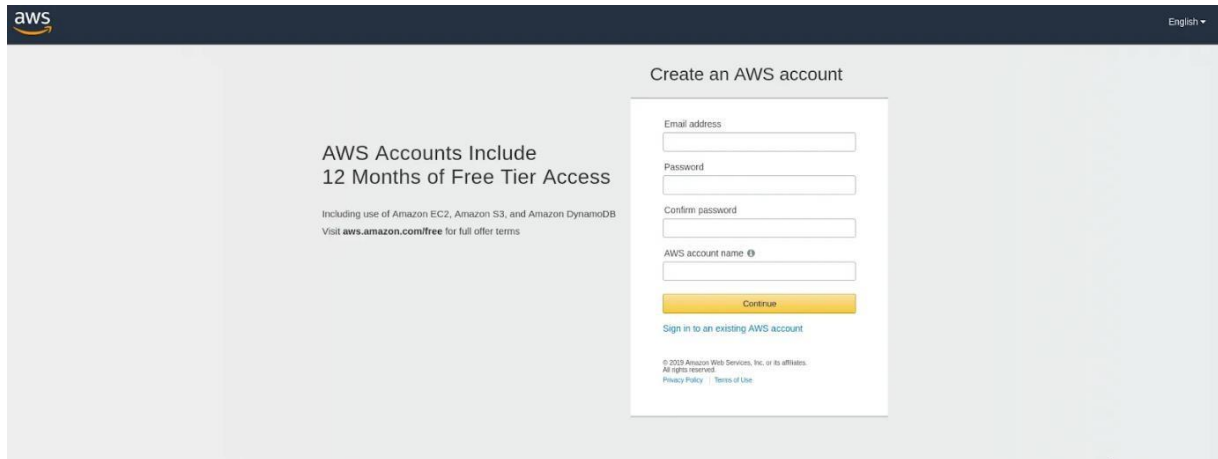


In order to continue, click the **Complete Sign Up** button in the middle of the screen or on the top right corner of the screen. You will see the below screen.



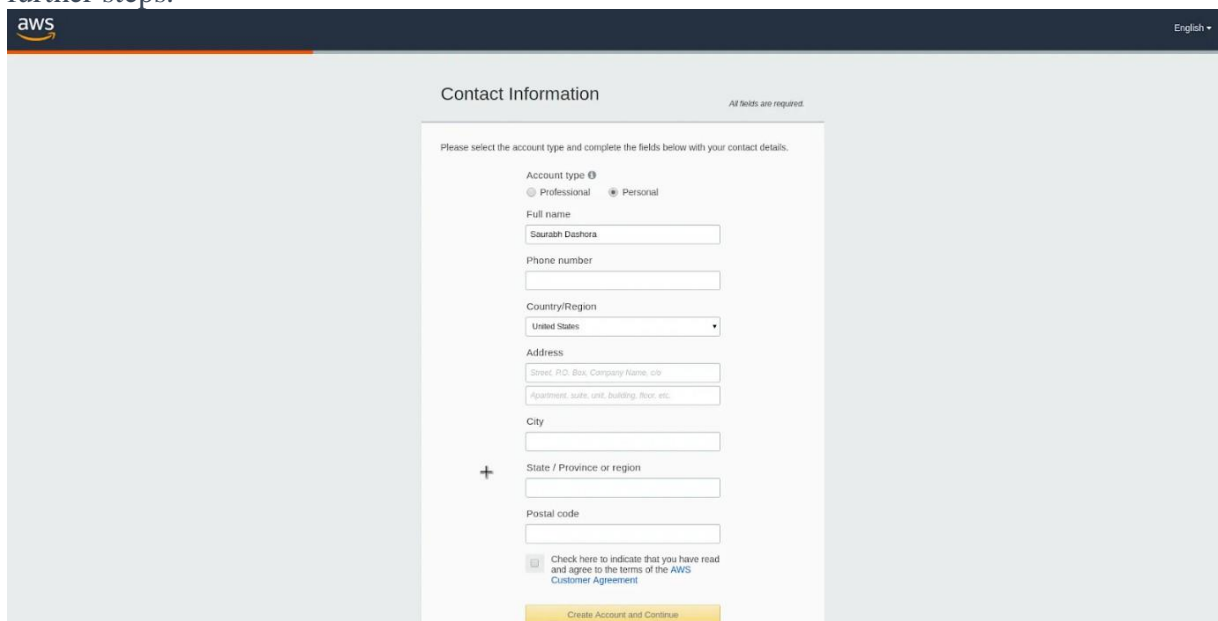
Step2– Entering User Details

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



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

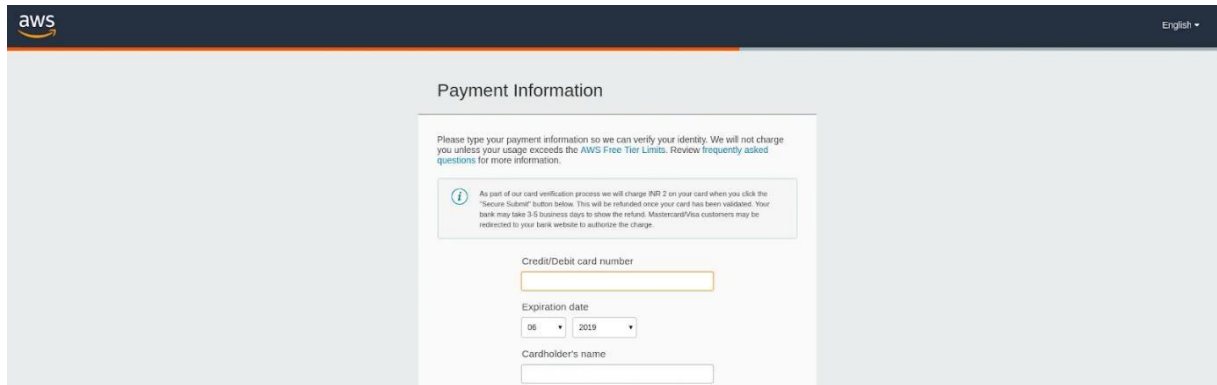
Next you will be asked to fill up your contact details such as contact number, country, address and so on. You should fill them up properly because your contact number is important for further steps.



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.

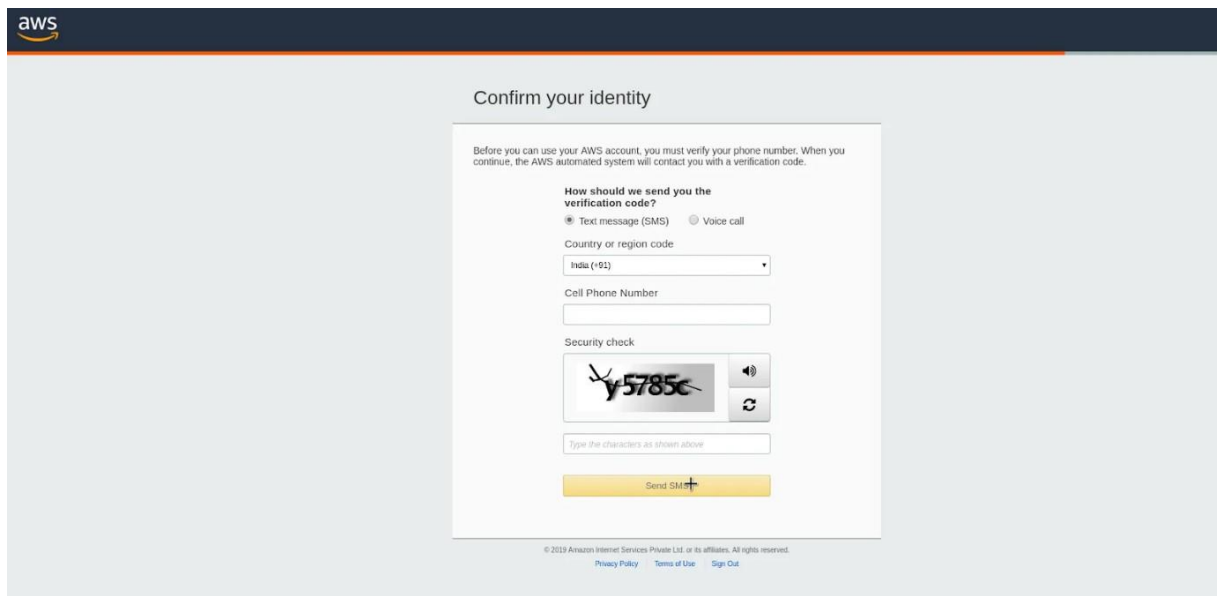


The screenshot shows the AWS 'Payment Information' form. At the top, there's an AWS logo and a language dropdown set to 'English'. The main heading is 'Payment Information'. Below it, a message states: 'Please type your payment information so we can verify your identity. We will not charge you unless your usage exceeds the AWS Free Tier Limits. Review frequently asked questions for more information.' An information icon (i) is next to a note: '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.' The form fields include: 'Credit/Debit card number' (text input), 'Expiration date' (two dropdown menus showing '06' and '2019'), and 'Cardholder's name' (text input).

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.

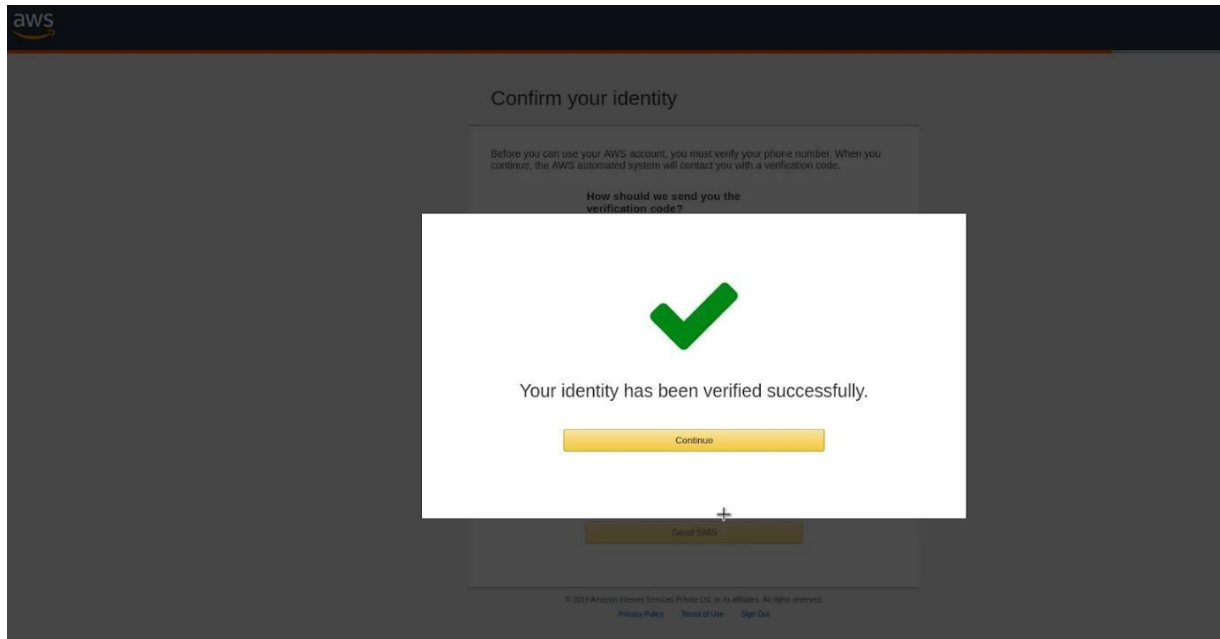
Step 4 – Identity Confirmation

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

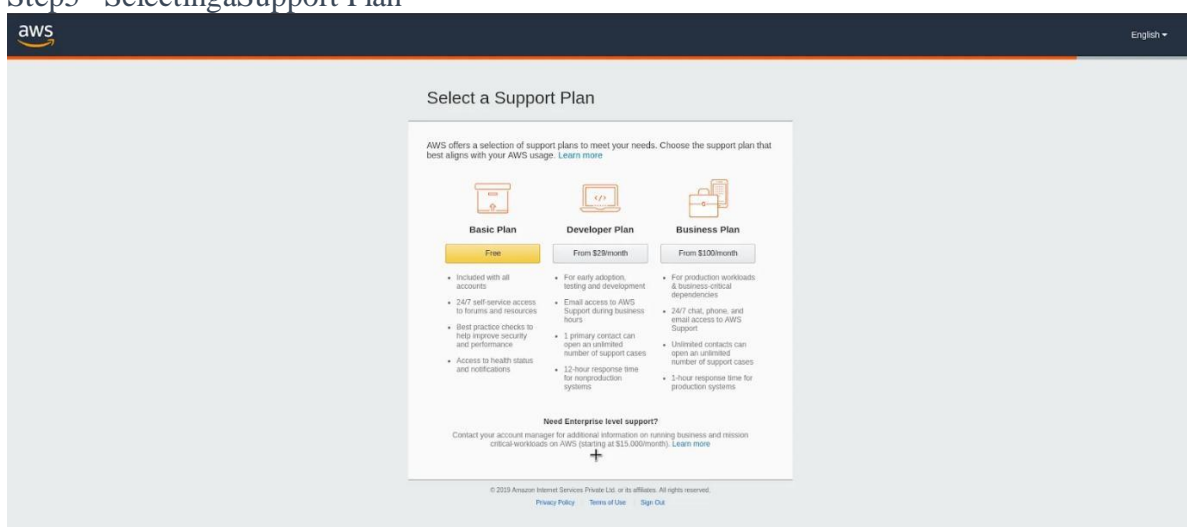


The screenshot shows the AWS 'Confirm your identity' form. At the top, there's an AWS logo. The main heading is 'Confirm your identity'. Below it, a message states: 'Before you can use your AWS account, you must verify your phone number. When you continue, the AWS automated system will contact you with a verification code.' The form asks 'How should we send you the verification code?' with two radio buttons: 'Text message (SMS)' (selected) and 'Voice call'. Below this is a 'Country or region code' dropdown menu showing 'India (+91)'. Then is a 'Cell Phone Number' text input field. A 'Security check' section shows a CAPTCHA image with the characters 'y5785c' and a text input field with the placeholder 'Type the characters as shown above'. At the bottom is a yellow 'Send SMS' button with a plus icon. The footer contains copyright information: '© 2019 Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.' and links for 'Privacy Policy', 'Terms of Use', and 'Sign Out'.

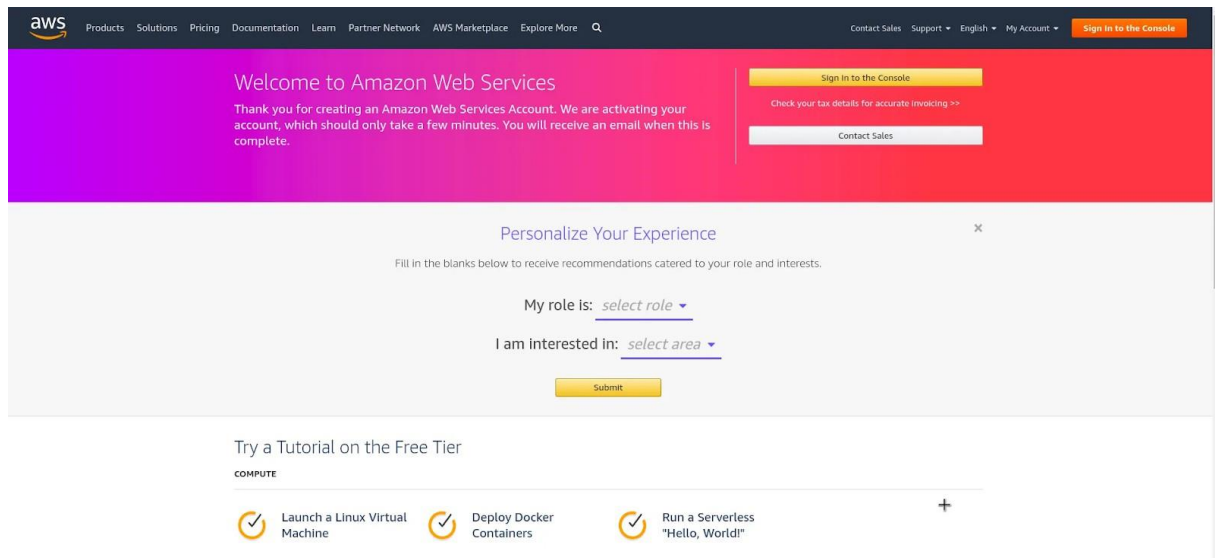
Once you have verified successfully, you should see a screen like below:



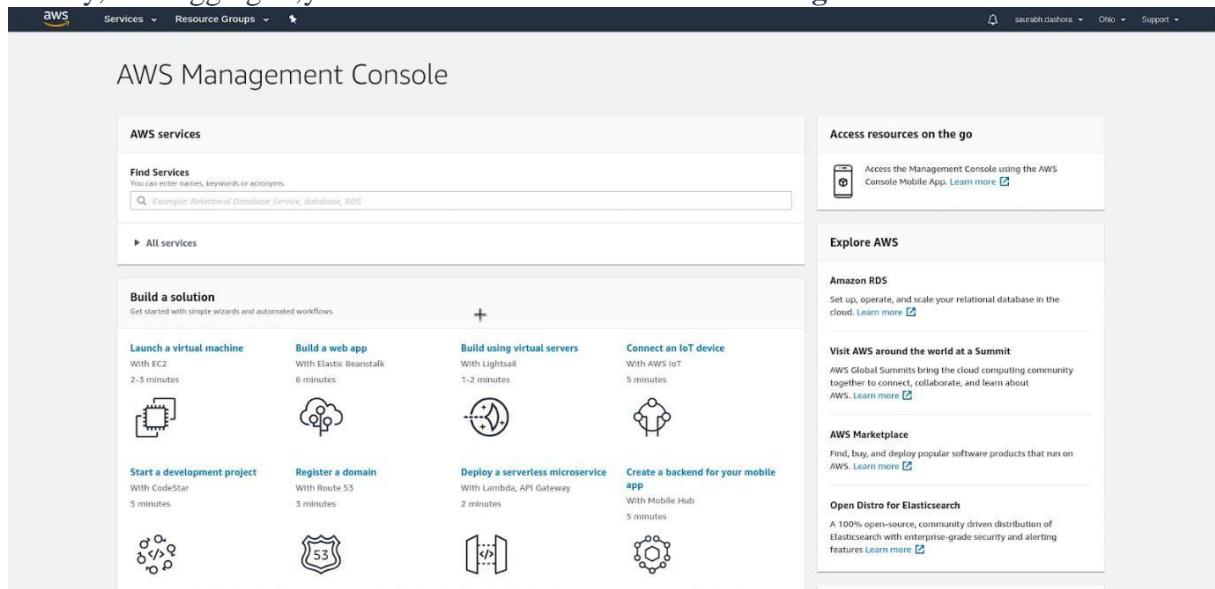
Step5– Selecting a Support Plan



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. Once you select your plan, you will see the below **Welcome** screen. From here on, you can sign in to your **AWS Console**.



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



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 HelloWorld application.

Step 5: Update and redeploy your "hello, world!" ... Step 6:

Clean up your "hello, world!"

Conclusion: