Deccan Education Society's

Navinchandra Mehta Institute of Technology and Development

C E R T I F I C A T E

This is to certify that Mr. / ~~Miss~~. Anonymous King of M.C.A.

Semester I with Roll No. C99999 has completed 11 practicals of Web Technologies under my supervision in this college during the year 2022-2023.

CO	**R1 (Journal)**	**R2 (Performance during lab session)**	**R3 (Implementation using different problem solving techniques)**	**R4 (Mock Viva)**	**Attendance**
CO1					
CO2					
CO3					
CO4					

Practical-in-charge

Head of Department

MCA Department

(NMITD)

MCAL14 Web Technology Lab
Index

2	Create an application to demonstrate various Node.js Events	**28/11/2022**	**CO1**	
3	Create an application to demonstrate Node.js Functions	**12/12/2022**	**CO1**	
4	Using File Handling demonstrate all basic file operations (Create, write, read, delete)	**15/12/2022**	**CO1**	
5	Create an HTTP Server and perform operations on it	**02/01/2023**	**CO1**	
6	Create an application to establish a connection with the MySQL database and perform basic database operations on it	**05/01/2023**	**CO2**	
7	Create an application using Filters	**09/01/2023**	**CO3**	
8	Create an application to demonstrate directives	**13/01/2023 & 19/01/2023**	**CO3**	
9	Demonstrate controllers in Angular.js through an application	**20/01/2023**	**CO3**	
10	Demonstrate features of Angular.js forms with a program	**30/01/2023**	**CO3 CO4**	
11	Create a SPA (Single Page Application)	**10/02/2023**	**CO4**	

Introduction to Node.js:

What is Node.js

o Node.js is an open source server environment o Node.js is free
o Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.) o Node.js uses JavaScript on the server

Node.js uses asynchronous programming!

Setup Development Environment: Installation of Node.js on Windows:

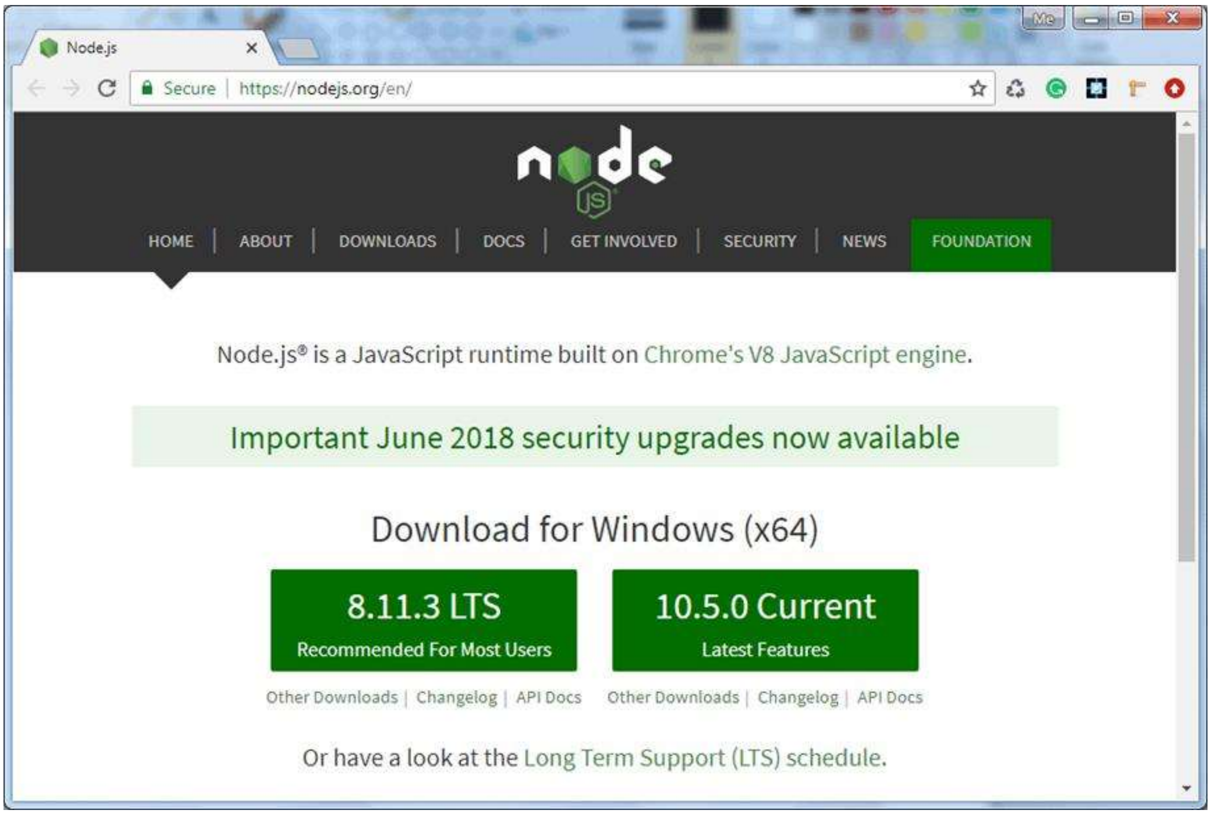
Node.js development environment can be setup in Windows, Mac, Linux and Solaris. The following tools/SDK are required for developing a Node.js application on any platform.

1. Node.js
2. Node Package Manager (NPM)
3. IDE (Integrated Development Environment) or TextEditor

NPM (Node Package Manager) is included in Node.js installation since Node version 0.6.0., so there is no need to install it separately.

Install Node.js on Windows

Visit Node.js official web site https://nodejs.org. It will automatically detect OS and display download link as per your Operating System. For example, it will display following download link for 64 bit Windows OS.


Node.js
Secure | https://nodejs.org/en/
HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | NEWS | FOUNDATION

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Important June 2018 security upgrades now available

Download for Windows (x64)

8.11.3 LTS
Recommended For Most Users

10.5.0 Current
Latest Features

Other Downloads | Changelog | API Docs

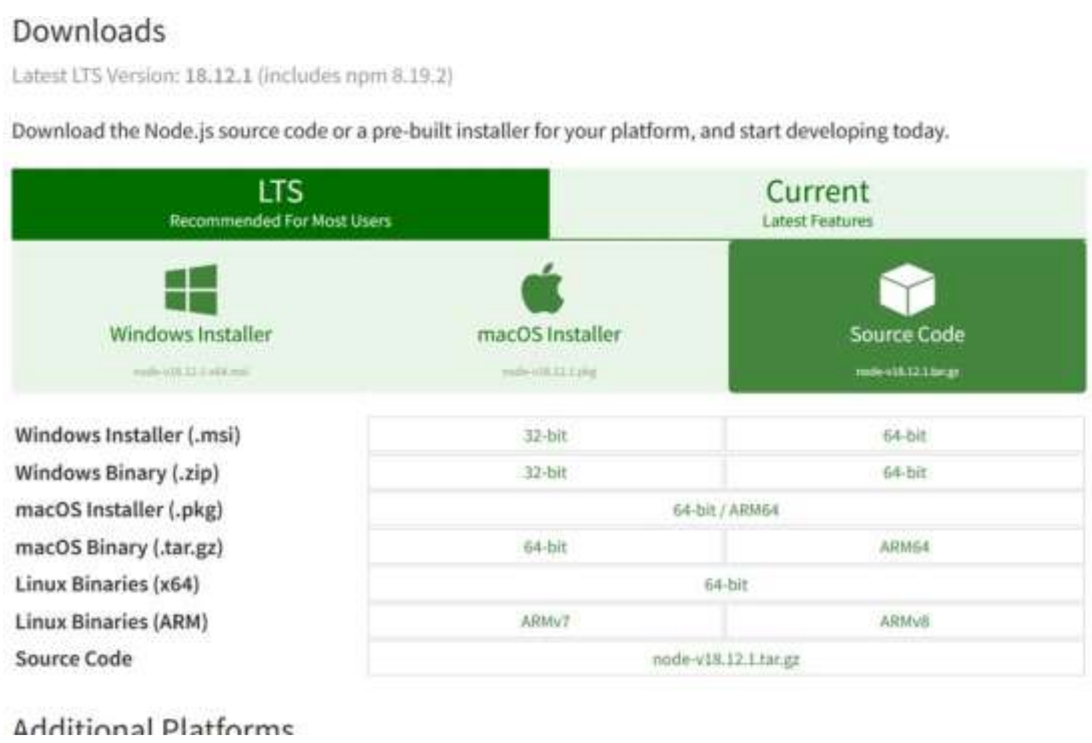
Other Downloads | Changelog | API Docs

Or have a look at the Long Term Support (LTS) schedule.

Download Node.JS Installer for Windows

Download node MSI for windows by clicking on 8.11.3 LTS or 10.5.0 Current button.

https://nodejs.org/en/download/



Downloads

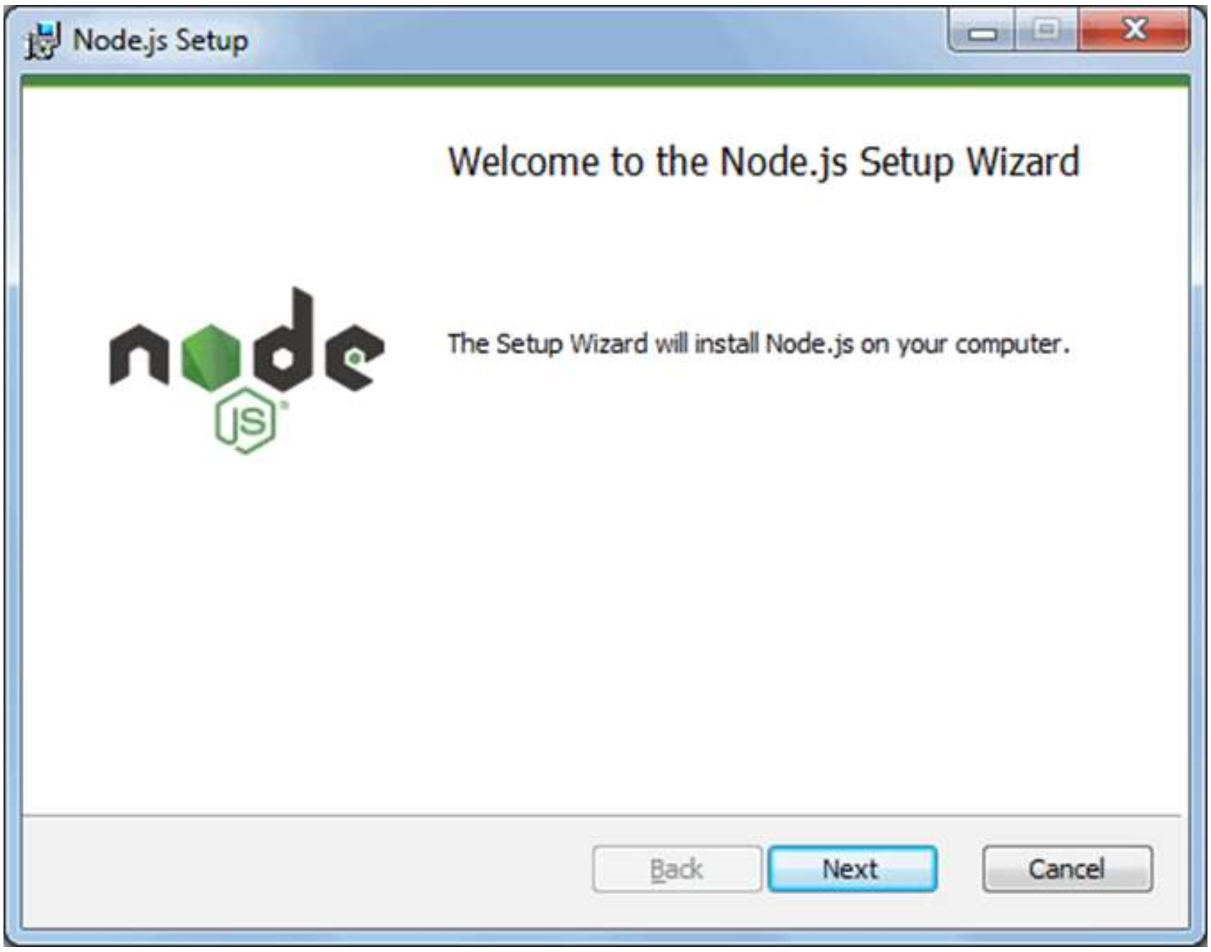
Latest LTS Version: 18.12.1 (includes npm 8.19.2)

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

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit / ARM64	
macOS Binary (.tar.gz)	64-bit	ARM64
Linux Binaries (x64)	64-bit	
Linux Binaries (ARM)	ARMv7	ARMv8
Source Code	node-v18.12.1.tar.gz	

Additional Platforms

After you download the MSI, double-click on it to start the installation as shown below.


Node.js Setup

Welcome to the Node.js Setup Wizard

The Setup Wizard will install Node.js on your computer.

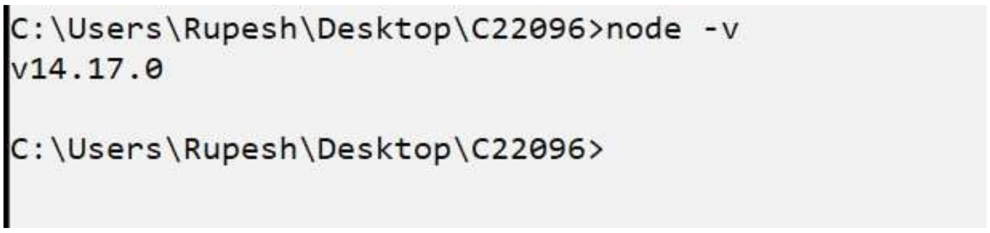
Node.js Installation

Click Next to read and accept the License Agreement and then click Install. It will install Node.js quickly on your computer. Finally, click finish to complete the installation.

Verify Installation

Once you install Node.js on your computer, you can verify it by opening the command prompt and typing node -v. If Node.js is installed successfully then it will display the version of the Node.js installed on your machine, as shown below.

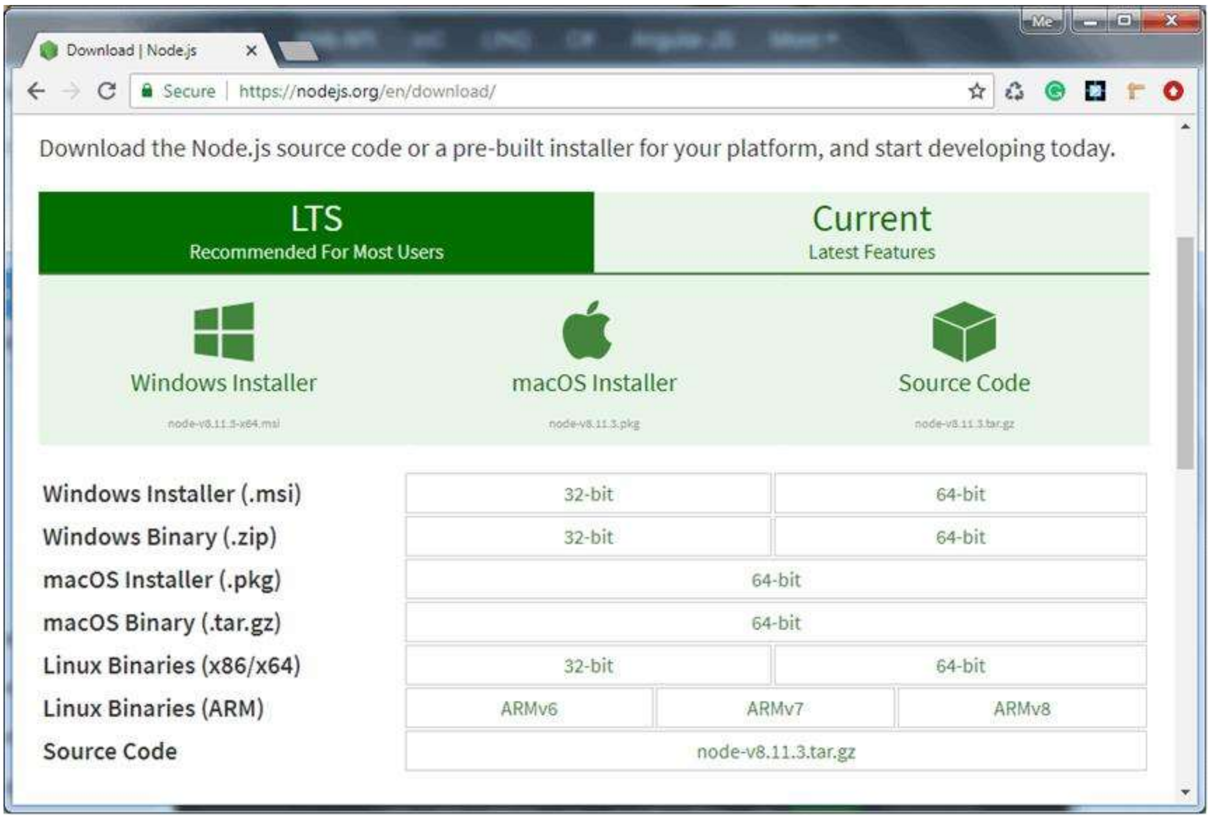
Verify Node.js Installation



```
C:\Users\Rupesh\Desktop\C22096>node -v
v14.17.0

C:\Users\Rupesh\Desktop\C22096>
```

Install Node.js on Mac/Linux

Visit Node.js official web site https://nodejs.org/en/download page. Click on the appropriate installer for Mac (.pkg or .tar.gz) or Linux to download the Node.js installer.



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

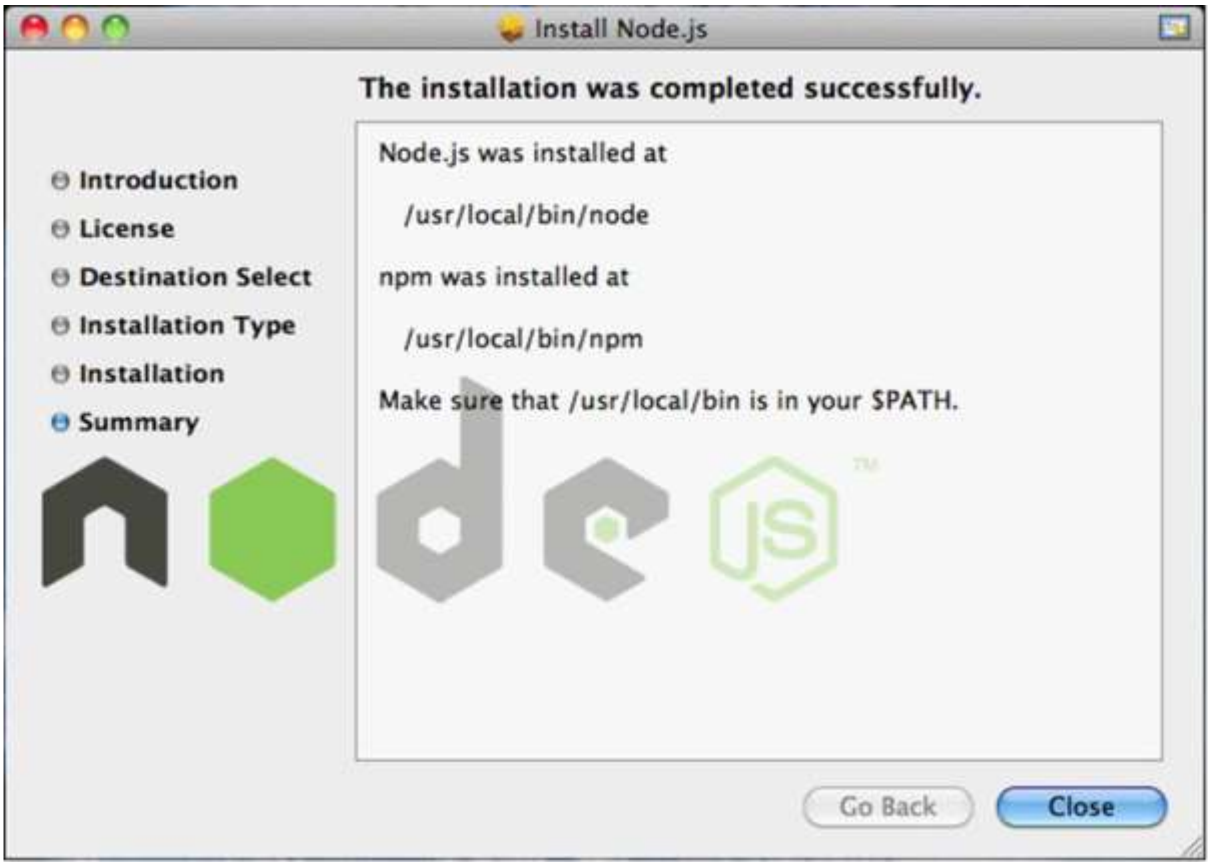
LTS Recommended For Most Users	Current Latest Features

Windows Installer	macOS Installer	Source Code
node-v8.11.3-x64.msi	node-v8.11.3.pkg	node-v8.11.3.tar.gz

Windows Installer (.msi)	32-bit	64-bit	
Windows Binary (.zip)	32-bit	64-bit	
macOS Installer (.pkg)	64-bit		
macOS Binary (.tar.gz)	64-bit		
Linux Binaries (x86/x64)	32-bit	64-bit	
Linux Binaries (ARM)	ARMv6	ARMv7	ARMv8
Source Code	node-v8.11.3.tar.gz		

Environment Setup

Once downloaded, click on the installer to start the Node.js installation wizard. Click on Continue and follow the steps. After successful installation, it will display summary of installation about the location where it installed Node.js and NPM.



The installation was completed successfully.

Node.js was installed at

/usr/local/bin/node

npm was installed at

/usr/local/bin/npm

Make sure that /usr/local/bin is in your $PATH.

Node.js Installation on OS X

After installation, verify the Node.js installation using terminal window and enter the following command. It will display the version number of Node.js installed on your Mac.

```
$ node -v
```

Optionally, for Mac or Linux users, you can directly install Node.js from the command line using Homebrew package manager for Mac OS or Linuxbrew package manager for Linux Operating System. For Linux, you will need to install additional dependencies, viz. Ruby version 1.8.6 or higher and GCC version 4.2 or higher before installing node. $ brew install node

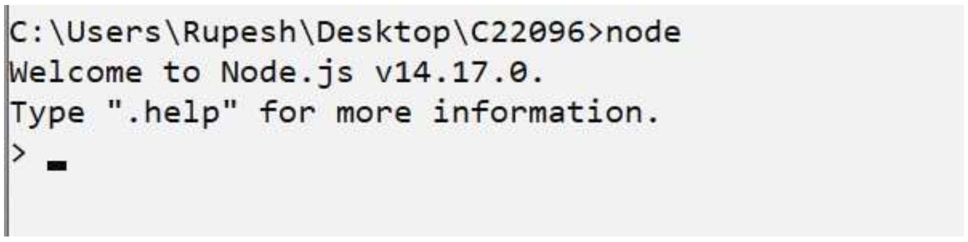
IDE

Node.js application uses JavaScript to develop an application. So, you can use any IDE or texteditor tool that supports JavaScript syntax. However, an IDE that supports auto complete features for Node.js API is recommended e.g. Visual Studio, Sublime text, Eclipse, Aptana etc.

Working in REPL

Node.js comes with virtual environment called REPL (aka Node shell). REPL stands for **ReadEval-Print-Loop**. It is a quick and easy way to test simple Node.js/JavaScript code.

To launch the REPL (Node shell), open command prompt (in Windows) or terminal (in Mac or UNIX/Linux) and type *node* as shown below. It will change the prompt to > in Windows and MAC.



```
C:\Users\Rupesh\Desktop\C22096>node
Welcome to Node.js v14.17.0.
Type ".help" for more information.
> ▬
```

Launch Node.js REPL

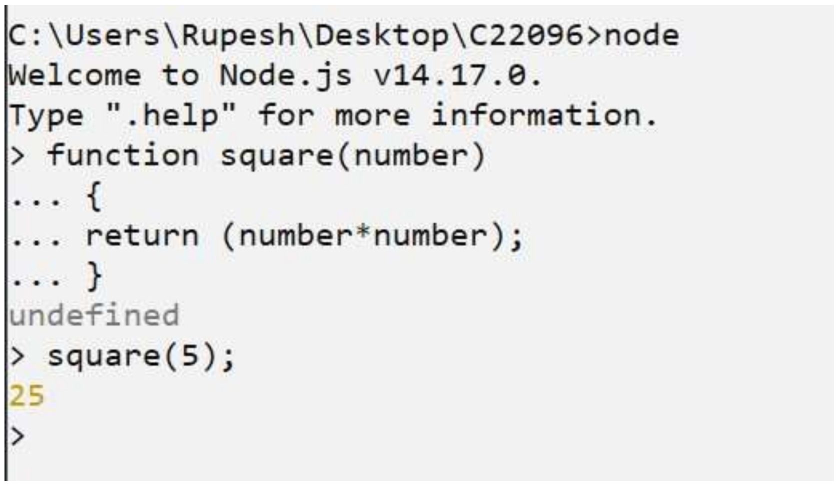
You can now test pretty much any Node.js/JavaScript expression in REPL. For example, if your write "10 + 20" then it will display result 30 immediately in new line.

The + operator also concatenates strings as in browser's JavaScript.

You can also define variables and perform some operation on them.

If you need to write multi line JavaScript expression or function then just press **Enter** whenever you want to write something in the next line as a continuation of your code. The REPL terminal will display three dots (...), it means you can continue on next line. Write .break to get out of continuity mode.

For example, you can define a function and execute it as shown below.



```
C:\Users\Rupesh\Desktop\C22096>node
Welcome to Node.js v14.17.0.
Type ".help" for more information.
> function square(number)
... {
... return (number*number);
... }
undefined
> square(5);
25
>
```

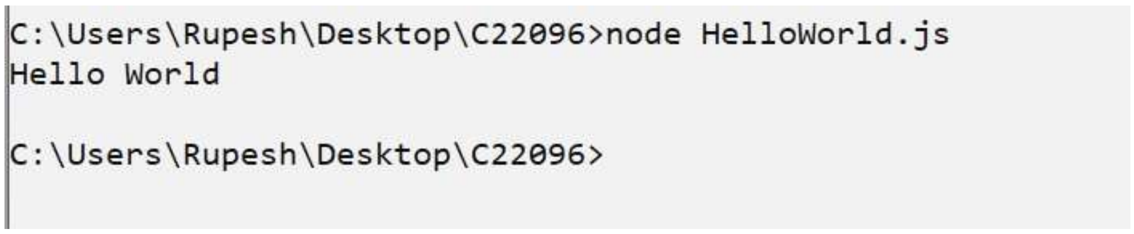
Node.js Example in REPL

You can execute an external JavaScript file by writing node fileName command. For example, assume that node-example.js is on C drive of your PC with following code.

node-example.js Copy

console.log("Hello World");

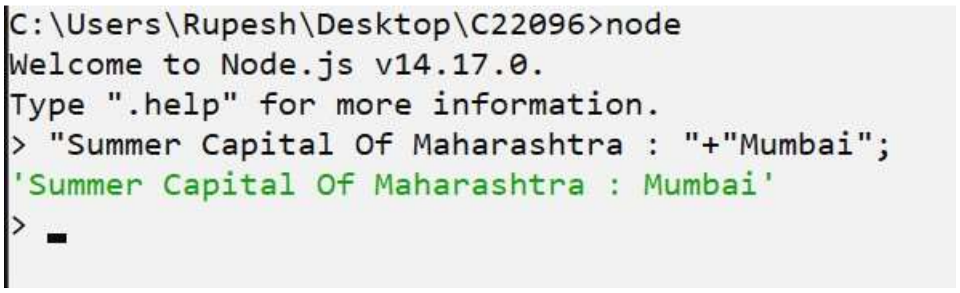
Now, you can execute node-exampel.js from command prompt as shown below.



```
C:\Users\Rupesh\Desktop\C22096>node HelloWorld.js
Hello World

C:\Users\Rupesh\Desktop\C22096>
```

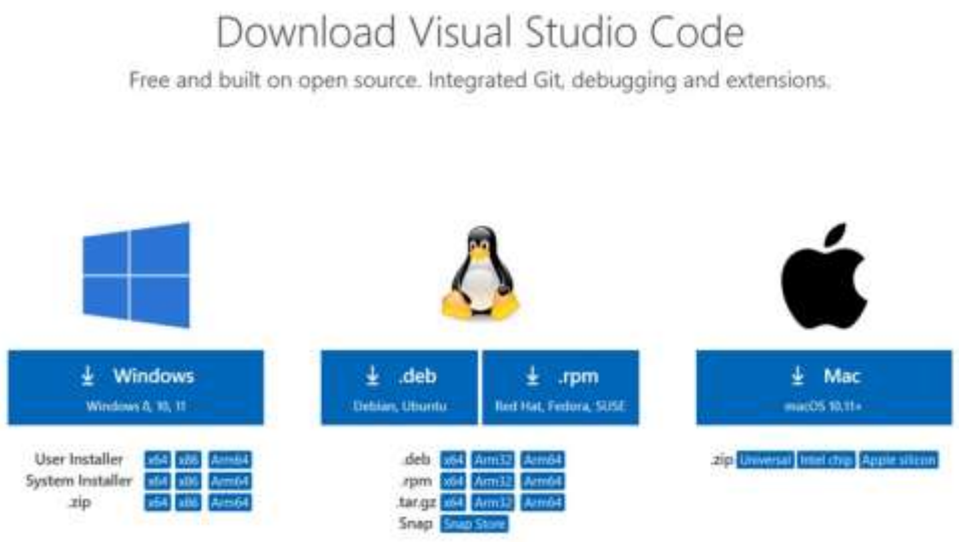
Run External JavaScript file

To exit from the REPL terminal, press Ctrl + C twice or write .exit and press Enter.



```
C:\Users\Rupesh\Desktop\C22096>node
Welcome to Node.js v14.17.0.
Type ".help" for more information.
> "Summer Capital Of Maharashtra : "+"Mumbai";
'Summer Capital Of Maharashtra : Mumbai'
>
```

https://code.visualstudio.com/download


Download Visual Studio Code
Free and built on open source. Integrated Git, debugging and extensions.
Windows
Windows 8, 10, 11
.deb
Debian, Ubuntu
.rpm
Red Hat, Fedora, SUSE
Mac
macOS 10.11+
User Installer x64 x86 Arm64
System Installer x64 x86 Arm64
.zip x64 x86 Arm64
.deb x64 Arm32 Arm64
.rpm x64 Arm32 Arm64
.tar.gz x64 Arm32 Arm64
Snap Snap Store
.zip Universal Intel chip Apple silicon

PRACTICAL NO.1

Aim: Create an application to demonstrate Node.js Modules

In Node.js, Modules are the blocks of encapsulated code that communicates with an external application on the basis of their related functionality. Modules can be a single file or a collection of a multiple files/folders.
To include a module, use the **require()** function with the name of the module:
var http=require('http');
Now our application has the access to the HTTP module, and is able to create a server.
Frequently used modules:

1. http
2. fs
3. express
4. event
5. net

We can create our own modules such as:

Code:-

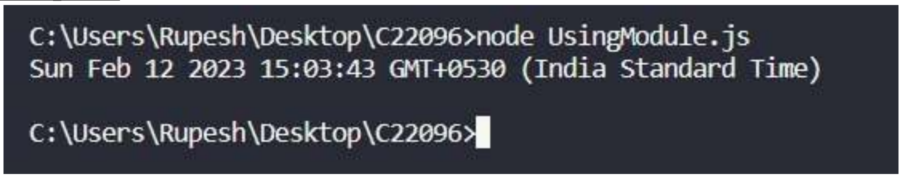
Using Module file:

```
var dt=require('./myfirstmodule');

console.log(dt.myDateFun());
```

myFirstModule file:

```
exports.myDateFun=function(){

   return Date();

}
```

Output:-



```
C:\Users\Rupesh\Desktop\C22096>node UsingModule.js
Sun Feb 12 2023 15:03:43 GMT+0530 (India Standard Time)

C:\Users\Rupesh\Desktop\C22096>
```

PRACTICAL NO.2

Aim: Create an application to demonstrate Node.js Modules .

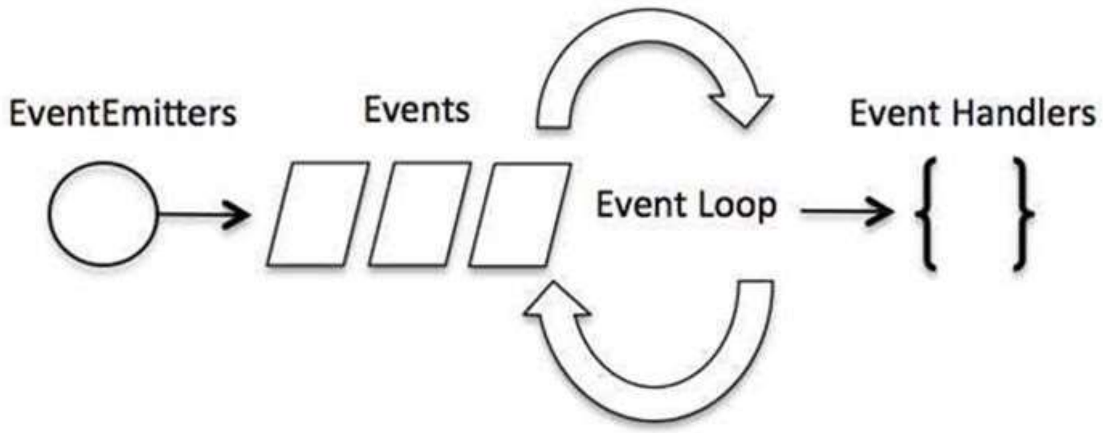
Node.js Events

In Node.js applications, Events and Callbacks concepts are used to provide concurrency. As Node.js applications are single threaded and every API of Node js are asynchronous. So it uses async function to maintain the concurrency. Node uses observer pattern. Node thread keeps an event loop and after the completion of any task, it fires the corresponding event which signals the event listener function to get executed.

Event Driven Programming

Node.js uses event driven programming. It means as soon as Node starts its server, it simply initiates its variables, declares functions and then simply waits for event to occur. It is the one of the reason why Node.js is pretty fast compared to other similar technologies.

There is a main loop in the event driven application that listens for events, and then triggers a callback function when one of those events is detected.



If several events take place during execution, or if they happen several times, standard callback pattern style doesn’t work as well. For instance, if you are interested in being notified every time data is available on a socket, the standard callback pattern is not very helpful. This is when the event emitter pattern can help.

You can use a standard interface to clearly separate the event emitter and the event listener. When you use an event emitter pattern, two or more objects are involved — the event emitter and one or more event listeners.

An event emitter is an object that — as the name says — emits events. An event listener is a part of the code that binds to the event emitter and listens for certain types of events.

EventEmitter Class

As we have seen in the previous section, EventEmitter class lies in the events module. It is accessible via the following code −

// Import events module var events =

require('events'); // Create an eventEmitter

object var eventEmitter = new

events.EventEmitter();

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

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

Common Patterns for EventEmitters

There are two common patterns that can be used to raise and bind an event using EventEmitter class in Node.js.

1. Return EventEmitter from a function
2. Extend the EventEmitter class

Return EventEmitter from a function

In this pattern, a constructor function returns an EventEmitter object, which was used to emit events inside a function. This EventEmitter object can be used to subscribe for the events.

Example 1:-

Code:-

const events = require("events"); const

eventEmitter = new events.EventEmitter();

function listner1(){ console.log("Event

received by listner1");

} function listner2(){ console.log("Event

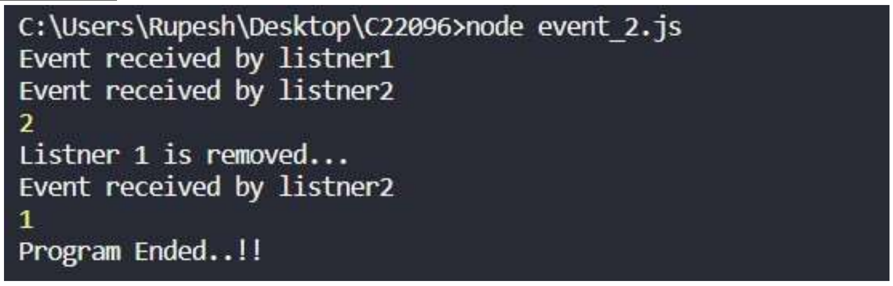
received by listner2");

} eventEmitter.addListener("write",listner1);

eventEmitter.on("write",listner2);

```
eventEmitter.emit("write");
console.log(eventEmitter.listenerCount("write"))
; eventEmitter.removeListener("write",listner1);
console.log("Listner 1 is removed...");
eventEmitter.emit("write");
console.log(eventEmitter.listenerCount("write"))
; console.log("Program Ended..!!");
```

Output:-



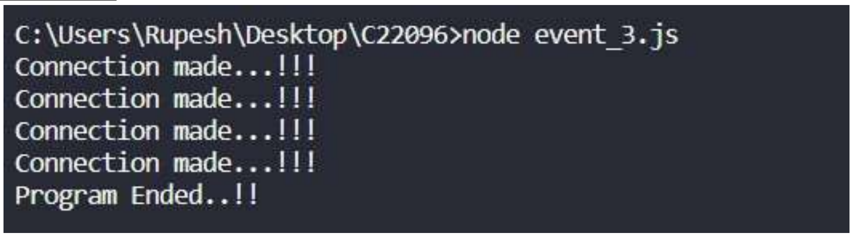
```
C:\Users\Rupesh\Desktop\C22096>node event_2.js
Event received by listner1
Event received by listner2
2
Listner 1 is removed...
Event received by listner2
1
Program Ended..!!
```

Example2:-

Code:-

```
const events = require("events");
const eventEmitter = new events.EventEmitter();

eventEmitter.on("Connection",handleConnectionEvent)
eventEmitter.emit("Connection");
eventEmitter.emit("Connection");
eventEmitter.emit("Connection");
eventEmitter.emit("Connection");

function handleConnectionEvent(){
   console.log("Connection made...!!!");
} console.log("Program
Ended..!!");
```

Output:-



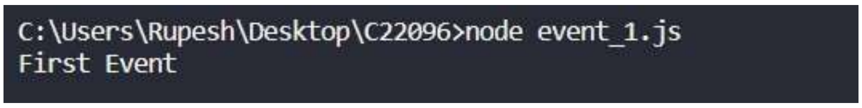
```
C:\Users\Rupesh\Desktop\C22096>node event_3.js
Connection made...!!!
Connection made...!!!
Connection made...!!!
Connection made...!!!
Program Ended..!!
```

Example 3:-

Code:-

```
const EventEmitter = require('events');
var eventEmitter = new EventEmitter();
eventEmitter.on('myevent',(msg)=>{
console.log(msg);
});
eventEmitter.emit('myevent',"First Event");
```

Output:-



```
C:\Users\Rupesh\Desktop\C22096>node event_1.js
First Event
```

PRACTICAL NO.3

Aim: Create an application to demonstrate Node.js Modules.

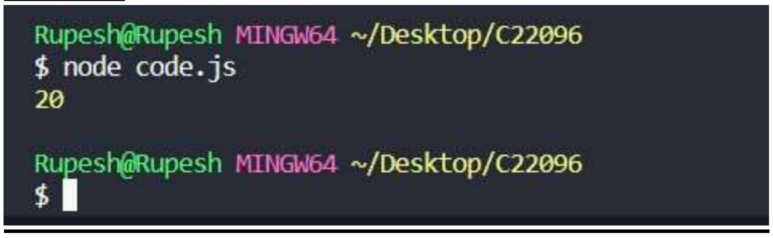
Functions include a return statement. However, if the return statement is missing, then the function returns undefined. After a function is defined, you can then call it. When you call a

function then it begins executing. Structure of function is as follows: function functionName (parameters) {
  //body of function
  //return statement }

Code:

```
function display_result(para){
console.log(para);
} function
calculate(x,y,mycallback){     let
sum=x+y;     mycallback(sum);
} calculate(10,10,display_result);
```

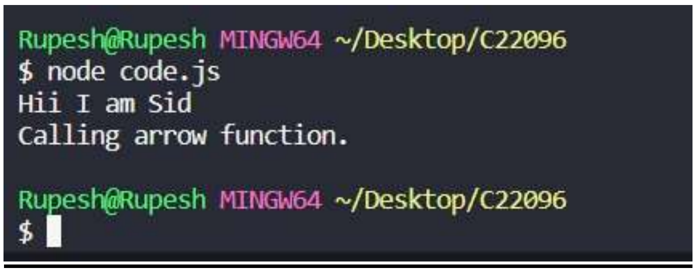
Output:-



```
Rupesh@Rupesh MINGW64 ~/Desktop/C22096
$ node code.js
20

Rupesh@Rupesh MINGW64 ~/Desktop/C22096
$
```

Arrow function provides a concise way to write functions in javascript. Unlike other functions, the value of this inside arrow functions is not dependent on how they are invoked or how they are defined. It depends only on its enclosing context. **Code**:

```
const message=function(){
console.log("Hii I am Sid");
}
setTimeout(message, 3000); setTimeout(()=>{
console.log("Calling arrow function.");
    },5000);
```



```
Rupesh@Rupesh MINGW64 ~/Desktop/C22096
$ node code.js
Hii I am Sid
Calling arrow function.

Rupesh@Rupesh MINGW64 ~/Desktop/C22096
$
```

PRACTICAL NO.4

Aim: Using File Handling Demonstrate all basic file operations(Create, Write, Read, Delete).

Node.js File System

Node.js includes fs module to access physical file system. The fs module is responsible for all the asynchronous or synchronous file I/O operations.

Let's see some of the common I/O operation examples using fs module.

Reading File

Use fs.readFile() method to read the physical file asynchronously.

Signature: fs.readFile(fileName

[,options], callback)

Parameter Description:

- filename: Full path and name of the file as a string.
- options: The options parameter can be an object or string which can include encoding and flag. The default encoding is utf8 and default flag is "r".
- callback: A function with two parameters err and fd. This will get called when readFile operation completes.

The following example demonstrates reading existing input1.txt asynchronously.

Code:- **input.txt:-** Hello World..!!

This is Node JS...!!

block.js:- var fs=require('fs');

fs.readFile("input.txt",function(err,data)

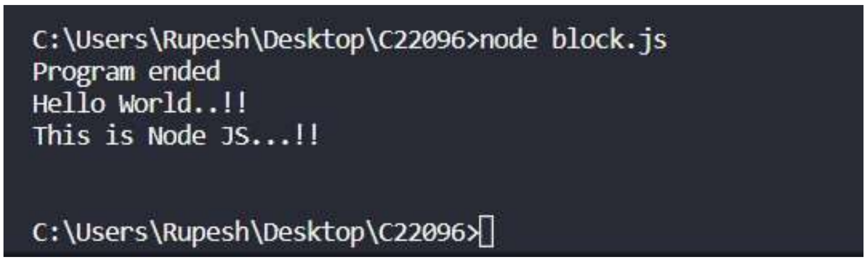
{ if(err) throw err;

console.log(data.toString());

}); console.log("Program

ended");

Output:-



```
C:\Users\Rupesh\Desktop\C22096>node block.js
Program ended
Hello World..!!
This is Node JS...!!


C:\Users\Rupesh\Desktop\C22096>
```

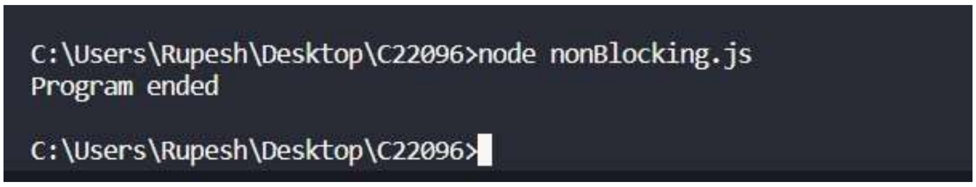
Use fs.readFileSync() method to read file synchronously as shown below:

Example: Reading File Synchronously:

Code:-

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

Output:-



```
C:\Users\Rupesh\Desktop\C22096>node nonBlocking.js
Program ended

C:\Users\Rupesh\Desktop\C22096>
```

Writing File

Use fs.writeFile() method to write data to a file. If file already exists then it overwrites the existing content otherwise it creates a new file and writes data into it.

Signature: fs.writeFile(filename, data[,

options], callback) Parameter Description:

- filename: Full path and name of the file as a string.
- Data: The content to be written in a file.
- options: The options parameter can be an object or string which can include encoding, mode and flag. The default encoding is utf8 and default flag is "r". ● callback: A function with two parameters err and fd. This will get called when write operation completes.

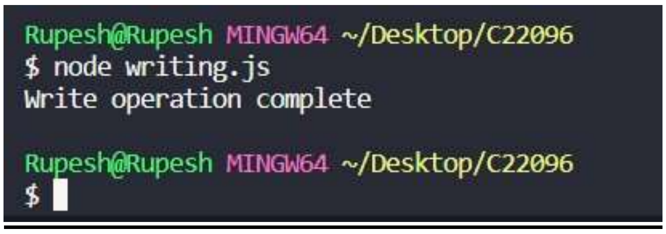
The following example creates a new file called test.txt and writes "Hello World" into it asynchronously.

Example: Writing File:

Code:-

```
var fs=require("fs"); fs.writeFile("input.txt"," Welcome to
NodeJS!!",function(err){     if(err){         return
console.log(err);
    }
else
    console.log("Write operation complete");
});
```

Output:-



```
Rupesh@Rupesh MINGW64 ~/Desktop/C22096
$ node writing.js
Write operation complete

Rupesh@Rupesh MINGW64 ~/Desktop/C22096
$
```

Deleting File:

Structure:

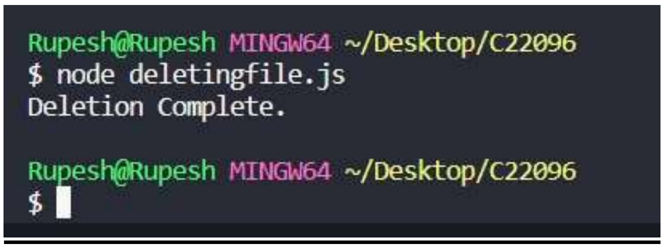
fs.unlink(path, callback);

Use fs.unlink() method to delete an existing file.

Code:

```
var fs=require("fs");
fs.unlink("input.txt",function(){
console.log("Deletion Complete.");
})
```

Output:



```
Rupesh@Rupesh MINGW64 ~/Desktop/C22096
$ node deletingfile.js
Deletion Complete.

Rupesh@Rupesh MINGW64 ~/Desktop/C22096
$
```

PRACTICAL NO.5

Aim: Create an HTTP server and perform operations on it.

HTTP Web Server:

Node.js Web Server

In this section, we will learn how to create a simple Node.js web server and handle HTTP requests.

To access web pages of any web application, you need a web server. The web server will handle all the http requests for the web application e.g IIS is a web server for ASP.NET web applications and Apache is a web server for PHP or Java web applications.

Node.js provides capabilities to create your own web server which will handle HTTP requests asynchronously. You can use IIS or Apache to run Node.js web application but it is recommended to use Node.js web server.

Create Node.js Web Server:-

Node.js makes it easy to create a simple web server that processes incoming requests asynchronously.The following example is a simple Node.js web server contained in server.js file.

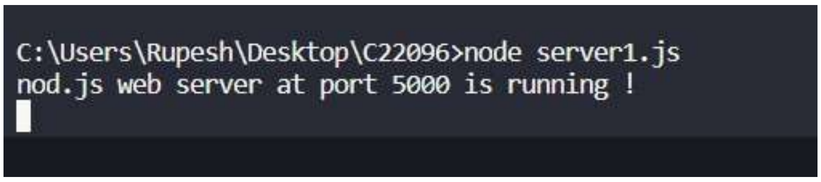
Example 1:-

Code:-

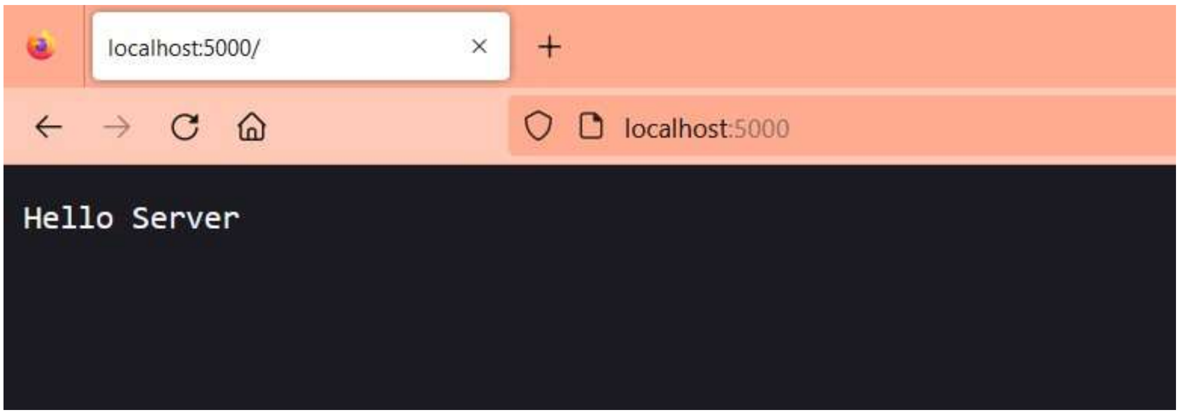
```
var http = require('http');
```

Now your application has access to the HTTP module, and is able to create a server:

var http = require('http'); var server =
http.createServer(function (req,res) {
res.write("Hello Server"); res.end();
}); server.listen(5000); console.log('nod.js web server at
port 5000 is running !') **Output:-**



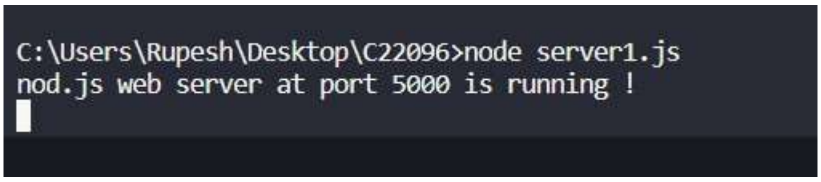
```
C:\Users\Rupesh\Desktop\C22096>node server1.js
nod.js web server at port 5000 is running !
```



```
Hello Server
```

In the above example, we import the http module using require() function. The http module is a core module of Node.js, so no need to install it using NPM. The next step is to call createServer() method of http and specify callback function with request and response parameter. Finally, call listen() method of server object which was returned from createServer() method with port number, to start listening to incoming requests on port 5000. You can specify any unused port here.

Run the above web server by writing node server.js command in command prompt or terminal window and it will display message as shown below.



```
C:\Users\Rupesh\Desktop\C22096>node server1.js
nod.js web server at port 5000 is running !
```

This is how you create a Node.js web server using simple steps. Now, let's see how to handle HTTP request and send response in Node.js web server.

Handle HTTP Request

The http.createServer() method includes request and response parameters which is supplied by Node.js. The request object can be used to get information about the current HTTP request e.g., url, request header, and data. The response object can be used to send a response for a current HTTP request.

The following example demonstrates handling HTTP request and response in Node.js.

Example 2:-

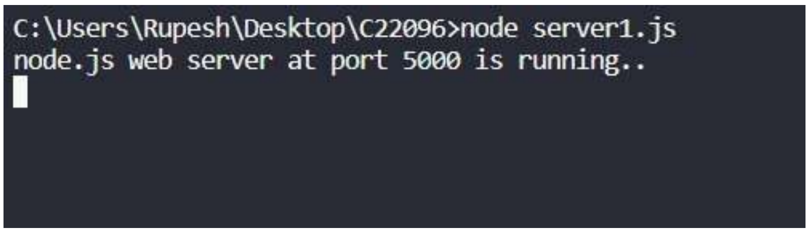
Code:-

```
var http = require('http'); var server = http.createServer(function(req,res){
if(req.url =='/'){        res.writeHead(200,{'Content-type': 'text/html'});
res.write('<html><body><p>This is home page.</p></body></html>');
res.end();
   }
   else if (req.url=="/student"){        res.writeHead(200,{'Content-
Type':'text/html'});        res.write('<html><body><p>This is student
page.</p></body></html>');        res.end();
   }
   else if(req.url=="/admin"){        res.writeHead(200,{'Content-
Type':'text/html'});        res.write('<html><body><p>This is admin
page.</p></body></html>');        res.end();
   }     else     res.end('Invalid
Request !');
}); server.listen(5000);
console.log('node.js web server at port 5000 is running..');
```

Output:-

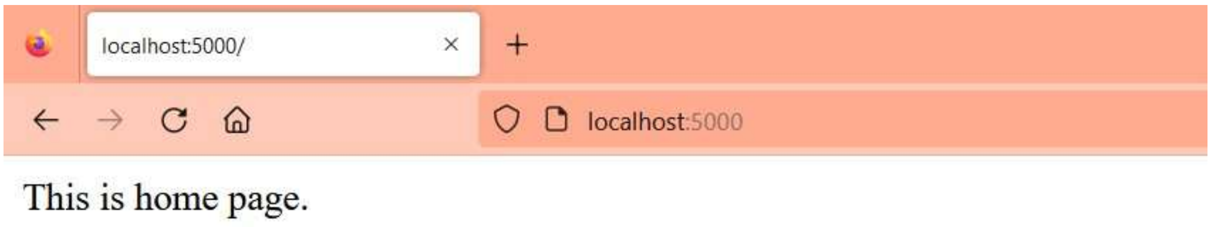
In the above example, req.url is used to check the url of the current request and based on that it sends the response. To send a response, first it sets the response header using writeHead() method and then writes a string as a response body using write() method. Finally, Node.js web server sends the response using end() method.

Now, run the above web server as shown below.



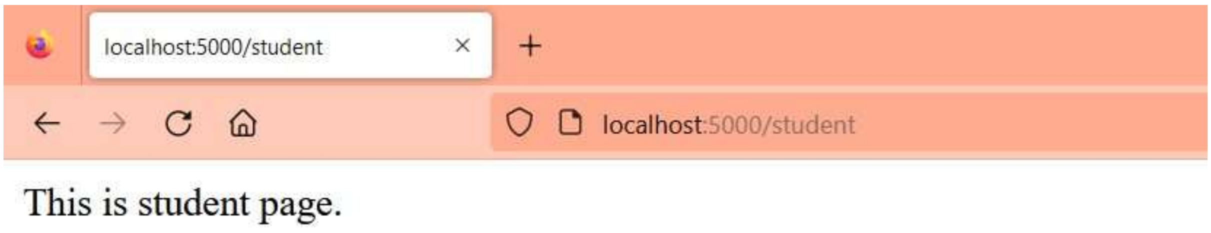
```
C:\Users\Rupesh\Desktop\C22096>node server1.js
node.js web server at port 5000 is running..
```

To test it, open your default browser and type- localhost:5000/



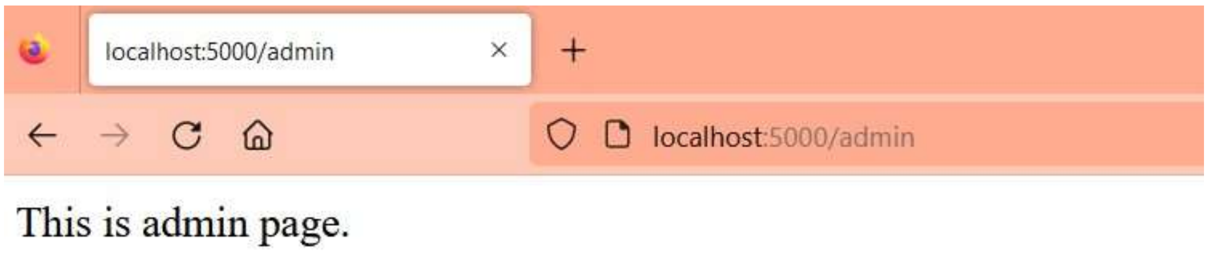
This is home page.

To test it, open your default browser and type- localhost:5000/student



This is student page.

To test it, open your default browser and type- localhost:5000/admin



This is admin page.

PRACTICAL NO.6

Aim: Create an application to establish a connection with the MySQL database and perform basic database operations on it.

Data Access in Node.js

Node.js supports all kinds of databases no matter if it is a relational database or NoSQL database. However, NoSQL databases like MongoDb are the best fit with Node.js.

To access the database from Node.js, you first need to install drivers for the database you want to use.

The following table lists important relational databases and respective drivers:

Relational Databases	Driver	NPM Command
MySQL	MySQL	npm install mysql

Connect and Communicate with a MySQL Database

To be able to experiment with the code examples, you should have MySQL installed on your computer.

You can download a free MySQL database at https://www.mysql.com/downloads/.

Install MySQL Driver

Once you have MySQL up and running on your computer, you can access it by using Node.js.

To access a MySQL database with Node.js, you need a MySQL driver. This tutorial will use the "mysql" module, downloaded from NPM.

To download and install the "mysql" module, open the Command Terminal and execute the following:

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

Now you have downloaded and installed a mysql database driver.

Node.js can use this module to manipulate the MySQL database:

```
var mysql = require('mysql');
```

Create Connection

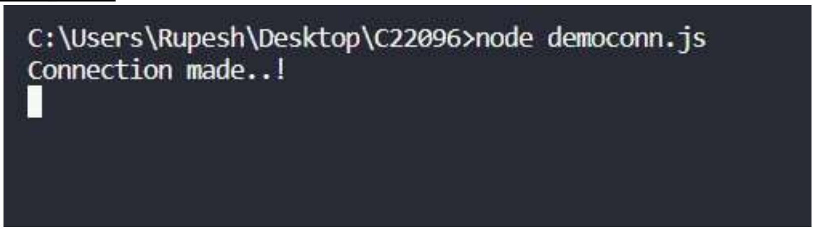
Start by creating a connection to the database.

Use the username and password from your MySQL database:

Code:-

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host:"localhost",
user: "root",
password:""
});
con.connect(function(err){
   if(err) throw err;
   console.log("Connection made..!");});
```

Output:-



```
C:\Users\Rupesh\Desktop\C22096>node democonn.js
Connection made..!
```

Adding data to the database

Creating a Database

Example:

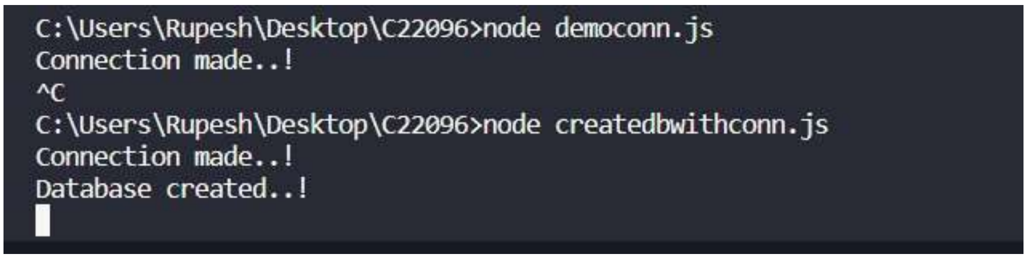
Code:-

```
var mysql = require('mysql');
var con = mysql.createConnection({
host:"localhost",
```

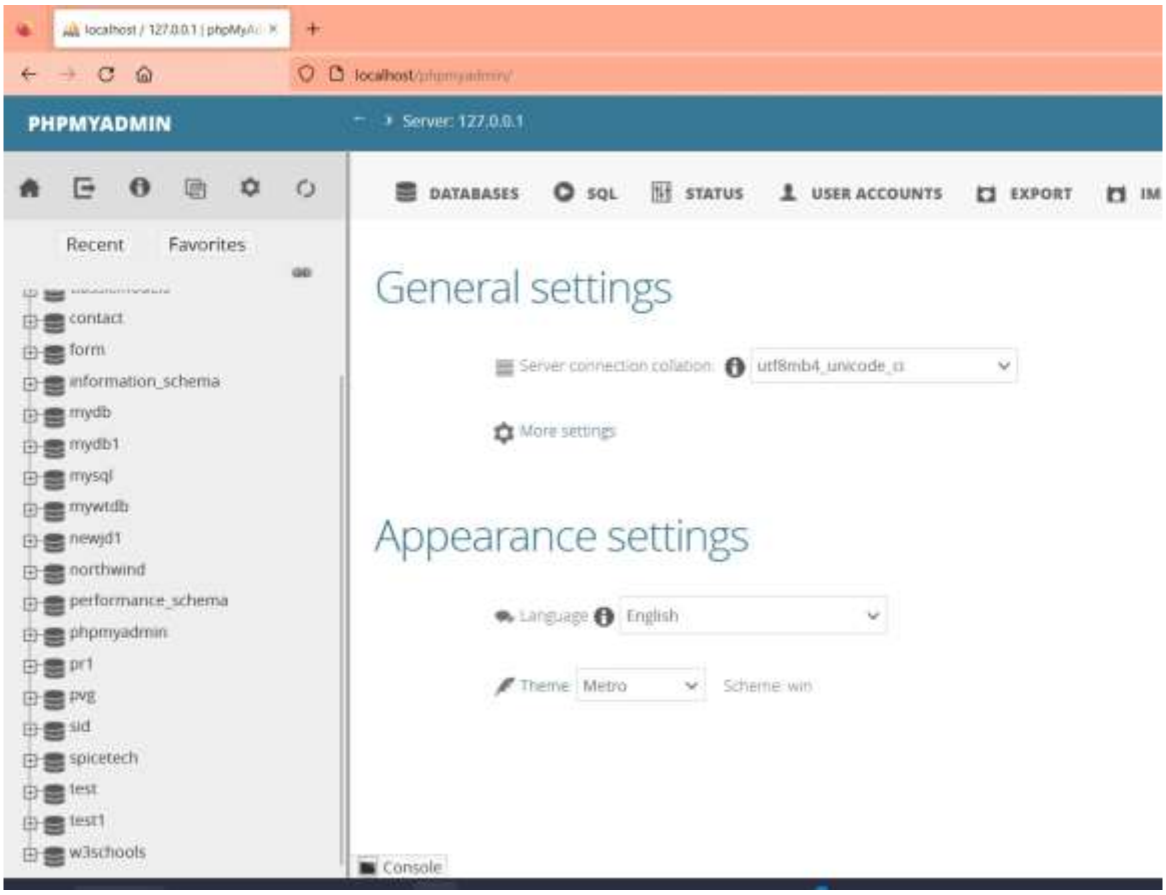
28

```
 user: "root",
password:""
});
con.connect(function(err){
if(err) throw err;
  console.log("Connection made..!");
  con.query("CREATE DATABASE pr1", function(err,result){
if(err) throw err;
     console.log("Database created..!");
  }); });
```

Output:-



```
C:\Users\Rupesh\Desktop\C22096>node democonn.js
Connection made..!
^C
C:\Users\Rupesh\Desktop\C22096>node createdbwithconn.js
Connection made..!
Database created..!
```



localhost/phpmyadmin/

PHPMYADMIN — Server: 127.0.0.1

DATABASES SQL STATUS USER ACCOUNTS EXPORT IM

Recent Favorites

contact
form
information_schema
mydb
mydb1
mysql
mywtdb
newjd1
northwind
performance_schema
phpmyadmin
pr1
pvg
sid
spicetech
test
test1
w3schools

General settings

Server connection collation: utf8mb4_unicode_ci

More settings

Appearance settings

Language: English

Theme: Metro Scheme: win

Console

Creating a Table

To create a table in MySQL, use the "CREATE TABLE" statement.

Make sure you define the name of the database when you create the connection: **Primary**

Key

When creating a table, you should also create a column with a unique key for each record.

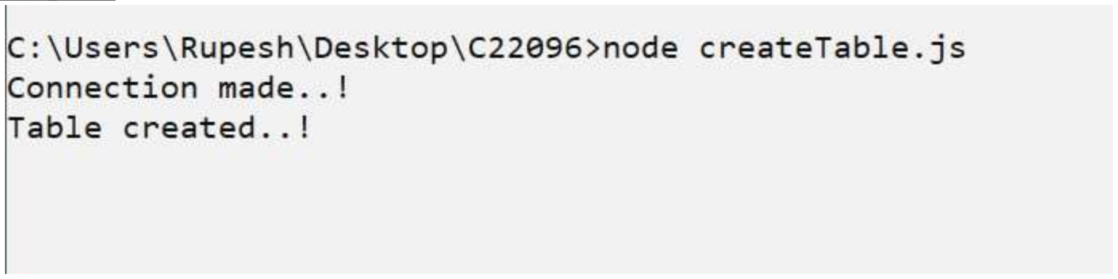
This can be done by defining a column as "INT AUTO_INCREMENT PRIMARY KEY" which will insert a unique number for each record. Starting at 1, and increased by one for each record.

Example:-

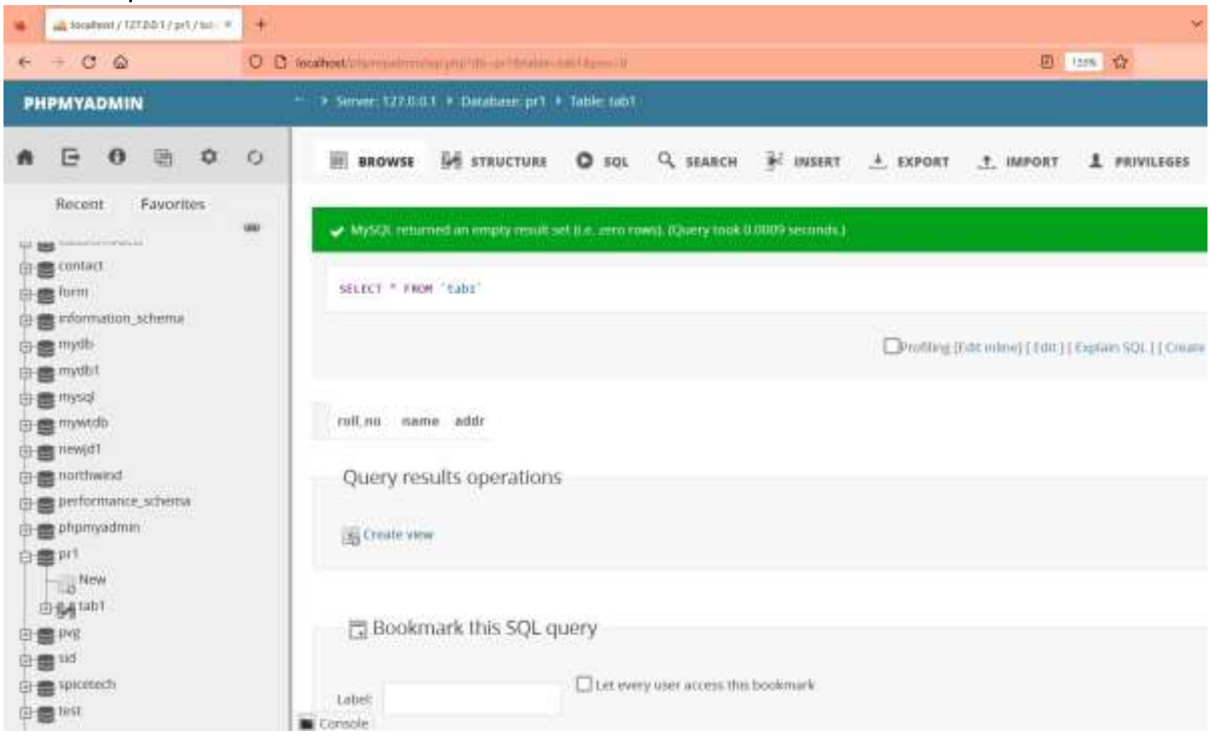
Code:-

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host:"localhost",
user: "root",
password:"",
database:"pr1"
});
con.connect(function(err){
if(err) throw err;
   console.log("Connection made..!");
   con.query("CREATE TABLE tab1 (roll_no INT, name VARCHAR(20), addr
VARCHAR(15))", function(err,result){
if(err) throw err;
       console.log("Table created..!");
   });
});
```

Output:-



```
C:\Users\Rupesh\Desktop\C22096>node createTable.js
Connection made..!
Table created..!
```



```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0009 seconds.)

SELECT * FROM `tab1`
```

```
roll_no   name   addr
```

ALTER TABLE

If the table already exists, use the ALTER TABLE keyword:

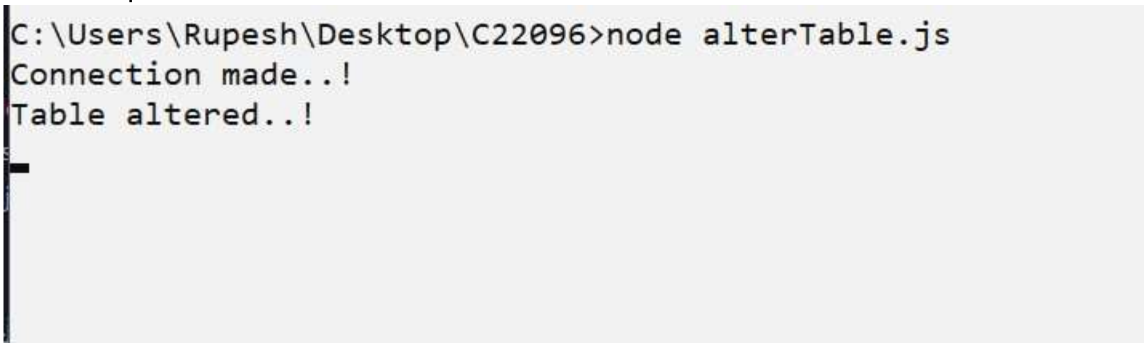
Example:-

Create primary key on an existing table:

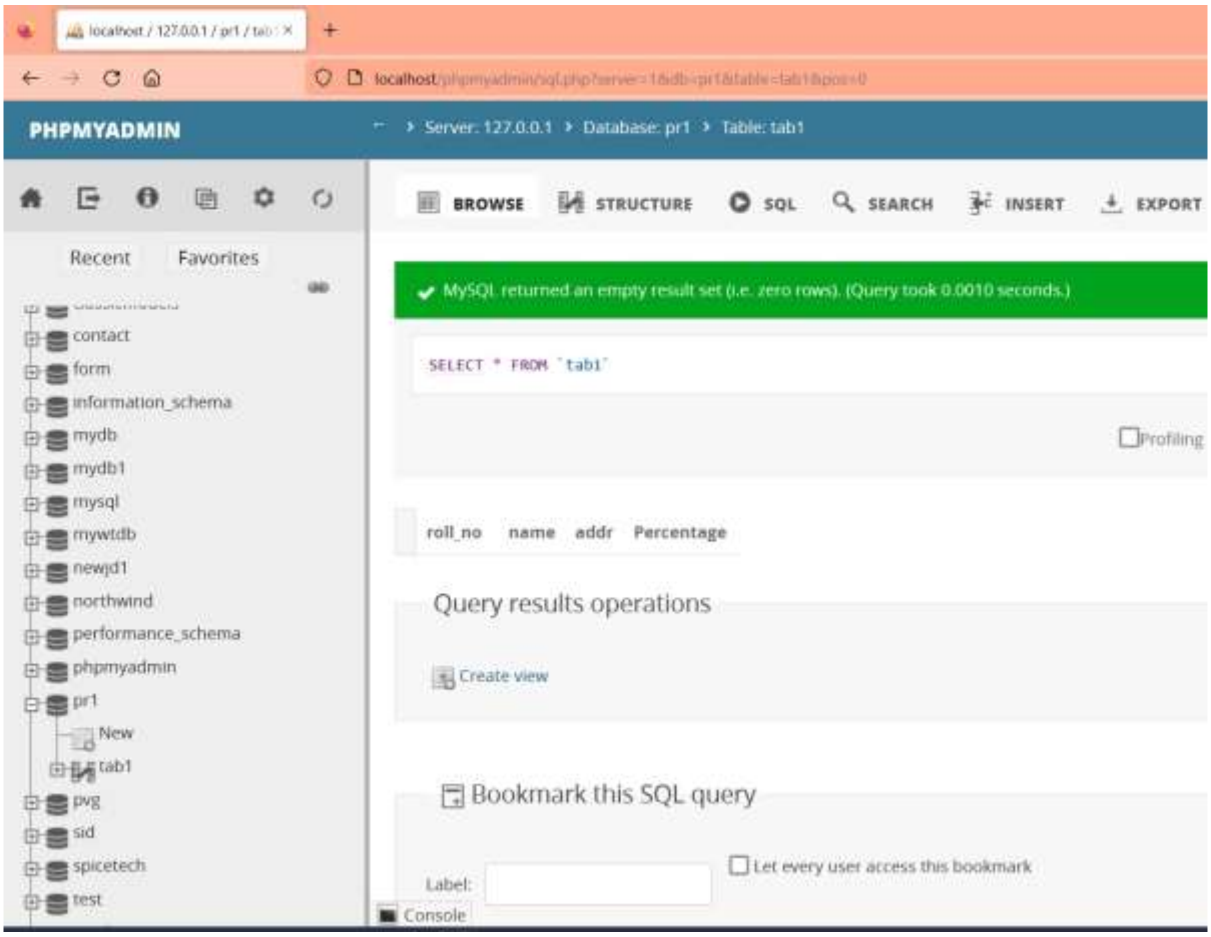
Code:-

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host:"localhost",
user: "root",
password:"",
database:"pr1"
});
con.connect(function(err){
if(err) throw err;
   console.log("Connection made..!");
   var sql = "ALTER TABLE tab1 ADD COLUMN Percentage INT  ;";
con.query(sql);    if(err) throw err;
       console.log("Table altered..!");
});
```

Output:-



```
C:\Users\Rupesh\Desktop\C22096>node alterTable.js
Connection made..!
Table altered..!
```



MySQL returned an empty result set (i.e. zero rows). (Query took 0.0010 seconds.)

```
SELECT * FROM `tab1`
```

roll_no	name	addr	Percentage

Insert record into Table

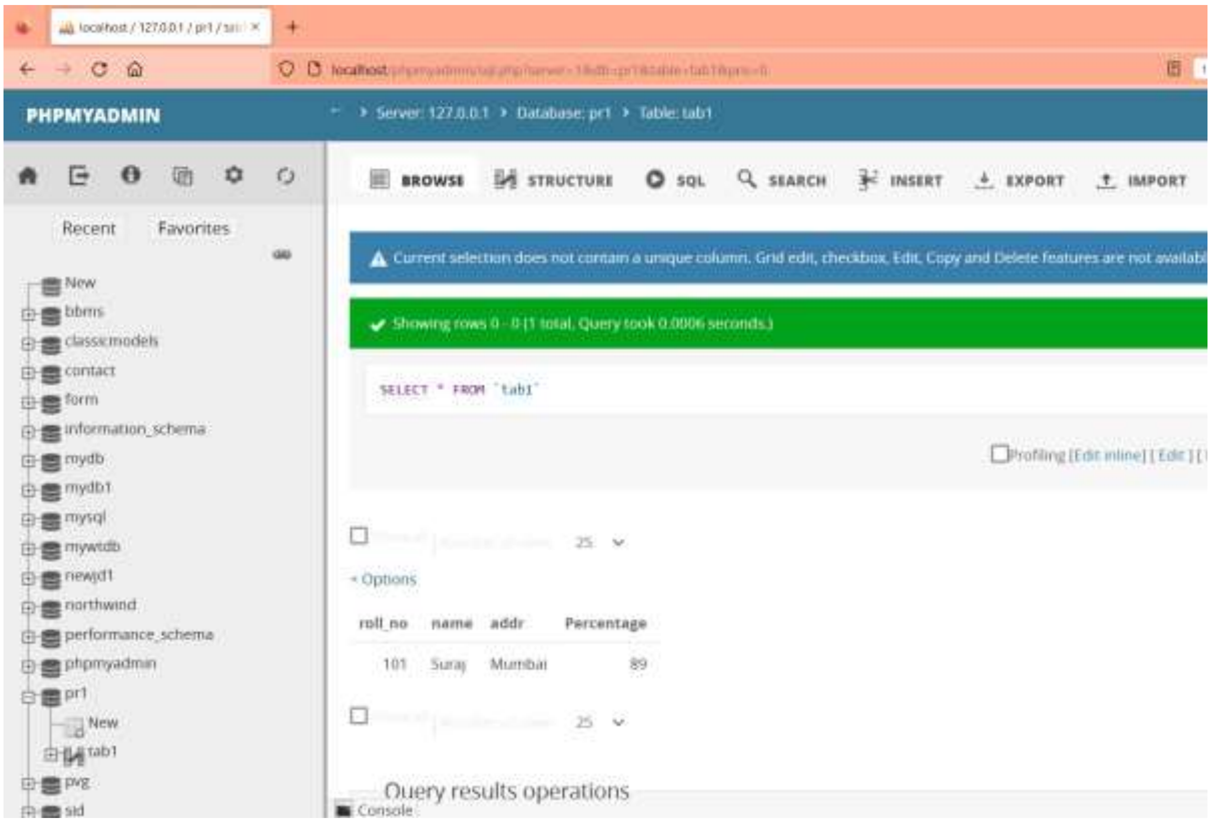
Insert a record in the "tab1" table: <u>Code:-</u>

var mysql = require('mysql');

```
var con = mysql.createConnection({
  host:"localhost",
user: "root",
password:"",
database:"pr1"
});
con.connect(function(err){
if(err) throw err;
   console.log("Connection made..!");
  var sql = "INSERT INTO tab1 VALUES(101,'Suraj','Mumbai',89)";
con.query(sql, function(err,result){        if(err) throw err;
       console.log("Records inserted..!");
   });
});
```

Output:-

```
C:\Users\Rupesh\Desktop\C22096>node insertRecord.js
Connection made..!
Records inserted..!
```



Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not availabl

Showing rows 0 - 0 (1 total, Query took 0.0006 seconds.)

```
SELECT * FROM `tab1`
```

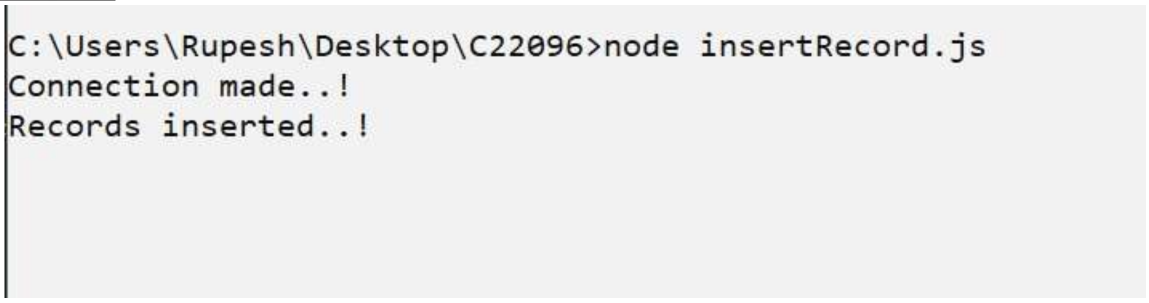
roll_no	name	addr	Percentage
101	Suraj	Mumbai	89

Query results operations

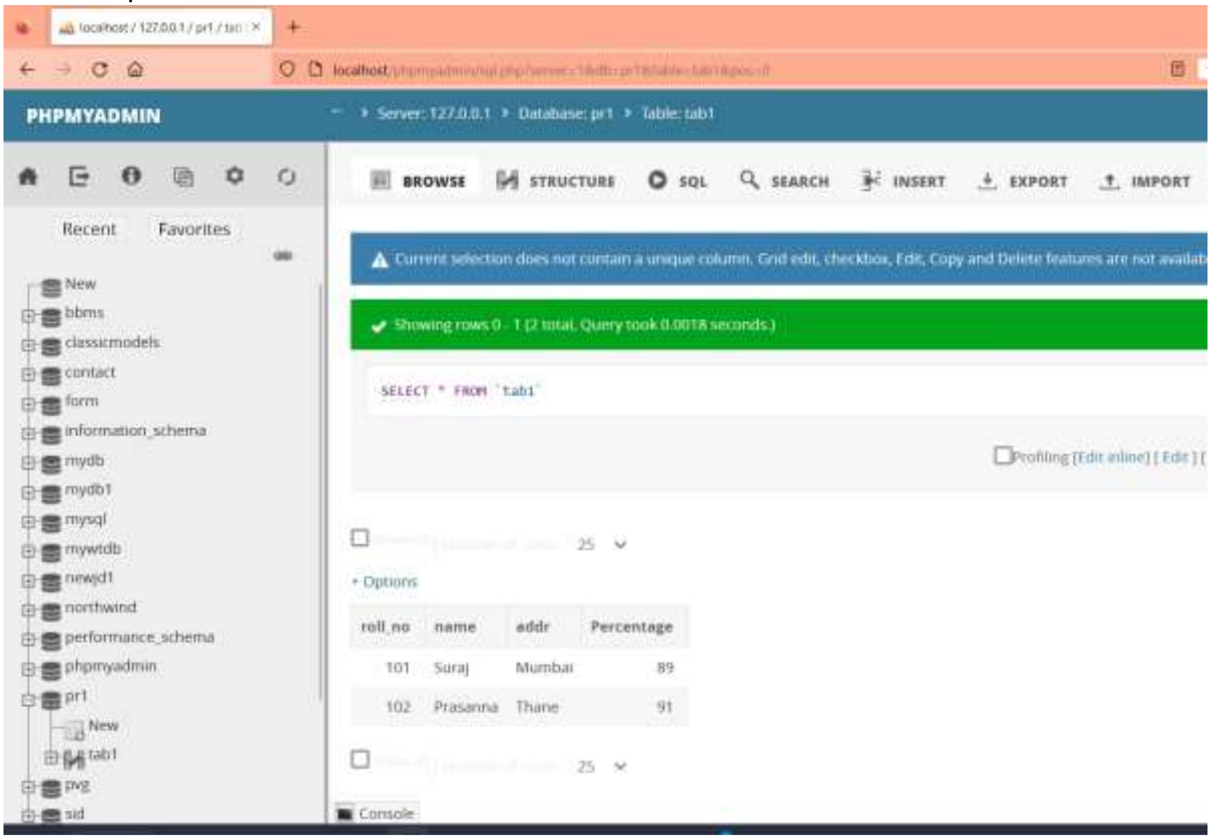
Code:-

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host:"localhost",
user: "root",
password:"",
database:"pr1"
});
con.connect(function(err){
if(err) throw err;
   console.log("Connection made..!");
var sql = "INSERT INTO tab1 VALUES(102,'Prasanna','Thane',91)";
con.query(sql, function(err,result){        if(err) throw err;
       console.log("Records inserted..!");
   });
});
```

Output:-



```
C:\Users\Rupesh\Desktop\C22096>node insertRecord.js
Connection made..!
Records inserted..!
```



Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not availab

Showing rows 0 - 1 (2 total, Query took 0.0018 seconds.)

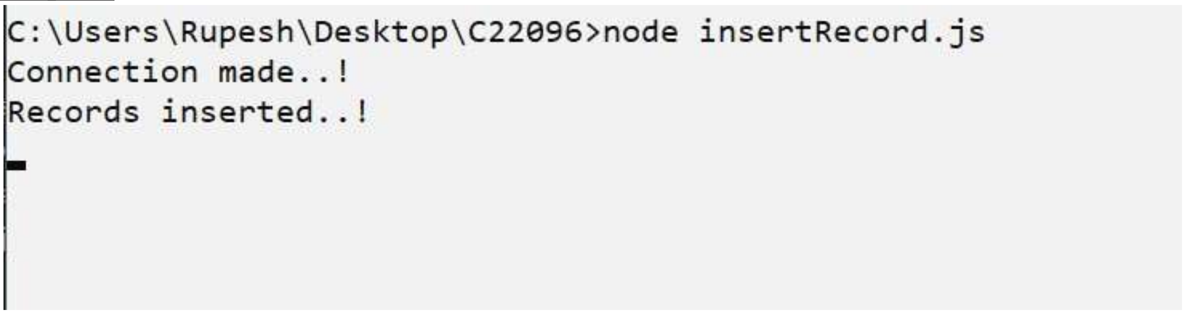
```
SELECT * FROM `tab1`
```

roll_no	name	addr	Percentage
101	Suraj	Mumbai	89
102	Prasanna	Thane	91

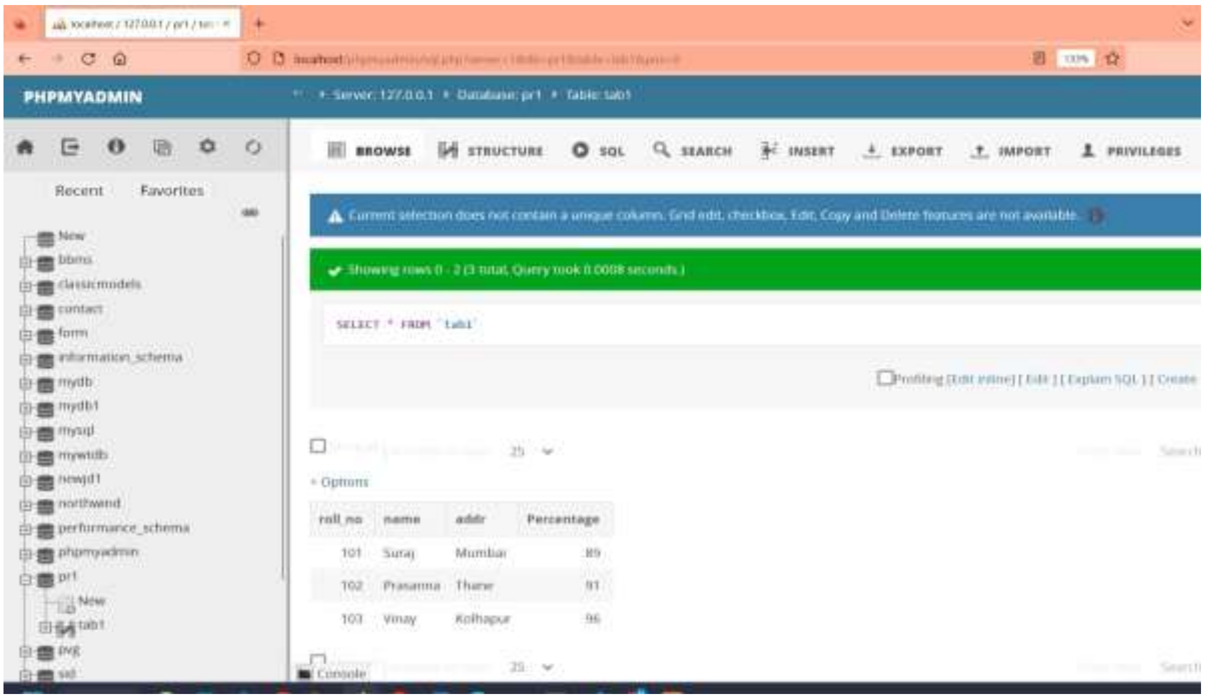
<u>Code:-</u>

```
var mysql = require('mysql');

var con = mysql.createConnection({
 host:"localhost",
user: "root",
password:"",
database:"pr1"
});
con.connect(function(err){
if(err) throw err;
  console.log("Connection made..!");
  var sql = "INSERT INTO tab1 VALUES(103,'Vinay','Pune',96)";

  con.query(sql, function(err,result){
if(err) throw err;
      console.log("Records inserted..!");
  });
});
```

Output:-



```
C:\Users\Rupesh\Desktop\C22096>node insertRecord.js
Connection made..!
Records inserted..!
```



Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 2 (3 total, Query took 0.0008 seconds.)

```
SELECT * FROM `tab1`
```

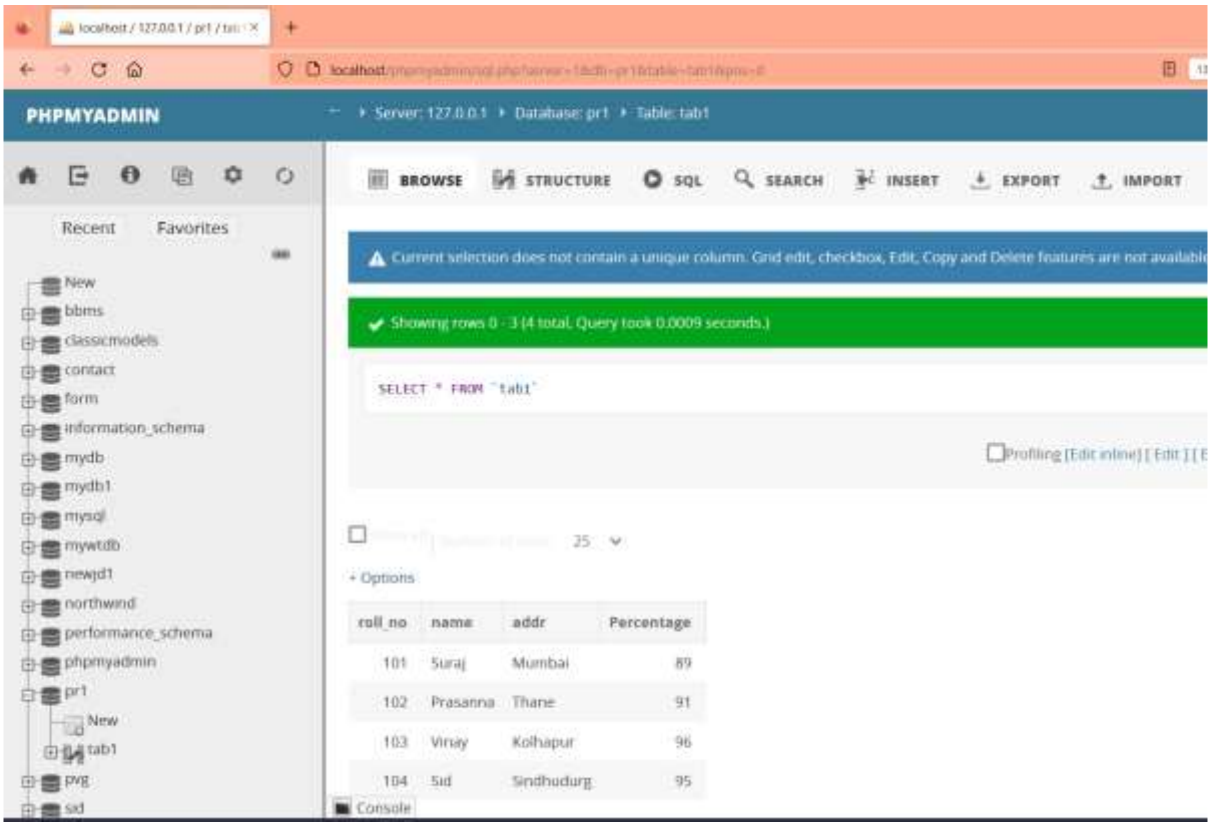
roll_no	name	addr	Percentage
101	Suraj	Mumbai	89
102	Prasanna	Thane	91
103	Vinay	Kolhapur	96

Code:-

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host:"localhost",
user: "root",
password:"",
database:"pr1"
});
con.connect(function(err){
   if(err) throw err;
   console.log("Connection made..!");
   var sql = "INSERT INTO tab1 VALUES(104,'Sid','Sindhudurg',95)";

   con.query(sql, function(err,result){
if(err) throw err;
       console.log("Records inserted..!");
   });
});
```

Output:-

```
C:\Users\Rupesh\Desktop\C22096>node insertRecord.js
Connection made..!
Records inserted..!
```



Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not availabl

Showing rows 0 - 3 (4 total, Query took 0.0009 seconds.)

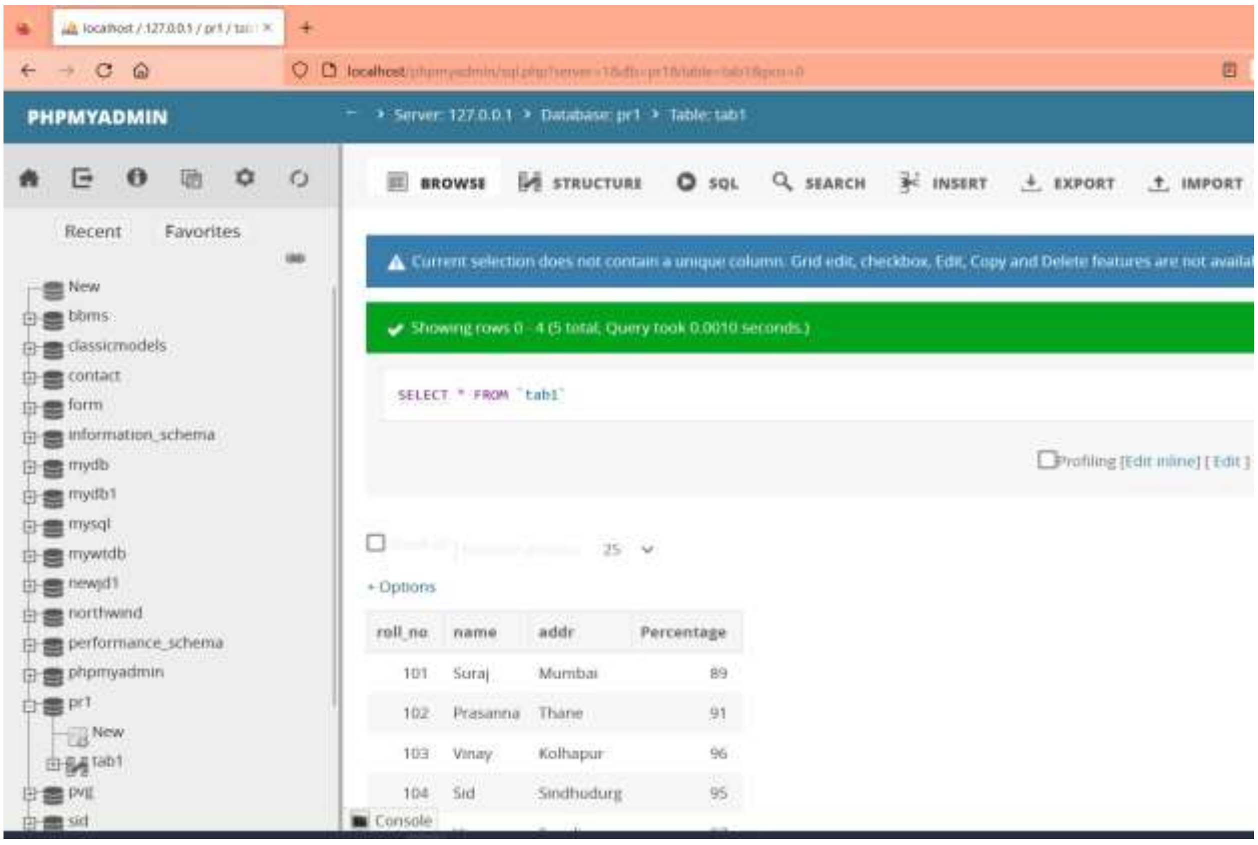
```
SELECT * FROM `tab1`
```

roll_no	name	addr	Percentage
101	Suraj	Mumbai	89
102	Prasanna	Thane	91
103	Vinay	Kolhapur	96
104	Sid	Sindhudurg	95

Code:-

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host:"localhost",
user: "root",
password:"",
database:"pr1"
});
con.connect(function(err){
if(err) throw err;
   console.log("Connection made..!");
  var sql = "INSERT INTO tab1 VALUES(105,'Vinu','Sangli',97)";
con.query(sql, function(err,result){       if(err) throw err;
      console.log("Records inserted..!");
  });
});
```

Output:-

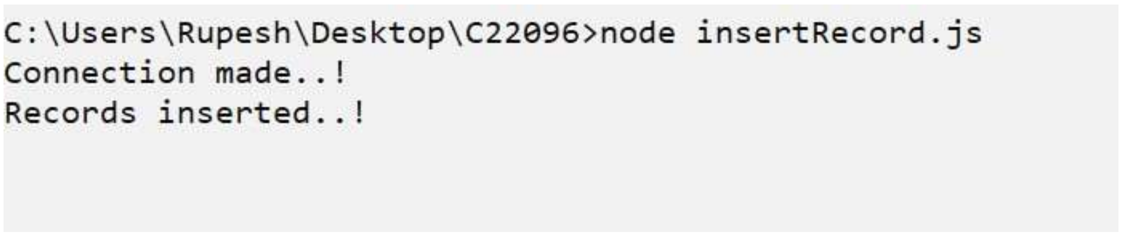


Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not availal

Showing rows 0 - 4 (5 total, Query took 0.0010 seconds.)

```
SELECT * FROM `tab1`
```

roll_no	name	addr	Percentage
101	Suraj	Mumbai	89
102	Prasanna	Thane	91
103	Vinay	Kolhapur	96
104	Sid	Sindhudurg	95



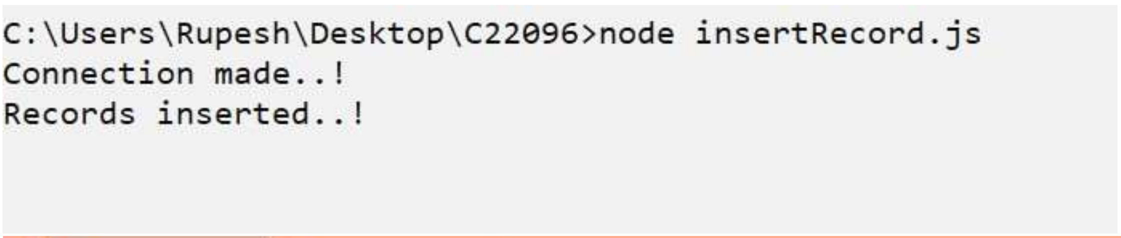
```
C:\Users\Rupesh\Desktop\C22096>node insertRecord.js
Connection made..!
Records inserted..!
```

Code:-

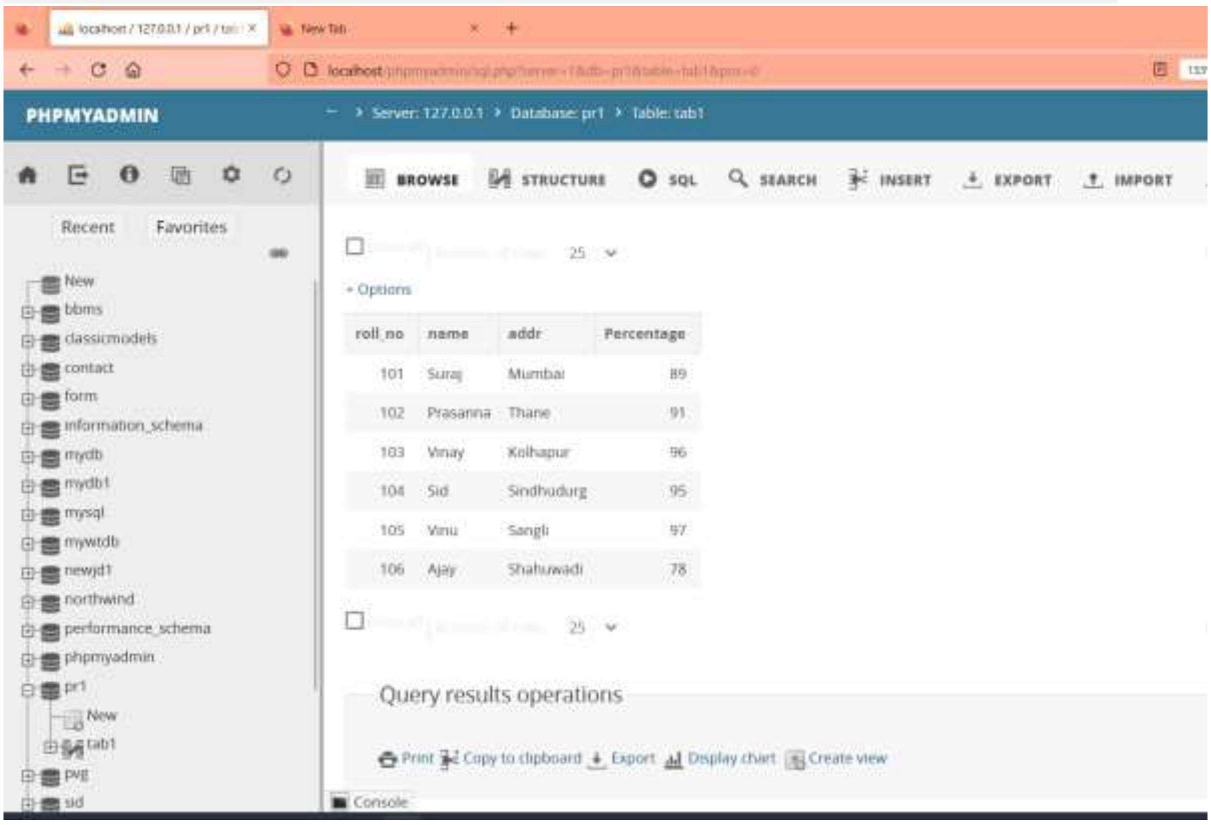
```
var mysql = require('mysql');

var con = mysql.createConnection({
  host:"localhost",
user: "root",
password:"",
database:"pr1"
});
```

```
con.connect(function(err){
if(err) throw err;
  console.log("Connection made..!");
  var sql = "INSERT INTO tab1 VALUES(106,'Ajay','Shahuwadi',78)";
con.query(sql, function(err,result){       if(err) throw err;
     console.log("Records inserted..!");
  });
});
```

<u>Output:-</u>



```
C:\Users\Rupesh\Desktop\C22096>node insertRecord.js
Connection made..!
Records inserted..!
```



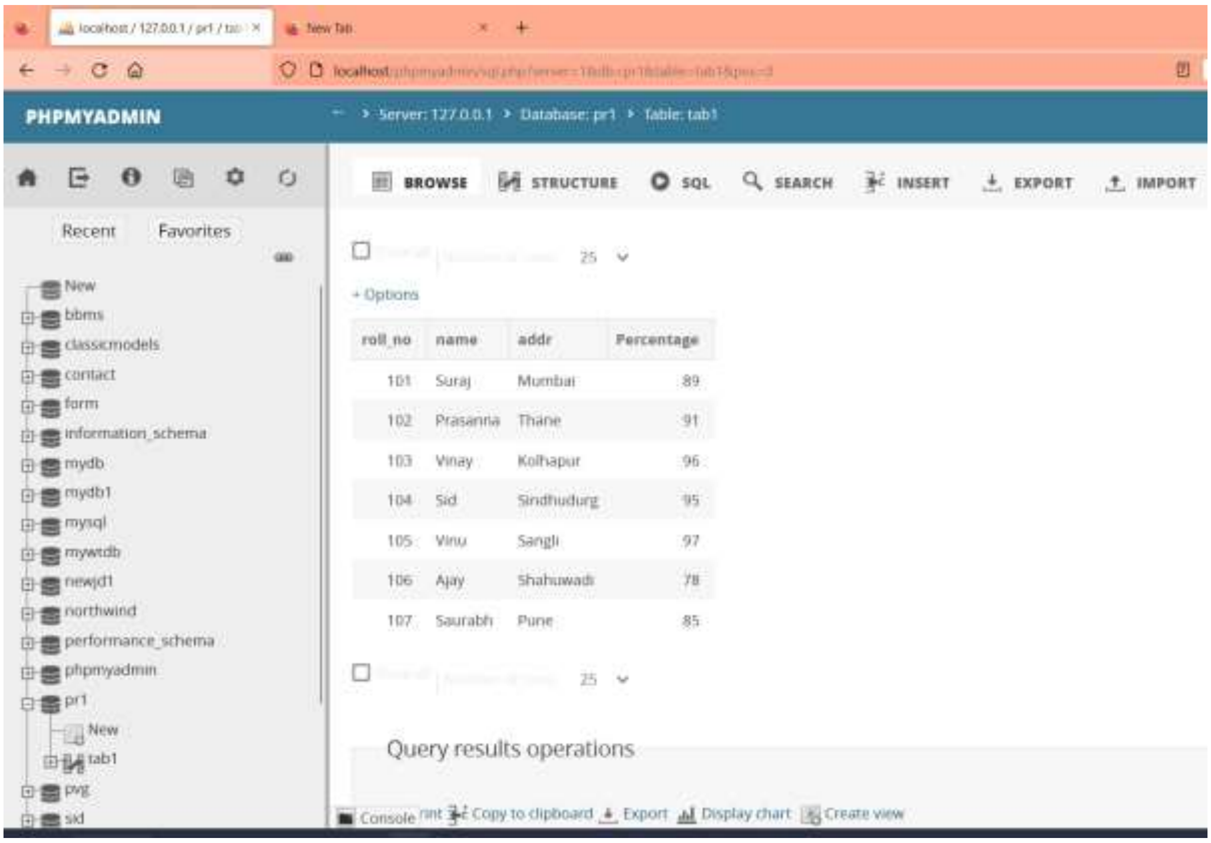
roll_no	name	addr	Percentage
101	Suraj	Mumbai	89
102	Prasanna	Thane	91
103	Vinay	Kolhapur	96
104	Sid	Sindhudurg	95
105	Vinu	Sangli	97
106	Ajay	Shahuwadi	78

<u>Code:-</u>

```
var mysql = require('mysql');

var con = mysql.createConnection({
 host:"localhost",
user: "root",
password:"",
database:"pr1"
});
con.connect(function(err){
if(err) throw err;
  console.log("Connection made..!");
var sql = "INSERT INTO tab1 VALUES(107,'Saurabh','Pune',85)";   con.query(sql,
function(err,result){        if(err) throw err;
      console.log("Records inserted..!");
  });
});
```

Output:-

```
C:\Users\Rupesh\Desktop\C22096>node insertRecord.js
Connection made..!
Records inserted..!
```

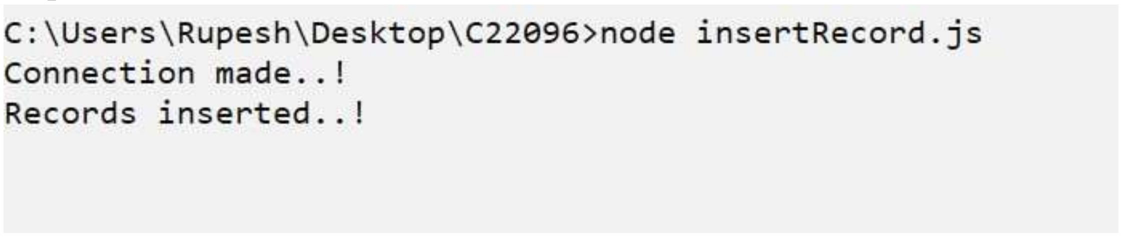


roll_no	name	addr	Percentage
101	Suraj	Mumbai	89
102	Prasanna	Thane	91
103	Vinay	Kolhapur	96
104	Sid	Sindhudurg	95
105	Vinu	Sangli	97
106	Ajay	Shahuwadi	78
107	Saurabh	Pune	85

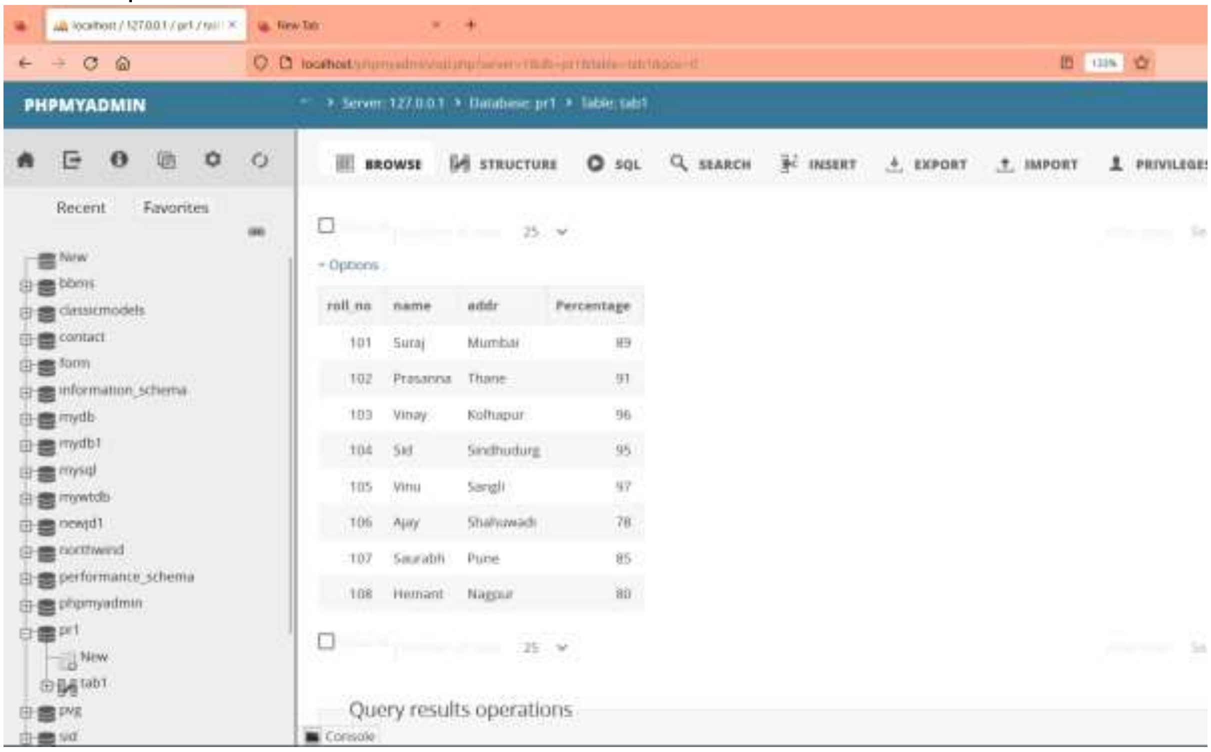
Code:-

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host:"localhost",
user: "root",
password:"",
database:"pr1"
});
con.connect(function(err){
if(err) throw err;
   console.log("Connection made..!");
   var sql = "INSERT INTO tab1 VALUES(108,'Hemant','Nagpur',80)";
con.query(sql, function(err,result){
       if(err) throw err;
       console.log("Records inserted..!");
   });
});
```

Output:-



```
C:\Users\Rupesh\Desktop\C22096>node insertRecord.js
Connection made..!
Records inserted..!
```



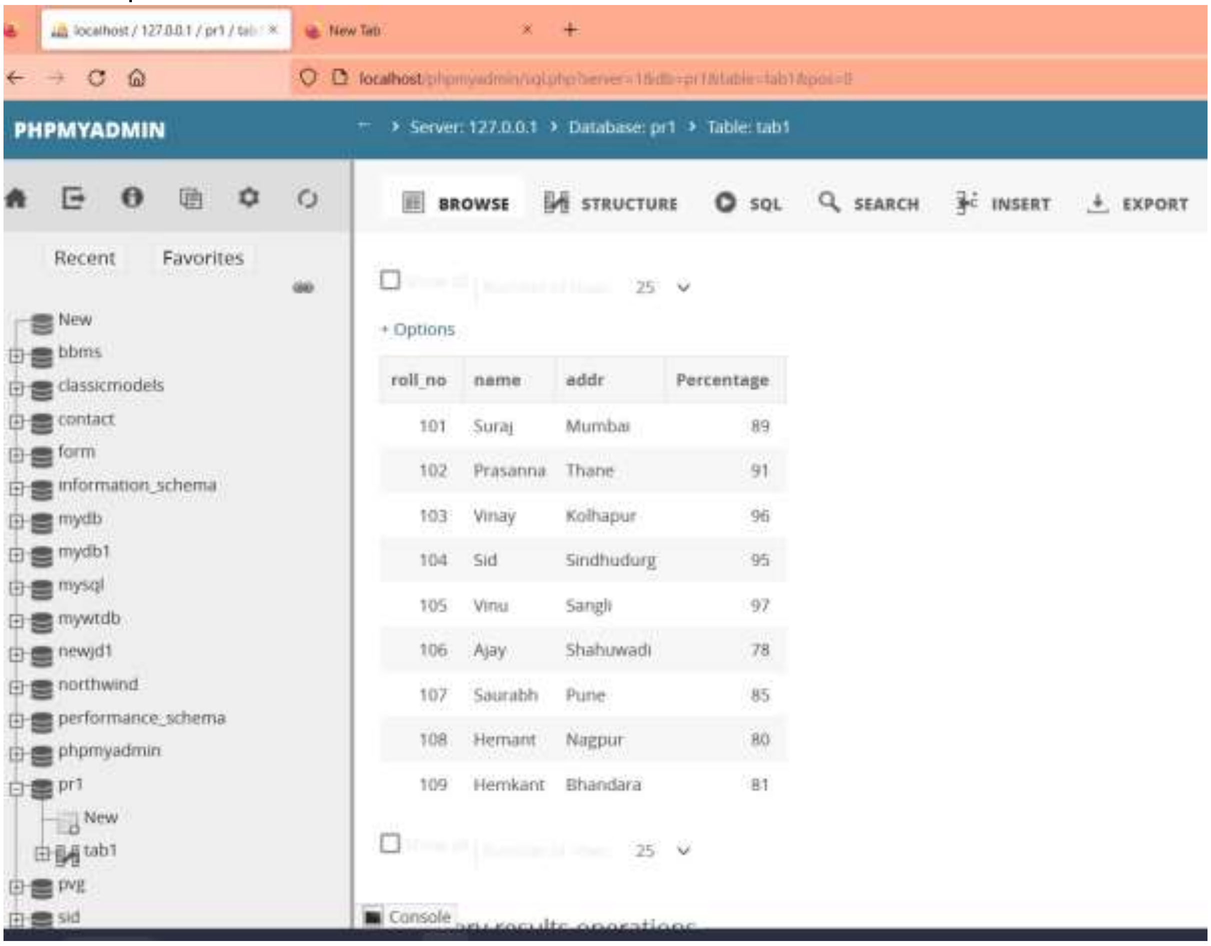
roll_no	name	addr	Percentage
101	Suraj	Mumbai	89
102	Prasanna	Thane	91
103	Vinay	Kolhapur	96
104	Sid	Sindhudurg	95
105	Vinu	Sangli	97
106	Ajay	Shahuwadi	78
107	Saurabh	Pune	85
108	Hemant	Nagpur	80

Code:-

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host:"localhost",
user: "root",
password:"",
database:"pr1"
});
con.connect(function(err){
   if(err) throw err;
   console.log("Connection made..!");
   var sql = "INSERT INTO tab1 VALUES(109,'Hemkant','Bhandara',81)";
con.query(sql, function(err,result){        if(err) throw err;
       console.log("Records inserted..!");
   });
});
```

Output:-

```
C:\Users\Rupesh\Desktop\C22096>node insertRecord.js
Connection made..!
Records inserted..!
```



roll_no	name	addr	Percentage
101	Suraj	Mumbai	89
102	Prasanna	Thane	91
103	Vinay	Kolhapur	96
104	Sid	Sindhudurg	95
105	Vinu	Sangli	97
106	Ajay	Shahuwadi	78
107	Saurabh	Pune	85
108	Hemant	Nagpur	80
109	Hemkant	Bhandara	81

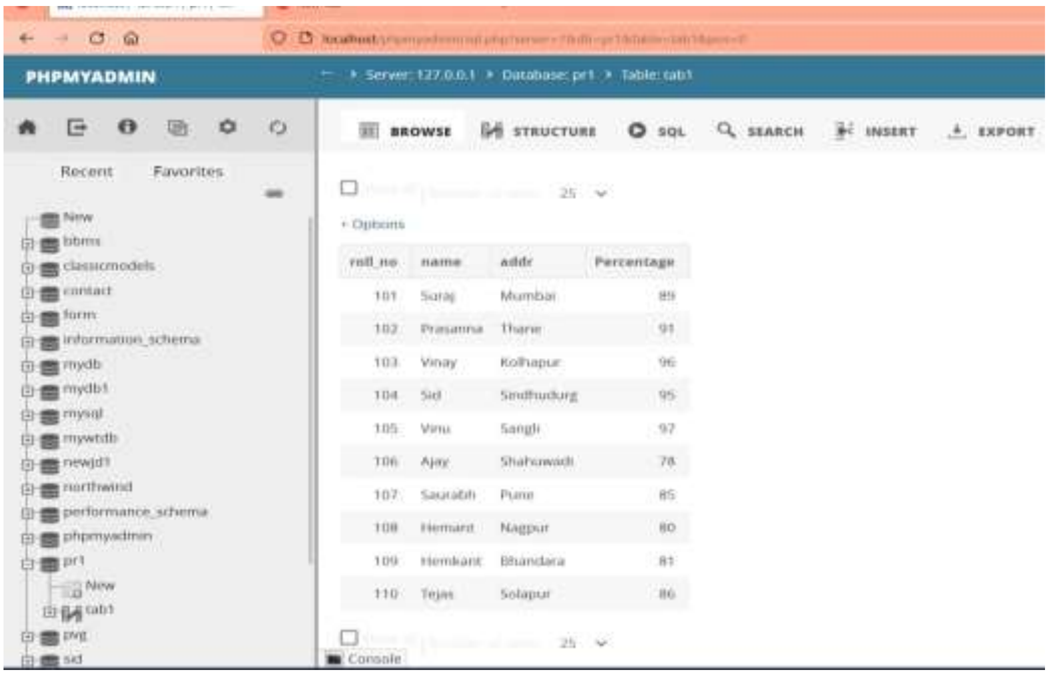
Code:-

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host:"localhost",
user: "root",
password:"",
database:"pr1"
});
con.connect(function(err){
if(err) throw err;
   console.log("Connection made..!");
var sql = "INSERT INTO tab1 VALUES(110,'Tejas','Solapur',86)";
   con.query(sql, function(err,result){
if(err) throw err;
       console.log("Records inserted..!");
   });
});
```

Name: Rupesh Nana Patil Roll No: C22096

Output:-

```
C:\Users\Rupesh\Desktop\C22096>node insertRecord.js
Connection made..!
Records inserted..!
```



roll_no	name	addr	Percentage
101	Suraj	Mumbai	89
102	Prasanna	Thane	91
103	Vinay	Kolhapur	96
104	Sid	Sindhudurg	95
105	Vinu	Sangli	97
106	Ajay	Shahuwadi	78
107	Saurabh	Pune	85
108	Hemant	Nagpur	80
109	Hemkant	Bhandara	81
110	Tejas	Solapur	86

Name: Anonymous King

Roll No: C99999

Reading data

Selecting From a Table

To select data from a table in MySQL, use the "SELECT" statement.

Example:-

Code:-

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host:"localhost",
user: "root",
password:"",
database:"pr1"
});
con.connect(function(err){
   if(err) throw err;
   console.log("Connection made..!");    var
sql = "SELECT * FROM tab1";
con.query(sql, function(err,result,fields){
if(err) throw err;        console.log(result);
   });
});
```

Output:-

```
Connection made..!

[

  RowDataPacket {

    roll_no: 101,

name: 'Suraj',

addr: 'Mumbai',

    Percentage: 89

  },
```

RowDataPacket {

Name: Anonymous King Roll No: C99999

```
    roll_no: 102,
name: 'Prasanna',
addr: 'Thane',
    Percentage: 91
  },
  RowDataPacket {
    roll_no: 103,
name: 'Vinay',
addr: 'Kolhapur',
    Percentage: 96
  },
  RowDataPacket {
    roll_no: 104,
name: 'Sid',     addr:
'Sindhudurg',
    Percentage: 95
  },
  RowDataPacket {
    roll_no: 105,
name: 'Vinu',
addr: 'Sangli',
    Percentage: 97
  },
  RowDataPacket {
```

roll_no: 106,

Name: Anonymous King Roll No: C99999

```
    name: 'Ajay',
addr: 'Shahuwadi',
    Percentage: 78
  },
  RowDataPacket {
    roll_no: 107,
name: 'Saurabh',
addr: 'Pune',
    Percentage: 85
  },
  RowDataPacket {
    roll_no: 108,
name: 'Hemant',
addr: 'Nagpur',
    Percentage: 80
  },
  RowDataPacket {
    roll_no: 109,
name: 'Hemkant',
addr: 'Bhandara',
    Percentage: 81
  },
  RowDataPacket {
```

roll_no: 110,

name: 'Tejas',

```
    addr: 'Solapur',
    Percentage: 86
  }
]
```

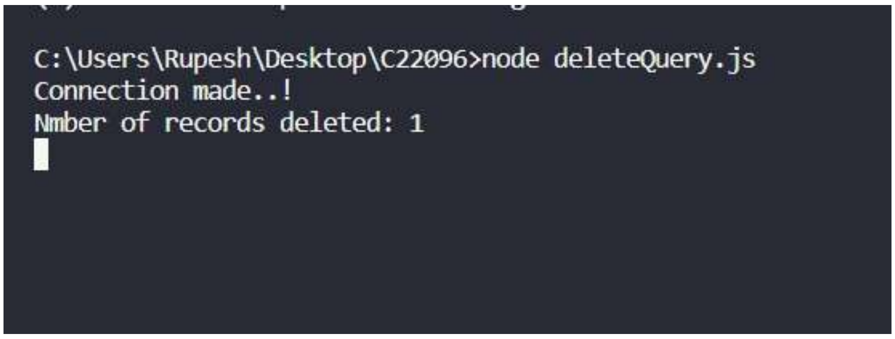
Delete Record

You can delete records from an existing table by using the "DELETE FROM" statement:

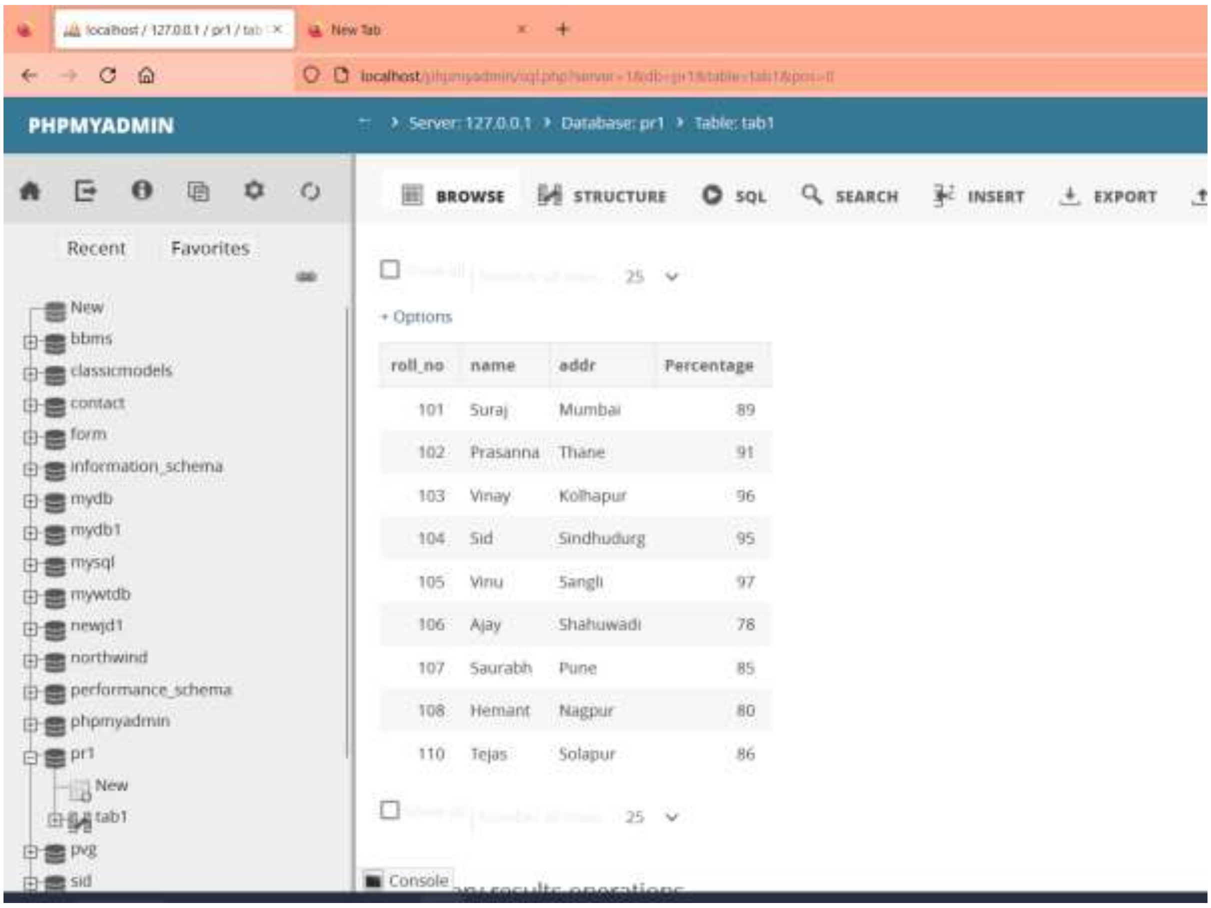
Code:-

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host:"localhost",
user: "root",
password:"",
database:"pr1"
});
con.connect(function(err){
if(err) throw err;
   console.log("Connection made..!");
   var sql = "DELETE FROM tab1 WHERE addr='Bhandara';";
con.query(sql, function(err,result,fields){
       if(err) throw err;
       console.log("Nmber of records deleted: "+result.affectedRows);
   }); });
```

Output:-



```
C:\Users\Rupesh\Desktop\C22096>node deleteQuery.js
Connection made..!
Nmber of records deleted: 1
```



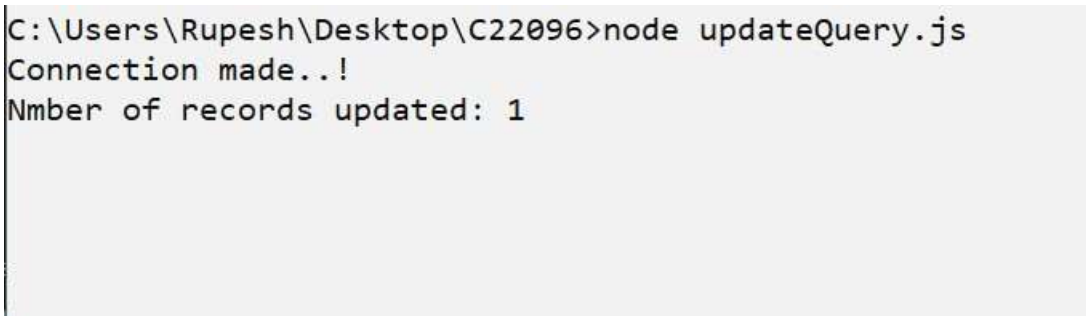
roll_no	name	addr	Percentage
101	Suraj	Mumbai	89
102	Prasanna	Thane	91
103	Vinay	Kolhapur	96
104	Sid	Sindhudurg	95
105	Vinu	Sangli	97
106	Ajay	Shahuwadi	78
107	Saurabh	Pune	85
108	Hemant	Nagpur	80
110	Tejas	Solapur	86

Update Record

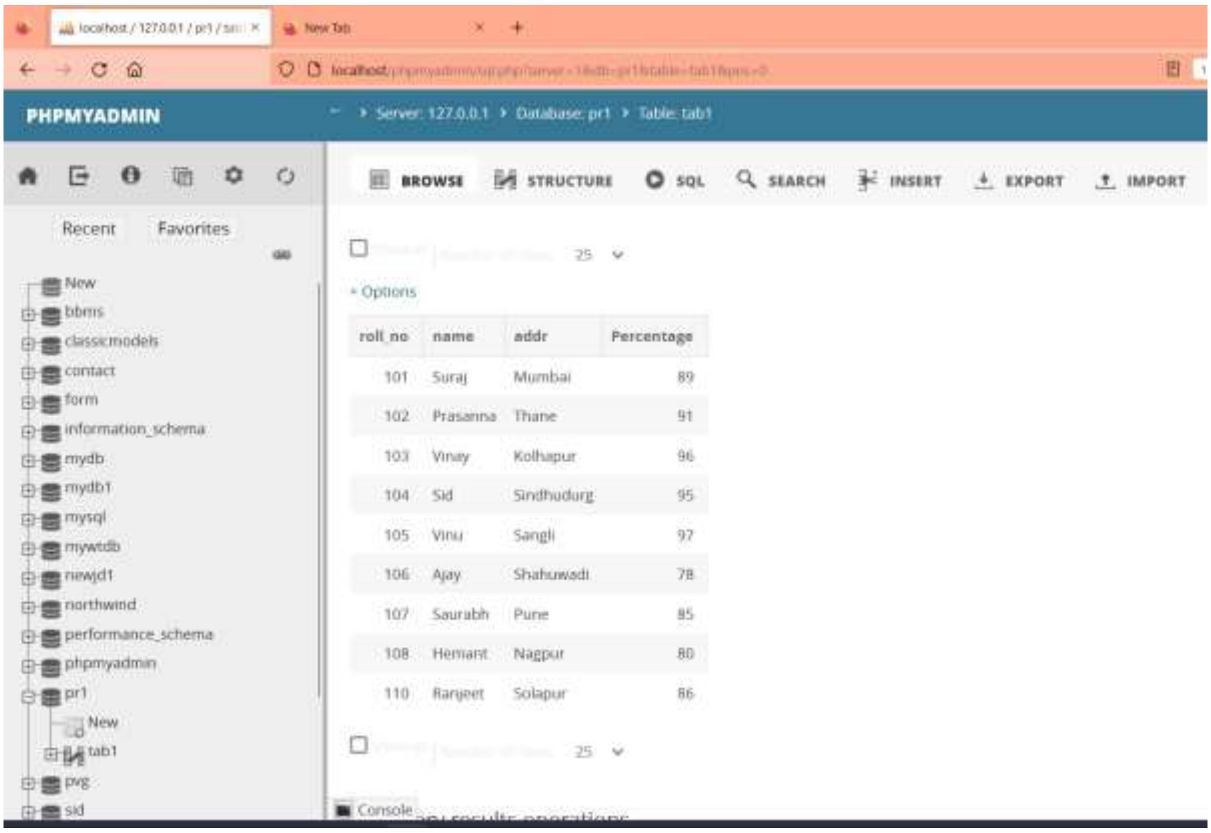
You can update existing records in a table by using the "UPDATE" statement: **Code:-**

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host:"localhost",
user: "root",
password:"",
database:"pr1"
});
con.connect(function(err){
if(err) throw err;
   console.log("Connection made..!");
   var sql = "UPDATE tab1 SET name = 'Ranjeet' WHERE addr='Solapur';";
con.query(sql, function(err,result,fields){        if(err) throw err;
       console.log("Nmber of records updated: "+result.affectedRows);
   }); });
```

Output:-



```
C:\Users\Rupesh\Desktop\C22096>node updateQuery.js
Connection made..!
Nmber of records updated: 1
```



roll_no	name	addr	Percentage
101	Suraj	Mumbai	89
102	Prasanna	Thane	91
103	Vinay	Kolhapur	96
104	Sid	Sindhudurg	95
105	Vinu	Sangli	97
106	Ajay	Shahuwadi	78
107	Saurabh	Pune	85
108	Hemant	Nagpur	80
110	Ranjeet	Solapur	86

PRACTICAL NO.7

Aim: Create an application using filters.

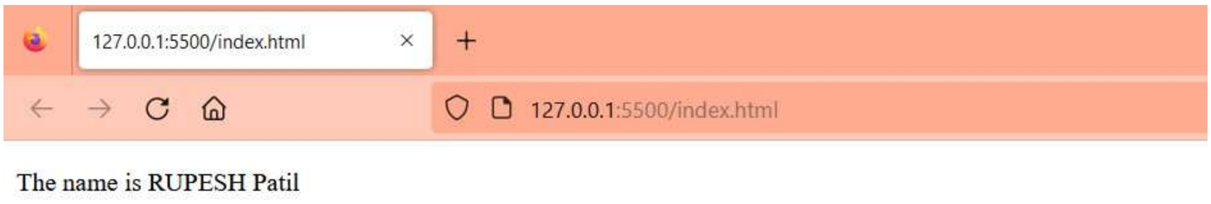
AngularJS provides filters to transform data:

- **currency** Format a number to a currency format.
- **date** Format a date to a specified format.
- **filter** Select a subset of items from an array.
- **json** Format an object to a JSON string.
- **limitTo** Limits an array/string, into a specified number of elements/characters.
- **lowercase** Format a string to lower case.
- **number** Format a number to a string.
- **orderBy** Orders an array by an expression.
- **uppercase** Format a string to upper case.

Code:-

```
<!DOCTYPE html>

<html lang="en">

   <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>

<body>

   <div ng-app = "MyNgApp" ng-controller = "personCtrl">

      <p>The name is {{firstName | uppercase}} {{ lastName}}</p>

   </div>

   <script>

angular.module('MyNgApp',[]).controller('personCtrl',function($scope){

         $scope.firstName = "Anonymous King";

         $scope.lastName = "Patil";

      });

   </script>

</body>

</html>
```

Output:-

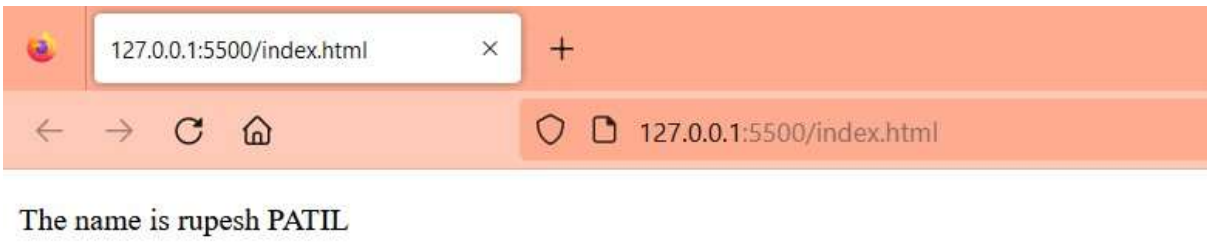

127.0.0.1:5500/index.html

The name is RUPESH Patil

Code:-

```
<!DOCTYPE html>
<html lang="en">
   <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
<body>
     <div ng-app = "MyNgApp" ng-controller = "personCtrl">
        <p>The name is {{firstName | lowercase}} {{lastName | uppercase}}</p>

     </div>
     <script>
        angular.module('MyNgApp',[]).controller('personCtrl',function($scope){
           $scope.firstName = "ANONYMOUS KING";
           $scope.lastName = "PATIL";
        });
     </script>
  </body>
  </html>
```

Output:-

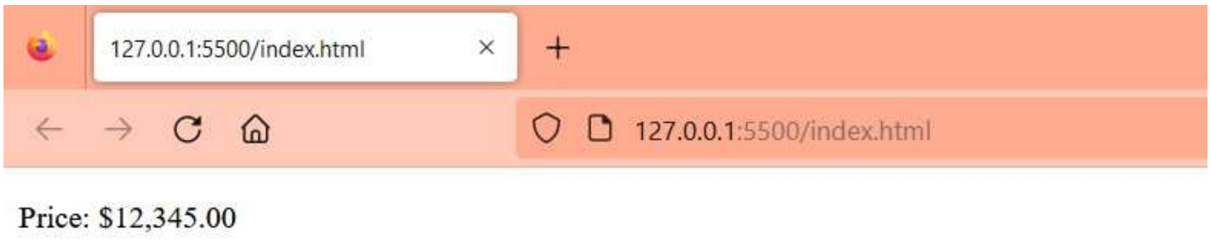


The name is rupesh PATIL

Code:-

```
<!DOCTYPE html>
<html lang="en">
   <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
<body>
     <div ng-app = "MyNgApp" ng-controller="currencyCtrl">
         <p>Price: {{price | currency}}</p>
     </div>
     <script>
         angular.module('MyNgApp',[]).controller('currencyCtrl',function($scope){
$scope.price=12345;
         });
     </script>
  </body>
  </html>
```

Output:-

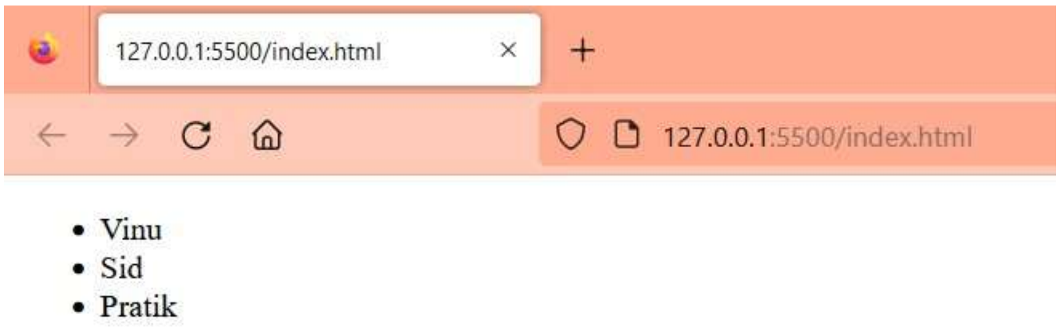


Price: $12,345.00

Code:-

```
<!DOCTYPE html>
<html lang="en">
   <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
<body>
     <div ng-app="MyNgApp" ng-controller="nameCtrl">
        <ul>
           <li ng-repeat="x in names | filter:'i'">
              {{x}}
           </li>
        </ul>
     </div>
     <script>
        angular.module('MyNgApp',[]).controller('nameCtrl',function($scope){
$scope.names=['Vinu','Sid','Prasanna','Pratik','Manas'];
        });
     </script>
  </body>
  </html>
```

Output:-



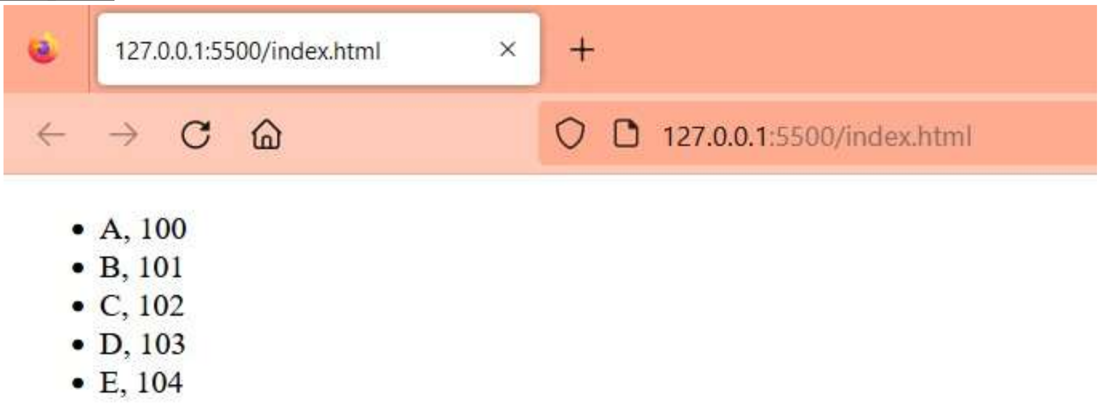
- Vinu
- Sid
- Pratik

Code:-

```
<!DOCTYPE html>
<html lang="en">
   <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
   <!DOCTYPE html>
<html lang="en">
   <script src="ang_js.js"></script>
<body>
   <div ng-app="MyNgApp" ng-controller="orderCtrl">
      <ul>
         <li ng-repeat="x in arrayName | orderBy:'rollno'">
            {{x.name+', '+x.rollno}}
         </li>
      </ul>
   </div>
   <script>
      angular.module('MyNgApp',[]).controller('orderCtrl',function($scope){
         $scope.arrayName=[
         {name:'A',rollno:100},
         {name:'B',rollno:101},
         {name:'C',rollno:102},
         {name:'D',rollno:103},
         {name:'E',rollno:104}
         ];
      });
   </script>
```

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

Output:-



```
• A, 100
• B, 101
• C, 102
• D, 103
• E, 104
```

PRACTICAL NO.8

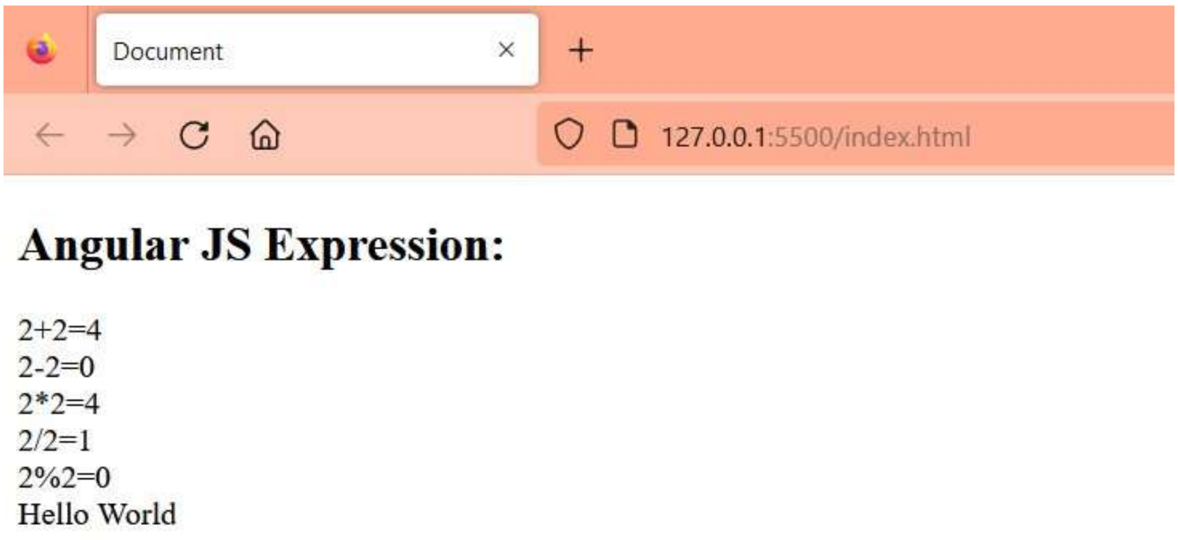
Aim: Create an application to demonstrate directives.

- AngularJS lets you extend HTML with new attributes called **Directives**.
- AngularJS has a set of built-in directives which offers functionality to your applications.
- AngularJS also lets you define your own directives.
- AngularJS directives are extended HTML attributes with the prefix **ng-**.
- The **ng-app** directive initializes an AngularJS application.
- The **ng-init** directive initializes application data.
- The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.

Code:-

```
<!DOCTYPE html>
<html lang=""en>
   <head>
      <meta charset="UTF-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
      <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
   <body>
      <h2>Angular JS Expression:</h2>
      <div ng-app="">
        2+2={{2+2}}<br />
        2-2={{2-2}}<br />
        2*2={{2*2}}<br />
        2/2={{2/2}}<br />
        2%2={{2%2}}<br />
        {{"Hello "+"World"}}<br />
      </div>
   </body>

</html>
```

Output:-



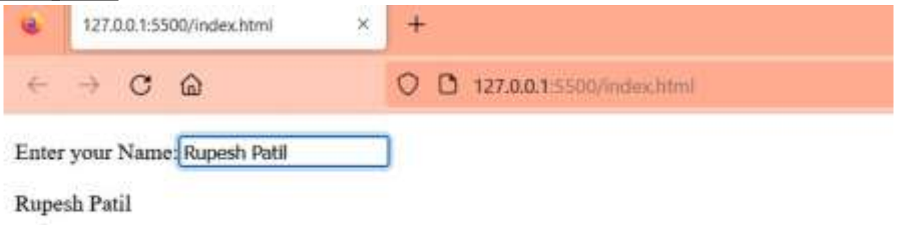
Angular JS Expression:

```
2+2=4
2-2=0
2*2=4
2/2=1
2%2=0
Hello World
```

Code:-

```
<!DOCTYPE html>
<html>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
<body>
    <div ng-app="">
        <p>Enter your Name:<input type="text" ng-model="name"></p>
        <p ng-bind="name"></p>
    </div>
  </body>
</html>
```

Output:-



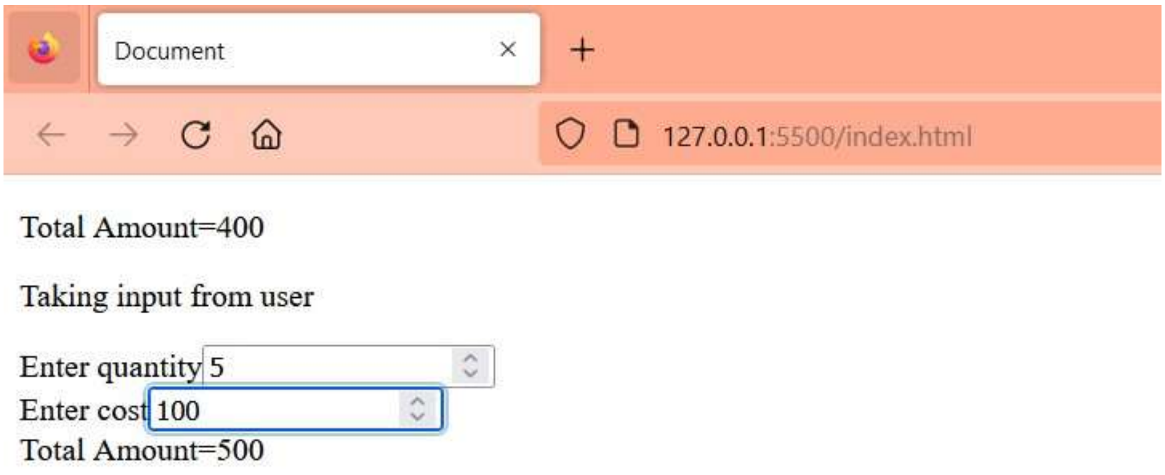
Enter your Name: Rupesh Patil

Rupesh Patil

Code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
   <script src="
https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script> </head>
<body>
   <div ng-app="" ng-init="quantity=2;cost=200">
      <p>Total Amount={{quantity*cost}}</p>
      <p>Taking input from user</p>
      Enter quantity<input type= "number" ng-model="quan"><br>
      Enter cost<input type= "number" ng-model="amt"><br>
      Total Amount={{quan*amt}}
   </div>
</body>
</html>
```

Output:-

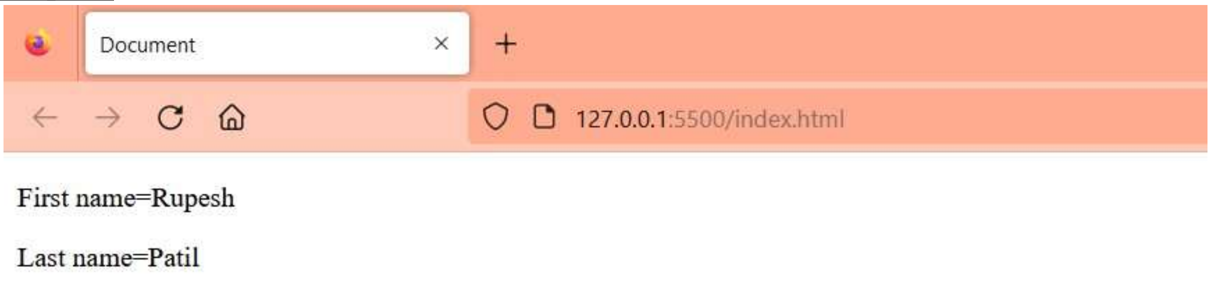


```
Total Amount=400

Taking input from user

Enter quantity 5
Enter cost 100
Total Amount=500
```

Code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
   <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script> </head>
<body>
  <div ng-app="" ng-init="person={fname:'Anonymous King',lname:'Patil'}">
     <p>First name=<span ng-bind='person.fname'></span></p>
     <p>Last name=<span ng-bind='person.lname'></span></p>
  </div>
</body>
</html>
```

Output:-



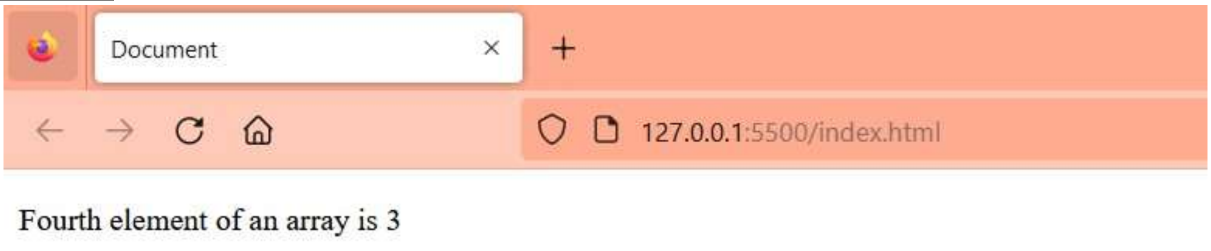
```
First name=Rupesh

Last name=Patil
```

Code:-

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>
  <div ng-app="" ng-init="Points=[1,2,3,4,5]">
      <p>Fourth element of an array is {{Points[2]}}</p>


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

Output:-

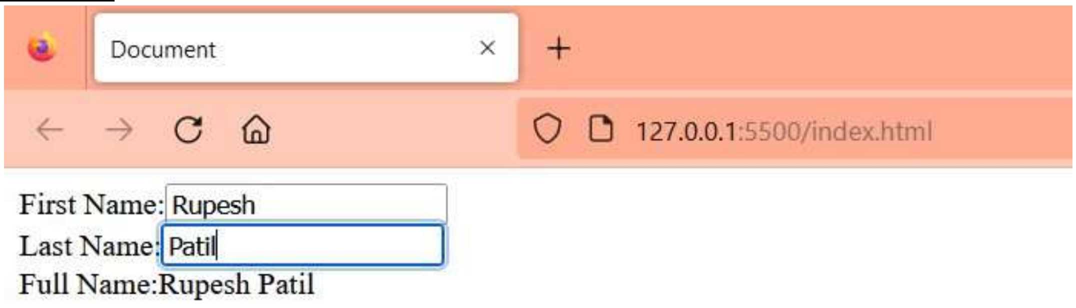


Fourth element of an array is 3

Code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
   <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script> </head>
<body>
   <div ng-app="myApp" ng-controller="myCtrl">
      First Name:<input type="text" ng-model="firstName"><br>
      Last Name:<input type="text" ng-model="lastName"><br>
      Full Name:{{firstName+' '+lastName}}
   </div>
   <script>        var
app=angular.module('myApp',[]);
app.controller('myCtrl',function($scope){
         $scope.firstName="Hello";
         $scope.lastName="World";
      });
   </script>
</body>
</html>
```

Output:-



```
First Name: Rupesh
Last Name: Patil
Full Name:Rupesh Patil
```

PRACTICAL NO.9

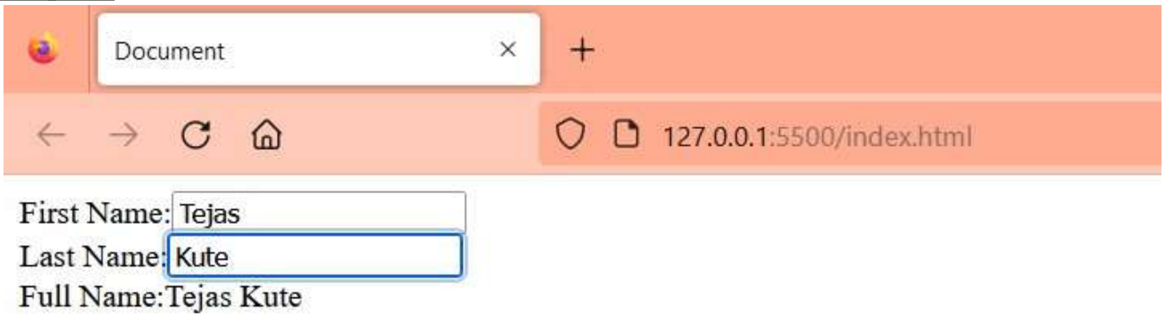
Aim: Demonstrate controllers in Angular.js through an application.

- AngularJS controllers control the data of AngularJS applications.
- AngularJS controllers are regular JavaScript Objects.
- AngularJS applications are controlled by controllers.
- The **ng-controller** directive defines the application controller.
- A controller is a JavaScript Object, created by a standard JavaScript object constructor.

Code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
   <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script> </head>
<body>
   <div ng-app="myApp" ng-controller="myCtrl">
      First Name:<input type="text" ng-model="firstName"><br>
      Last Name:<input type="text" ng-model="lastName"><br>
      Full Name:{{fullName()}}
   </div>
   <script>        var
app=angular.module('myApp',[]);
app.controller('myCtrl',function($scope){
        $scope.firstName="Hello";
        $scope.lastName="World";
$scope.fullName=function(){
         return $scope.firstName+" "+$scope.lastName;
        }
      });
   </script>
</body>
</html>
```

Output:-

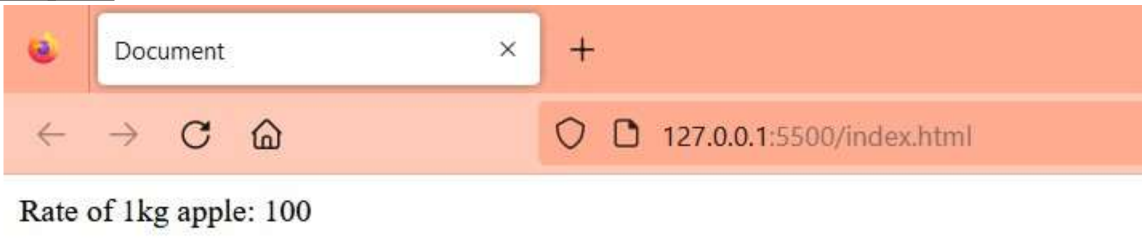


First Name: Tejas
Last Name: Kute
Full Name:Tejas Kute

Code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
   <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script> </head>
<body ng-app="myApp" >
  <div ng-controller="myCtrl">
     {{message}}
     {{price}}
  </div>   <script>      var
ngapp=angular.module('myApp',[]);
ngapp.controller('myCtrl',function($scope){
         $scope.message="Rate of 1kg apple: ";
         $scope.price=100;
     });
  </script>
</body>
</html>
```

Output:-



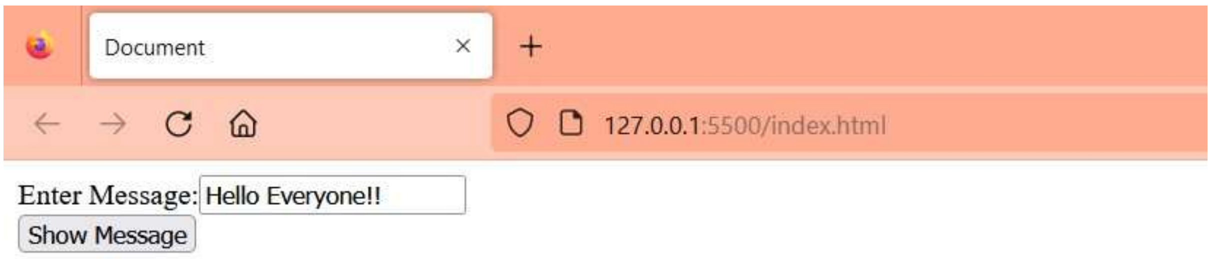
Rate of 1kg apple: 100

Code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
   <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script> </head>
<body ng-app="myApp" >
   <div ng-controller="myCtrl">
      Enter Message:<input type="text" ng-model="message" />
      <br />
      <button ng-click="showMsg(message)">Show Message</button>
</div>
   <script>       var
ngapp=angular.module('myApp',[]);
ngapp.controller('myCtrl',function($scope){
         $scope.message="Hello Everyone!!";
         $scope.showMsg=function(msg){
            alert(msg);
         };
      });
   </script>
```

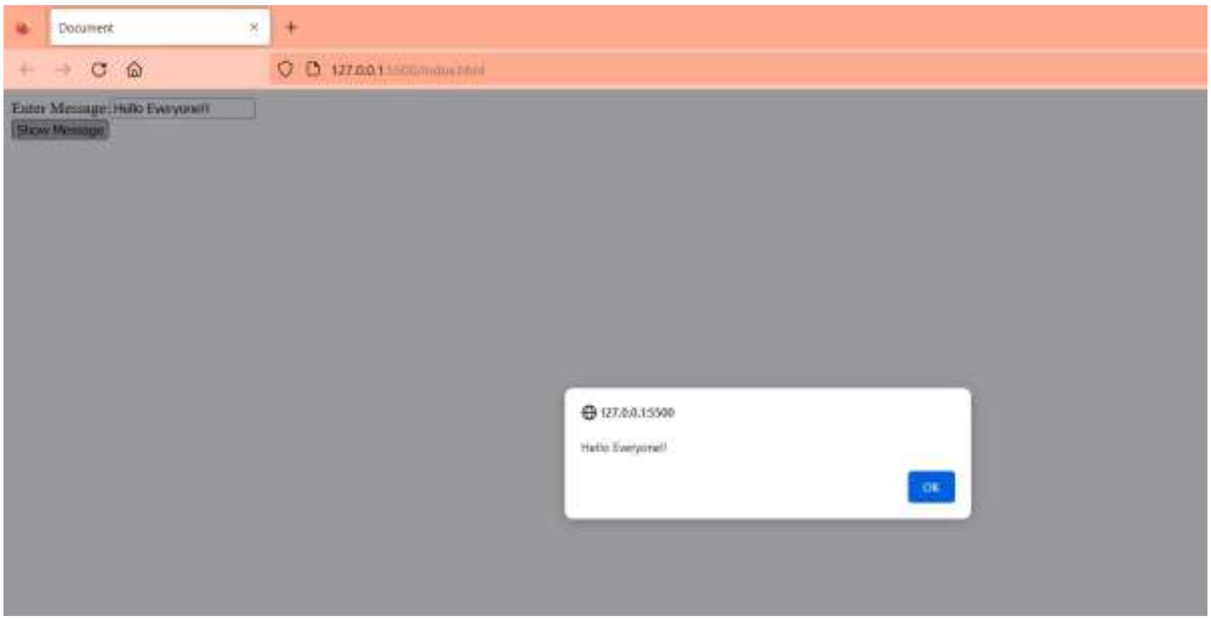
```
</body>
</html>
```

Output:-



127.0.0.1:5500/index.html

Enter Message: Hello Everyone!!



127.0.0.1:5500/index.html

Enter Message: Hello Everyone!!

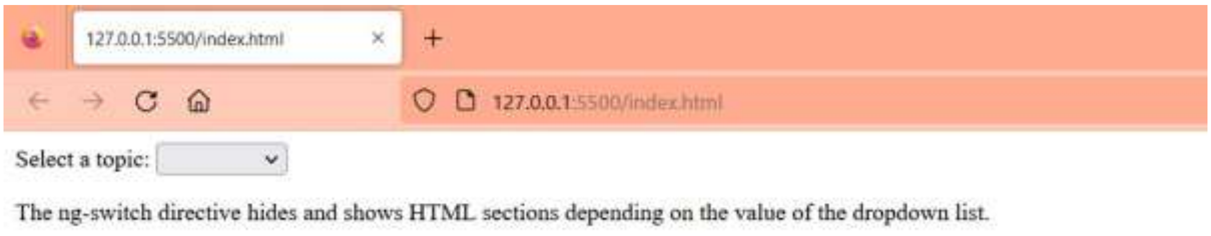
127.0.0.1:5500

Hello Everyone!!

PRACTICAL NO.10

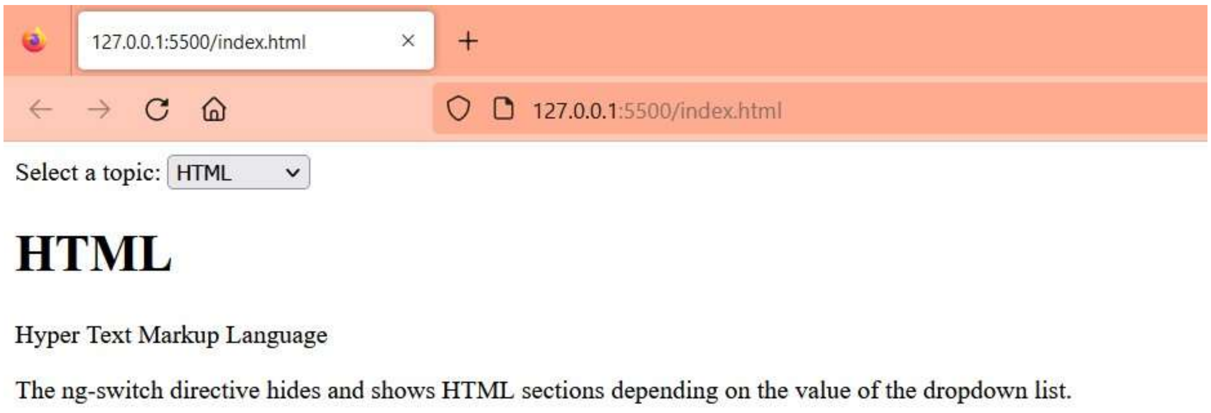
Aim: Demonstrate features of Angular.js forms with a program.

```
<!DOCTYPE html>
<html lang="en">
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
<body ng-app="">
  <form>
    Select a topic:
    <select ng-model="myVar">
      <option value ="">
      <option value ="HTML">HTML
      <option value ="Node js">Node js
      <option value ="Angular js">Angular js
    </select>
  </form>
  <div ng-switch="myVar">
    <div ng-switch-when="HTML">
      <h1>HTML</h1>
      <p>Hyper Text Markup Language</p>
    </div>
    <div ng-switch-when="Node js">
      <h1>Node js</h1>
      <p>Node js is asynchronous programming language</p>
</div>
    <div ng-switch-when="Angular js">
      <h1>Angular js</h1>
      <p>Welcome to Angular js</p>
    </div>
  </div>
  <p>The ng-switch directive hides and shows HTML sections depending on the value of the
dropdown list.</p>
</body>
</html>
```


127.0.0.1:5500/index.html

Select a topic:

The ng-switch directive hides and shows HTML sections depending on the value of the dropdown list.

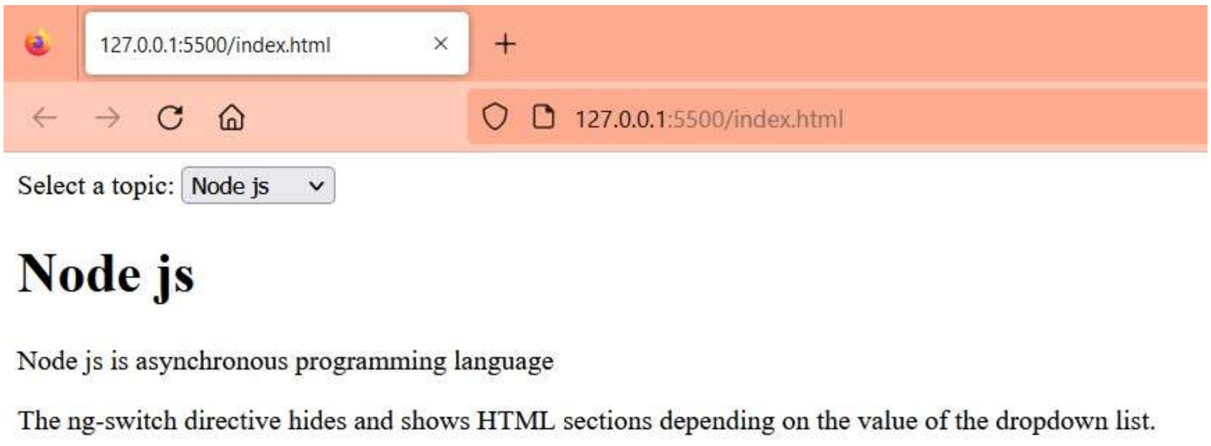

127.0.0.1:5500/index.html

Select a topic: HTML

HTML

Hyper Text Markup Language

The ng-switch directive hides and shows HTML sections depending on the value of the dropdown list.

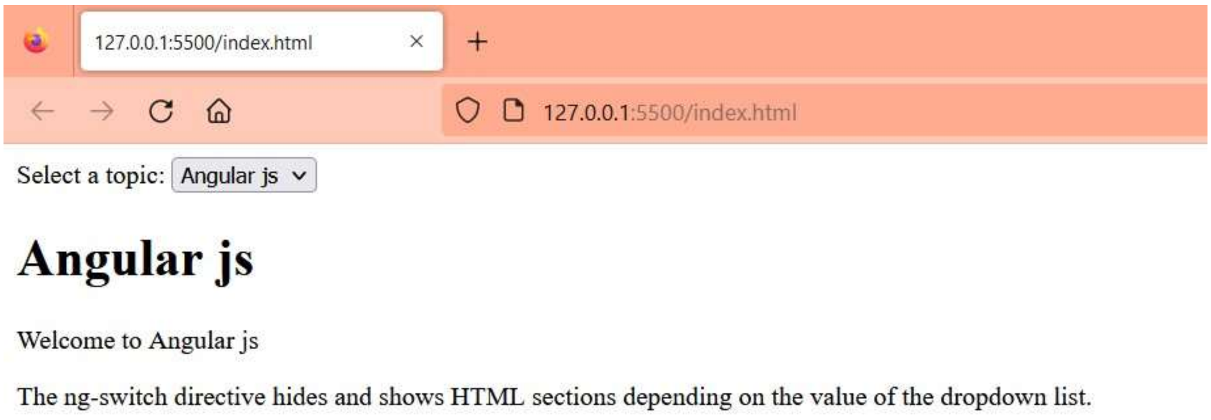

127.0.0.1:5500/index.html

Select a topic: Node js

Node js

Node js is asynchronous programming language

The ng-switch directive hides and shows HTML sections depending on the value of the dropdown list.



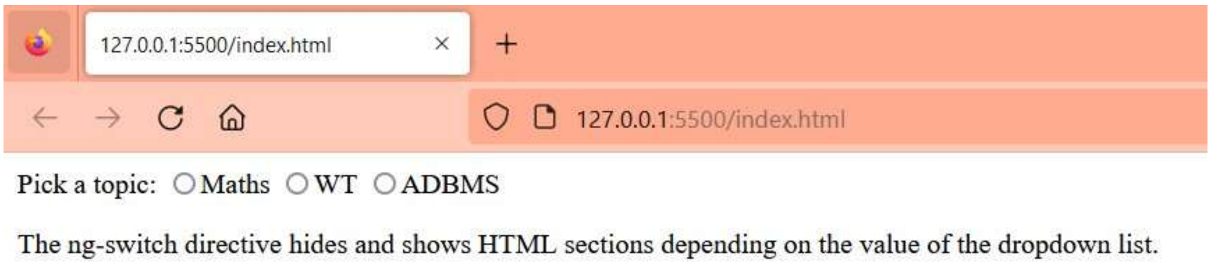
Select a topic: Angular js

Angular js

Welcome to Angular js

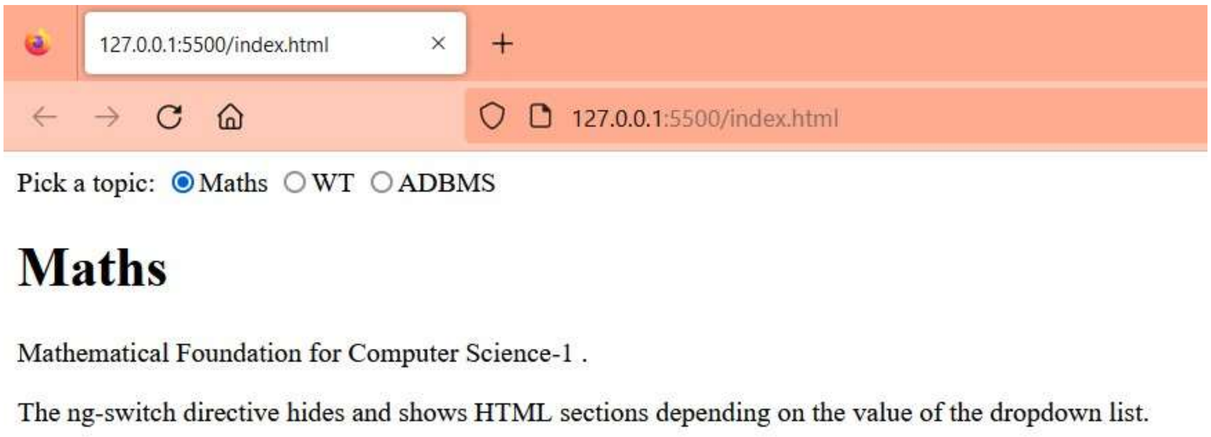
The ng-switch directive hides and shows HTML sections depending on the value of the dropdown list.

```
<!DOCTYPE html>
<html lang="en">
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
<body ng-app="">
  <form>
    Pick a topic:
    <input type="radio" ng-model="myVar" value="Maths">Maths
    <input type="radio" ng-model="myVar" value="WT">WT
    <input type="radio" ng-model="myVar" value="ADBMS">ADBMS
</form>
  <div ng-switch="myVar">
    <div ng-switch-when="Maths">
      <h1>Maths</h1>
      <p>Mathematical Foundation for Computer Science-1 .</p>
</div>
    <div ng-switch-when="WT">
      <h1>Web Technology</h1>
      <p>Web Technology contains Nodejs and Angularjs.</p>
</div>
    <div ng-switch-when="ADBMS">
      <h1>ADBMS</h1>
      <p>Advanced Database Management System.</p>
    </div>
  </div>
  <p>The ng-switch directive hides and shows HTML sections depending on the value of the
dropdown list.</p>
</body>
</html>
```



Pick a topic: ○ Maths ○ WT ○ ADBMS

The ng-switch directive hides and shows HTML sections depending on the value of the dropdown list.

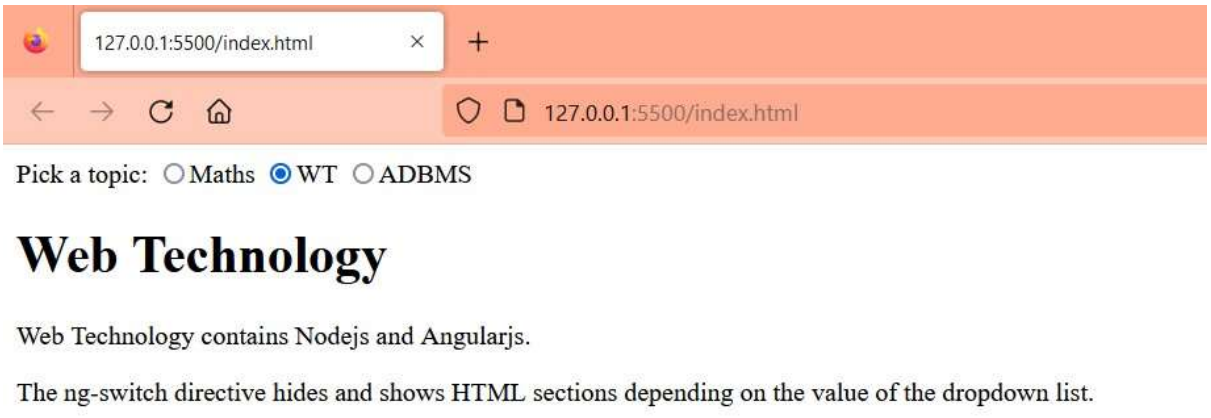


Pick a topic: [✓ Maths] ○ WT ○ ADBMS

Maths

Mathematical Foundation for Computer Science-1 .

The ng-switch directive hides and shows HTML sections depending on the value of the dropdown list.

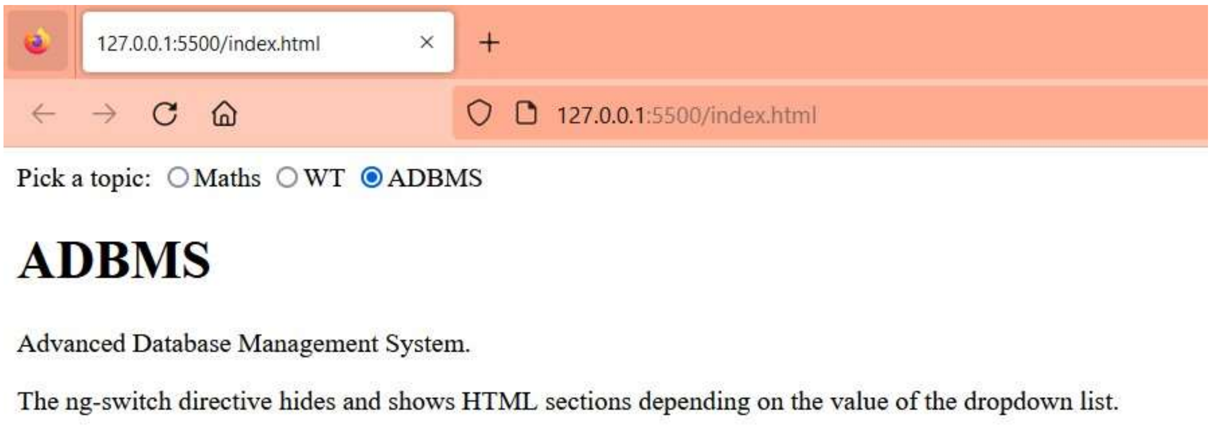


Pick a topic: ○ Maths [✓ WT] ○ ADBMS

Web Technology

Web Technology contains Nodejs and Angularjs.

The ng-switch directive hides and shows HTML sections depending on the value of the dropdown list.



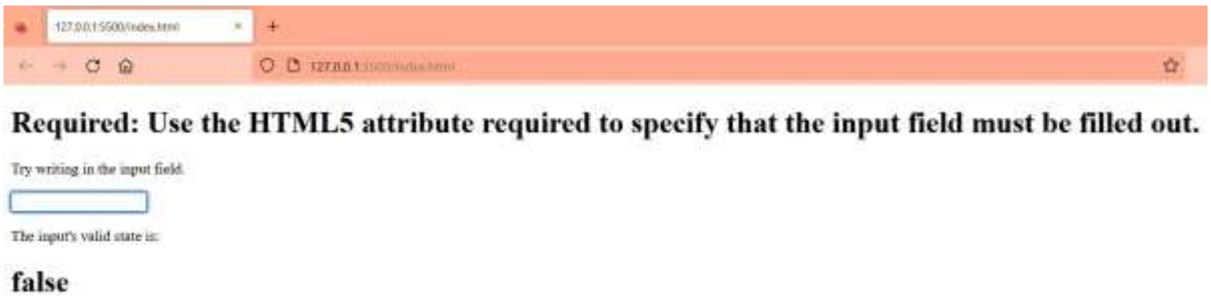
Pick a topic: ○ Maths ○ WT [✓ ADBMS]

ADBMS

Advanced Database Management System.

The ng-switch directive hides and shows HTML sections depending on the value of the dropdown list.

```
<!DOCTYPE html>
<html lang="en"> <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.
js"></script>
<body ng-app="">
  <h1>Required: Use the HTML5 attribute required to specify that the
input field must be filled out.</h1>   <p>Try writing in the input
field.</p>

  <form name="myForm">
      <input name="myInput" ng-model="myInput" required>
  </form>

  <p>The input's valid state is:</p>
  <h1>{{myForm.myInput.$valid}}</h1>
</body>
</html>
```

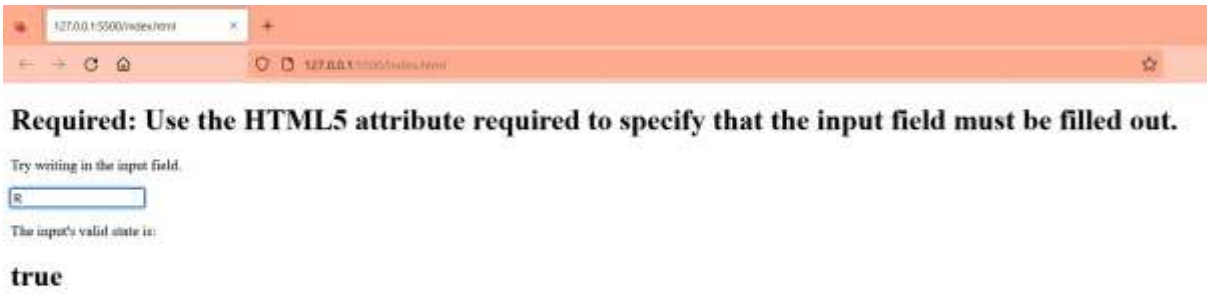


Required: Use the HTML5 attribute required to specify that the input field must be filled out.

Try writing in the input field.

The input's valid state is:

false


127.0.0.1:5500/index.html

Required: Use the HTML5 attribute required to specify that the input field must be filled out.

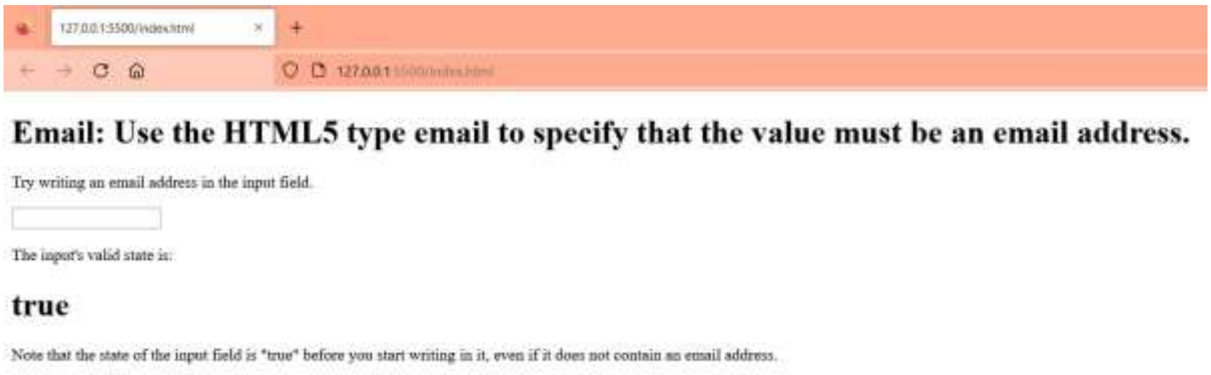
Try writing in the input field.

R

The input's valid state is:

true

```
<!DOCTYPE html>
<html lang="en"> <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.
js"></script>
<body ng-app="">
  <h1>Email: Use the HTML5 type email to specify that the value must be
an email address.</h1>
  <p>Try writing an email address in the input field.</p>

  <form name="myForm">
      <input type="email" name="myInput" ng-model="myInput">
  </form>

  <p>The input's valid state is:</p>
  <h1>{{myForm.myInput.$valid}}</h1>
  <p>Note that the state of the input field is "true" before you start
writing in it, even if it does not contain an email address.</p>
</body>
</html>
```


127.0.0.1:5500/index.html

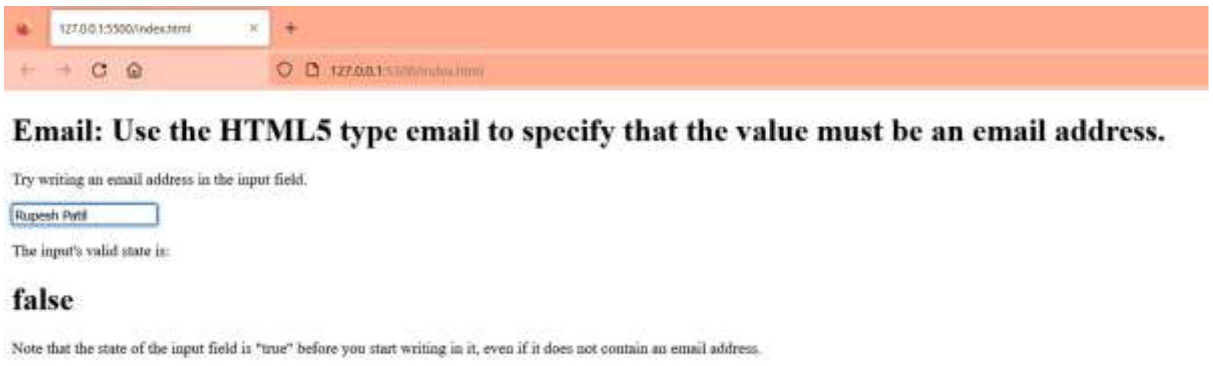
Email: Use the HTML5 type email to specify that the value must be an email address.

Try writing an email address in the input field.

The input's valid state is:

true

Note that the state of the input field is "true" before you start writing in it, even if it does not contain an email address.



Email: Use the HTML5 type email to specify that the value must be an email address.

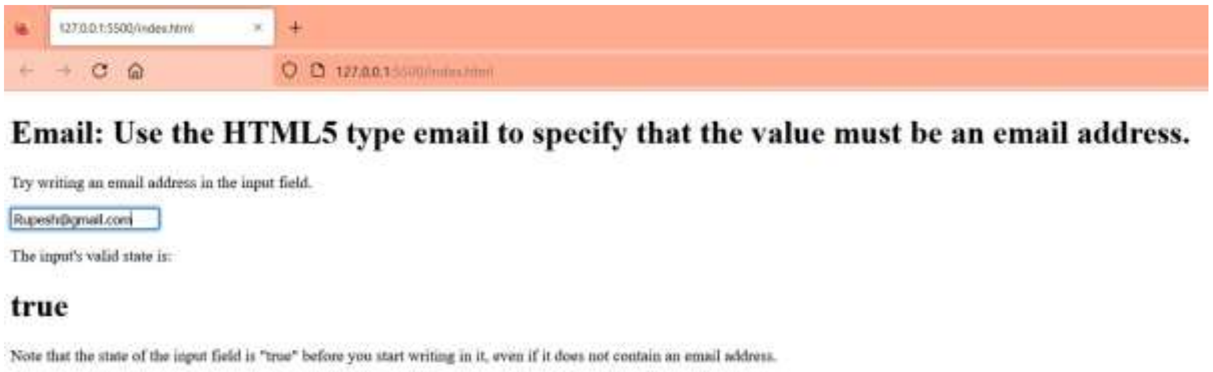
Try writing an email address in the input field.

Rupesh Patil

The input's valid state is:

false

Note that the state of the input field is "true" before you start writing in it, even if it does not contain an email address.



Email: Use the HTML5 type email to specify that the value must be an email address.

Try writing an email address in the input field.

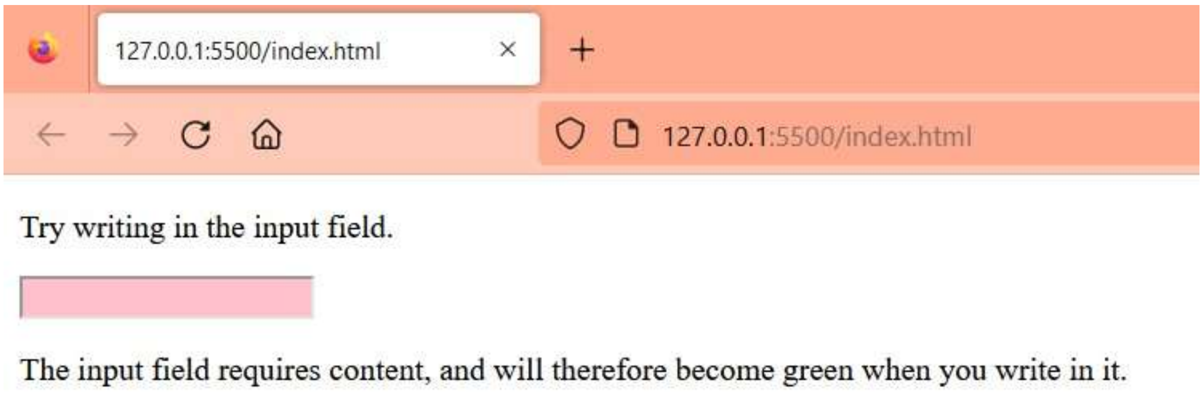
Rupesh@gmail.com

The input's valid state is:

true

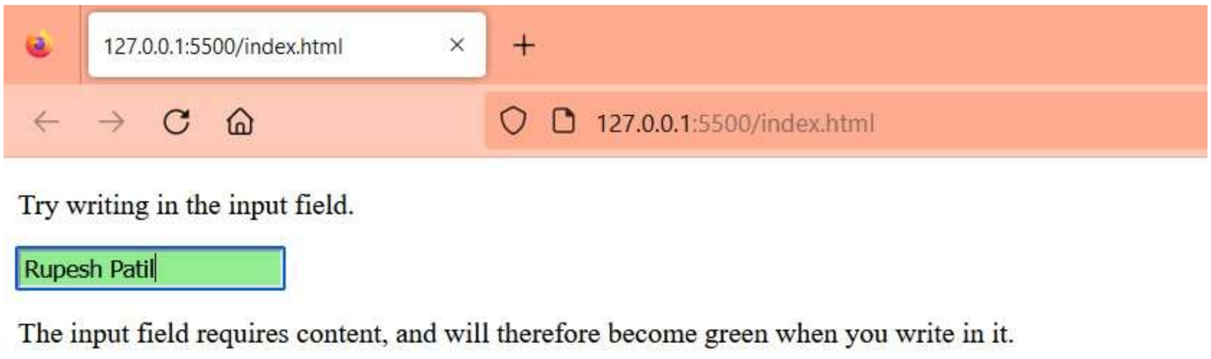
Note that the state of the input field is "true" before you start writing in it, even if it does not contain an email address.

```
<!DOCTYPE html>
<html lang="en">
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
<style>   input.ng-invalid{
background-color: pink;
  }
  input.ng-valid{
      background-color: lightgreen;
  }
</style>

<body ng-app="">

 <p>Try writing in the input field.</p>

 <form name="myForm">
     <input name="myName" ng-model="myName" required>
 </form>

 <p>The input field requires content, and will therefore become green when you write in it.</p>
 <h1>{{myForm.myInput.$valid}}</h1>
</body>
</html>
```



Try writing in the input field.

The input field requires content, and will therefore become green when you write in it.



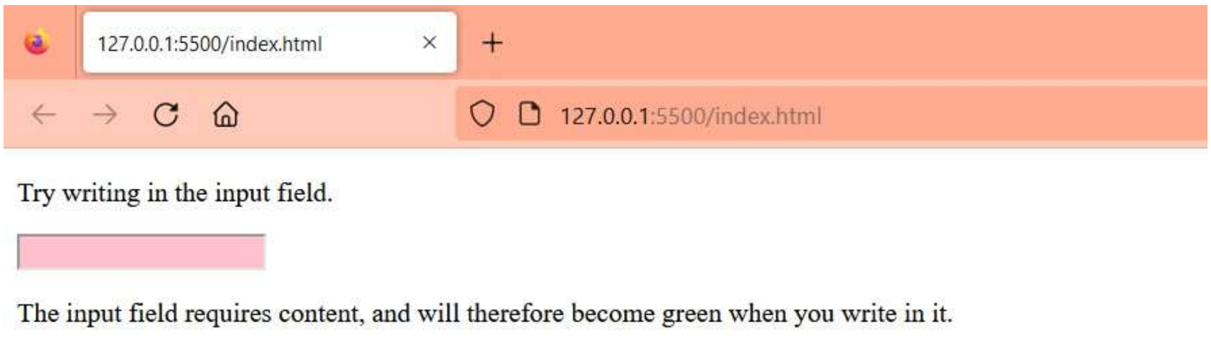
Try writing in the input field.

Rupesh Patil

The input field requires content, and will therefore become green when you write in it.

```
<!DOCTYPE html>
<html lang="en">
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
<style>   input.ng-pristine{
background-color: pink;
  }
 input.ng-dirty{
     background-color: lightgreen;
  }
</style>
```

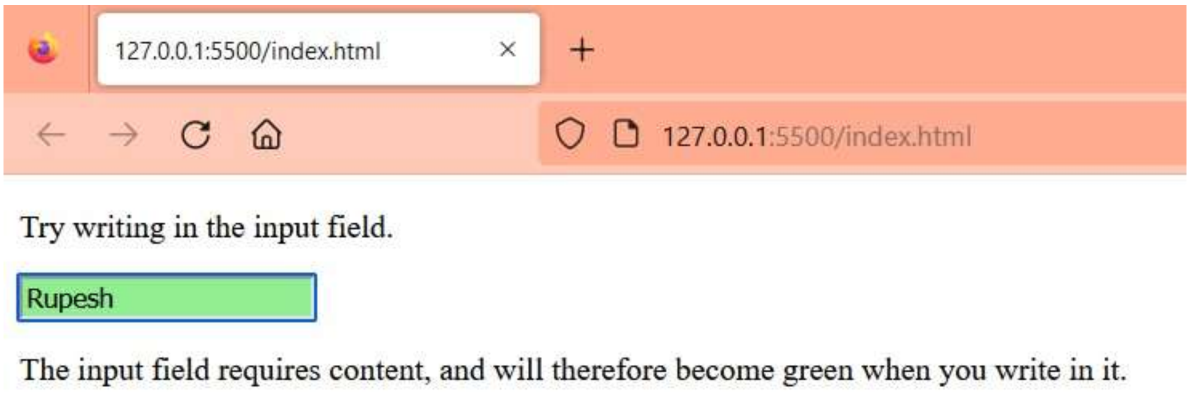
```
<body ng-app="">

 <p>Try writing in the input field.</p>

 <form name="myForm">
    <input name="myName" ng-model="myName" required>
 </form>

 <p>The input field requires content, and will therefore become green when you write in
it.</p> </body>
</html>
```



127.0.0.1:5500/index.html

Try writing in the input field.

The input field requires content, and will therefore become green when you write in it.



127.0.0.1:5500/index.html

Try writing in the input field.

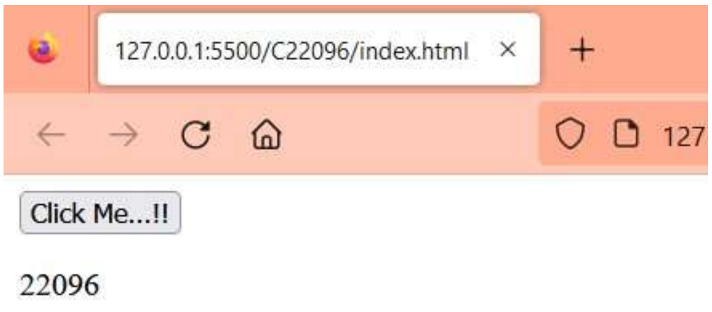
Rupesh

The input field requires content, and will therefore become green when you write in it.

Mouseclick:

```
<!DOCTYPE html>
<html lang="en">
 <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>

<body>
    <div ng-app="myApp" ng-controller="myCtrl">
        <button ng-click="count=count+1">Click Me...!!</button>
        <p>{{count}}</p>
    </div>
    <script>         var
app=angular.module('myApp',[]);
app.controller('myCtrl',function($scope){
            $scope.count=0;
        });
    </script>
</body>
</html>
```

Output:



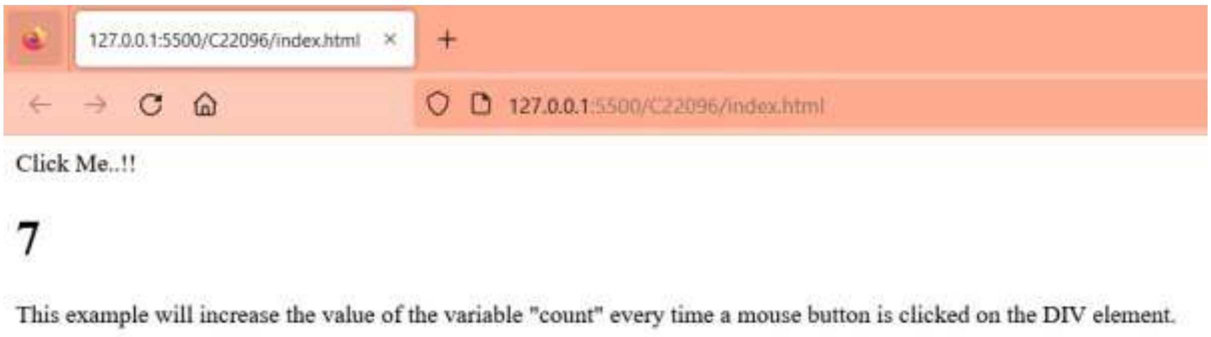
```
22096
```

MouseDown:

```
<!DOCTYPE html>
<html lang="en">
 <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
```

```
<body ng-app="">
   <div  ng-mousedown="count=count+1" ng-init="count=0">Click Me..!!</div>
   <h1>{{count}}</h1>
   <p>This example will increase the value of the variable "count" every time a mouse button is
clicked on the DIV element.</p>
</body>
</html>
```

Output:

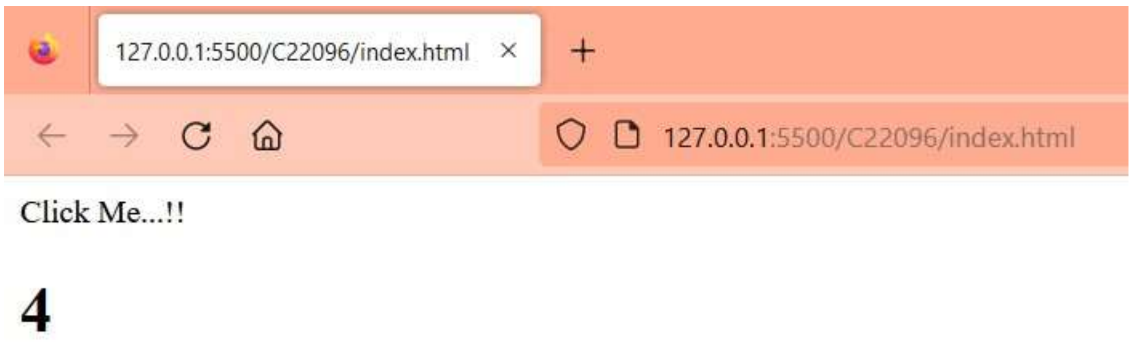


```
Click Me..!!

7

This example will increase the value of the variable "count" every time a mouse button is clicked on the DIV element.
```

MouseEnter:

```
<!DOCTYPE html>
<html lang="en">
 <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>

 <body ng-app="">
   <div ng-mouseenter="count=count+1">Click Me...!!
      <h1>{{count}}</h1>
   </div>
</body> </html>
```

Output:

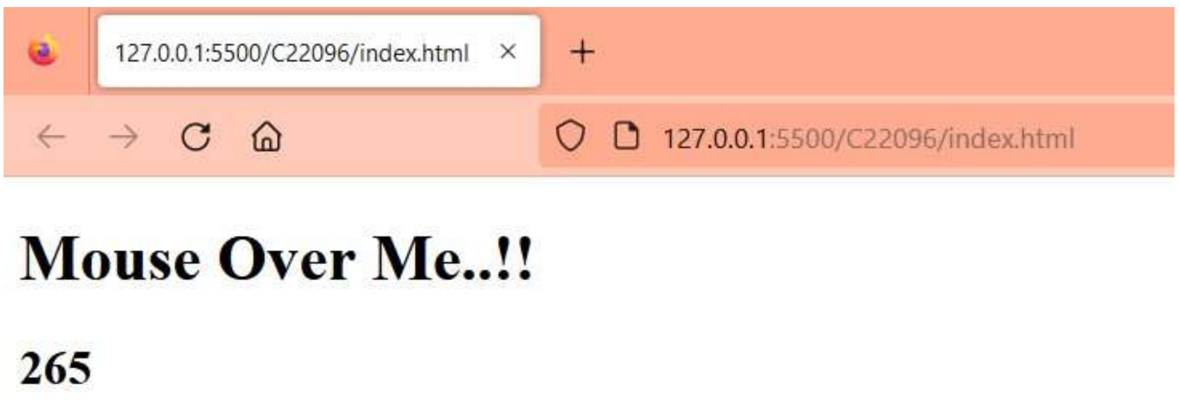

Click Me...!!

4

MouseMove:

```
<!DOCTYPE html>
<html lang="en">
 <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>

 <body>
    <div ng-app="myApp" ng-controller="myCtrl">
        <h1 ng-mousemove="count=count+1">Mouse Over Me..!!</h1>
        <h2>{{count}}</h2>
    </div>
    <script>         var
app=angular.module('myApp',[]);
app.controller('myCtrl',function($scope){
            $scope.count=0;
        });
    </script>
</body>
</html>
```

Output:


Mouse Over Me..!!

265

PRACTICAL NO.11

Aim: Create a SPA (Single Page Application).

- A single-page application (SPA) is a web application or website that interacts with the user by dynamically rewriting the current web page with new data from the web server, instead of the default method of the browser loading entire new pages.
- The goal is faster transitions that make the website feel more like a native app.
- Web browser uses Javascript frameworks and libraries, such as AngularJS, Ember.js, ExtJS, Knockout.js, Meteor.js, React, Vue.js and Svelte have adopted SPA principles.
- The **ngRoute** module helps your application to become a Single Page Application.
- The **ngRoute** module routes your application to different pages without reloading the entire application. **Code: spa.html**

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular-route.js"></script>
<body ng-app="myApp">
<p><a href="#/!">Main</a></p>
<a href="#!kolhapur">Kolhapur</a>
<a href="#!sangli">Sangli</a>
<a href="#!satara">Satara</a>
<div ng-view></div>
<script>
var app = angular.module("myApp", ["ngRoute"]);  app.config(function($routeProvider)
{
   $routeProvider
.when("/", {
      templateUrl : "main.html"
   })
   .when("/satara", {
      templateUrl : "satara.html"
```

```
    })
    .when("/kolhapur", {
        templateUrl : "kolhapur.html"
    })
    .when("/sangli", {
        templateUrl : "sangli.html"
    });
});
</script>
<p>Click on the links to navigate to "kolhapur.html", "satara.html", "sangli.html", or back to
"main.html"</p>
</body>  </html>
```

main.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
</head>
<body>
<h1>Welcome Page</h1>
</body>
</html>
```

kolhapur.html <!DOCTYPE

```
html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
p1
      {
      background-color: antiquewhite;
      display: block;
margin-top: 10px;
width: 800px;       font-size:
20px;
      }       img{
width: 300px;
height: 200px;
      }
    </style>
</head>
<body >
  <center>
  <h1>कोल्हापरू </h1>
<img src="kolhapur.jpg"/>
```

<p1>कोल्हापरू हे महाराष्ट्राच्या दक्षिण भागातील मोठे शहर आहे. येथील मख्यु भाषा मराठी आहे. येथील श्री करवीर निवासिि ी आई अंबाबाईचे मंददर हे महाराष्ट्रातील देवीच्या िाडेतीि शक्ततपीठांपैकी एक आहे. पंचगंगा इथली प्रमखु िदी आहे. शहराच्या आिापाि पन्हाळा, गगिबावडा, िसिहं ृ वाडी, खखद्रापरू , ववशाळगड, राधाि गरी, दाजीपरू अभयारण्य शाहूवाडी इत्यादी दठकाणे आहेत. [शाहू महाराज छत्रपती शाहूमहाराजांच्या] काळात म्हणजेच १८७४ ते १९२२ मध्ये शहराचा मोठा ववकाि झाला. कोल्हापरू हे प्रसिद्ध िेत्र आहे.कोल्हापरू आपल्या रंगबेरंगी पोशाख आखण कोल्हापरीु चप्पल यांिोबतच आणखीि काही गोष्ट्ीं िाठी प्रसिद्ध आहे.
येथील खाण्यावपण्यापािि ू फिरण्यापयतं च्या िवचव गोष्ट्ी येथे खाि आहेत. येथे येणाऱ्या पय्व कांचे येथील िंस्कृतीची िंदरताु पाहूि मिच भरत िाही.
महाराष्ट्रातील या िंदरु शहरामध्ये अशा बऱ्याच आकषकव गोष्ट्ी आहेत.</p1>

<h3>खाद्यिंस्कृती</h3>

<p1>कोल्हापरमध्येू समळणारे प्रसिद्ध आखण रुचकर खाद्य पदाथव म्हणजे कोल्हापरचीू समिळ. महाराष्ट्रातील ही प्रसिद्ध समिळ कोल्हापूरमध्ये अिेक दठकाणी वेगवेगळ्या प्रकारे समळते. उदा० बावडा समिळ, िडतरे समिळ, चोरगे समिळ, हॉ ेल िाकोली समिळ, खाि बाग समिळ वगैरे' मांिाहारी खाद्य पदाथांच्या िंदभावत कोल्हापरातलाु तांबडा रस्िा (म्णाचे िपू), पांढरा रस्िा (म्णाचे िूप ितत पांढऱ्या रंगाचे),म्णाचे लोणचे, खखमा राईि बॉल्ि प्रसिद्ध आहेत.

कोल्हापरातल्याु खाऊ गल्लीत अिलेली राजाभाऊंची भेळ प्रसिद्ध आहे. समरजकर नतक्ी, गंगावेि येथे दधू कट्टट्टयावर उत्तम प्रकारचे ताजे दधू समळते. इथले महालक्ष्मी अंबाबाईचे मंददर हे देवीच्या िाडेतीि शक्ततपीठांपकै ी एक आहे. पंचगंगा इथली प्रमखु िदी. शहराच्या अिापाि पन्हाळा, गगिबावडा, िसिहंृ वाडी, खखद्रापरू , ववशाळगड, राधािगरी, दाजीपरू अभयारण्य आदी भे् देण्यािारखी दठकाणे आहेत. कोल्हपरचीू लवंगी समरची प्रसिद्ध आहे. कोल्हापरू क्जल्हा (जिाु) शाहूमहाराजांचा क्जल्हा म्हणिहीू िवपव ररचचत आहे.

एका आख्यानयकेििािरु , प्राचीि काळी केशी रािािचा मलगाु कोल्हािरु हा इथे राज्य करत होता. त्यािं िगळ्यांिा त्रस्त करूि िोडले होते म्हणिू देवांच्या ववितंीवरूि महालक्ष्मीिे त्याच्याशी यद्धु केले. हे यद्धु िऊ ददवि चालले होते. आक्ववि शद्धु पंचमीला महालक्ष्मीिे कोल्हािराचाु वध केला. कोल्हािरु महालक्ष्मीला शरण गेला आखण त्यािे नतच्याकडे, आपल्या िगराची कोल्हापरू व करवीर ही जी िावे आहेत ती तशीच पढेहीु राहावीत, अिा वर माचगतला. त्यािििािरु या िगरीला कोल्हापरू वा करवीर या िावािे ओळखले जाते.

कोल्हापरचाू वपवळाधमक गळू आखण मिाले कुणारे डखं हेही कोल्हापरचेू वैसशष्ट्ट्टय. िध्या भारतातल्या बहुतके हॉ्ेलांत व्हेज कोल्हापुरी, कोल्हापरीु चचकि हे पदाथव उपलब्ध अितात; पण ‘व्हेज कोल्हापरीु ’ म्हणजे तांबडा रस्िा व पांढरा रस्िा एकेक वा्ी घेऊि त्याबरोबर चचकि फकंवा म्ण खायचे. म्हणजेच तांबडा रस्िा नतख् लागला तर त्यात पांढरा रस्िा समिळूि आपल्याला योग्य अशा चवीचा रस्िा खाता येतो. हा रस्िा बाकी रववयांप्रमाणे फकंवा ग्रेव्हीप्रमाणे द्काऊ िाही. तांबडा रस्िा तयार करतािा कोल्हापरीु च्णी वापरावी लागते.

कोल्हापरचाू अजिू एक प्रकार म्हणजे दावणचगरी लोणी डोिा . हा डोिा ‘ बऱ्याच दठकाणी समळतो. भरपरू लोणी आखण जाळीदारपणा ही त्याची खासियत. कोल्हापरातु मांिाहारी, शाकाहारी स्वयंपाकाची घराघरातली पद्धत जवळपाि िारखीच अिते. पांढरा तांबडा रस्िा, िकेु म्ण, खखमा हे प्रकार आख्ख्या कोल्हापरातु होत अितात.

त्याचबरोबर नतथल्या पोळ्यांचीही (चपात्या) एक खासियत आहे. तीि पदर ि्लेलीु व खरपू तेल लावू भाजलेली ही गरम गरम चपाती अप्रनतम लागते. इथली आणखी एक खासियत म्हणजे 'पोकळा' िवाची पालेभाजी. तव्यावर केलेली ही भाजी, भाकरी आखण खडाव यांिा तोड िाही. याबरोबरच दधकट्टट्टयावरू समळणारे आखण ग्राहकािमोरच काढण्यात येणारे म्हशीचे धारोष्ट्ण दधू . थे् पेल्यातच धार काढायची व पेला तोंडाला लावायचा! आणखी एक अिलातू प्रकार म्हणजे, ताज्या दधातु िोडा घालू प्यायची पद्धत अशा पद्धतीिे दधाचे निजतंकीकरणु होते.

कोल्हापरचू महालक्ष्मी मंददर महाराष्ट्रातील िाडेतीि शक्ततपीठांपैकी एक आहे. प्राचीि भारतातील ववसभन्ि पराणांिीु क्जथे शततीची देवी प्रक् झाली आहे अशा१०८ शक्ततपीठांची िचीू तयार केली आहे. यामध्ये जेथे आजचे वतमव ाि कोल्हापरू शहर आहे, त्या करवीर
िेत्राच्या महालक्ष्मीचे ववशेष महत्त्व आहे.श्री महालक्ष्मी मंददर कोल्हापरच्याू िहा शततीपैकी एक आहे फक क्जथे कोणतीही इच्छा परीु होण्याबरोबर मततीहीु समळू शकते.यािाठी या स्थळाला काशीच्या तलितु अचधक महत्त्व प्राप्त होते

िेत्र िेहमीच धन्य आहे अिे मािले जाते कारण महालक्ष्मी आपल्या उजव्या हातािे आई जगदंबेच्या रूपात प्रक् होते यािाठी हे िेत्र िवचव प्रकारच्या ववािशापािू िरक्षितु आहे.या िेत्राच्या महािति े अिेक िंतािा व कवींिा आपल्याकडे आकवषतव केले आहे. या करवीर िेत्रािे आपल्या भततांवर केलेले प्रेमािी ददलेले आशीवावद हे अतलीयु आहे.अिेही मािले जाते की श्री दत्तात्रेय अजिू पण या दठकाणी दाि करण्यािाठी दपारच्याु वेळेि येतात. महालक्ष्मी मंददर वास्तकलाु रचिेवर जे ििी कामाचे िंकेत आहेत त्यावरूि अिे िांगण्यात येईल की हे मदंदर चालतयु शािि काळातील म्हणजेच इिवी िि ६०० ते ७०० काळातील अिावे. मंददराची बाहेरची िंरचिा आखण स्तंभांची कलाकृती हा कलेचा एक अद्भतू िमिाु आहे. देवीची मतीू रत्िांपािू बिवली आहे. हीिद्धाु जवळ जवळ ५००० ते ६००० वषावपवीचीू अिावी अिे मािले जाते. महालक्ष्मीची मतीू ही ४० फकलो वजिाची आहे.

कोल्हापरू पय्व िामध्ये महालक्ष्मी मंददर हे प्रमखु स्थाि आहे. यािाठीच देवीच्या दशिव ाािाठी दरवषी लाखो भाववक भे् देतात

कोल्हापरू हे महाराष्ट्रातील एक महत्त्वाचे शहर अििू प्राचीि काळापाििू दक्षिण काशी या िावाि े ते प्रसिद्ध आहे.महालक्ष्मी मंददर आखण जोनतबाचे मंददर यामळेु हे एक प्रसिद्ध तीथिव ेत्र म्हणिहीू ओळखले जाते. तिच ऐनतहासिकदृष्ट्ट्टया कोल्हापरू महत्त्वाचे शहर अििू प्राचीि काळापाििू ते ववववध राजघराण्याचे राजधािीचे दठकाण होते.

दहदं ू धमशवास्त्राििारु , प्राचीि काळी केशी रािािाचा मलगाु कोल्हाििरु हा इथे राज्य करीत होता. याि े राज्यात अिाचार व िािाि त्रस्त करूि िोडले होते. म्हणिू देवांच्या प्राथिव ेवरूि महालक्ष्मीि े त्याच्याशी यद्धु केले. हे यद्धु िऊ ददवि चालले होते . आखण अक्ववि शद्धु पंचमीि महालक्ष्मीि े कोल्हाििरू या रािािाचा वध केला होता. त्यावेळेि कोल्हाििरू महालक्ष्मीला शरण गेला आखण त्याि े नतच्याकडे आपल्या िगराची कोल्हापरू व करवीर ही िावे आहेत तशीच चालू ठेवावीत अिा वर माचगतला. त्याप्रमाणे या िगरीला कोल्हापरू वा करवीर या िावाि े ओळखले जाते. ["श्री करवीर निवासिनी"] कोल्हापरावरु काही काळापवीू छत्रपती भोिले घराण्याचे शािि होते.

```
    </p1>
  </center>
  </body>
</html>
```

sangli.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
   <style>
p1
```

```
      {
      background-color: antiquewhite;
      display: block;
margin-top: 10px;       width:
800px;
      font-size: 20px;
      }
    </style>
</head>
<body >
  <center>
  <h1>िांगली</h1>
    <img src="sangliganapati.jpg"/>
```

<p1>िांगली हे महािगर पक्वचम महाराष्ट्रात (पक्वचम महाराष्ट्रातील प्रमखु महािगर आहे) विलेले आहे.

िांगली शहर क्जल््याचे मख्यु दठकाण. हे कृष्ट्णा िदीच्या काठावर विले अिू कृष्ट्णा-वारणा िद्यांचे िंगमस्थाि हररपरू या शहराच्या िैर्वत्येि ४ फकमी. वर आहे. िांगली येथे इ.ि १८७६ ते ८ िेब्रवारीु १९९८ पयतं िगरपासलका होती. ९ िेब्रवारीु १९९८ पाििू िांगली, समरज आखण कुपवाड शहर समळूि महािगरपासलका अक्स्तत्वात आली. इ.ि २०११ च्या जिगणििरु या महापासलकेची एकूण लोकिंख्या ५,०२,७९३ अिू त्यामध्ये २,४९,१५३ परूषु व २,५३,६४० क्स्त्रया होत्या. एकूण िेत्रिळ ११८ चौ. फकमी. अिू प्रभागांची िंख्या २० आहे. महाराष्ट्र औद्योचगक ववकाि महामंडळाचे (एमआयडीिी) प्रादेसशक कायावलय िांगली येथे अिू क्जल््याचे मख्यु दठकाण अिल्यामळेु क्जल्हास्तरावरील कें द्र व राज्यस्तरावरील शािकीय, निमशािकीय व महामंडळांची कायावलये येथे आहेत. वाहतकू व दळणवळणाच्या दृष्ट्ीिे िांगलीला प्रामख्यािेु कृषी उत्पादिांचा व्यापार मोठा आहे. हळदीची आसशयातील िवांत मोठी बाजारपेठ येथे अिल्यािे या शहराला ‘हळदीचे शहर’ अिे िंबोधले जाते. शैिखणक व िांस्कृनतक दृष्ट्ट्याही िांगली प्रसिद्घ आहे. कला वाखणज्य, ववज्ञाि, सशिणशास्त्र, ववधी, व्यवस्थापिशास्त्र, असभयांत्रकी, वास्तसशल्पु , औषधनिमावणशास्त्र, आयवेददकु , वैद्यक, दंतववज्ञाि, पयाववरण ववज्ञाि इ. शाखांची महाववद्यालये िांगलीत

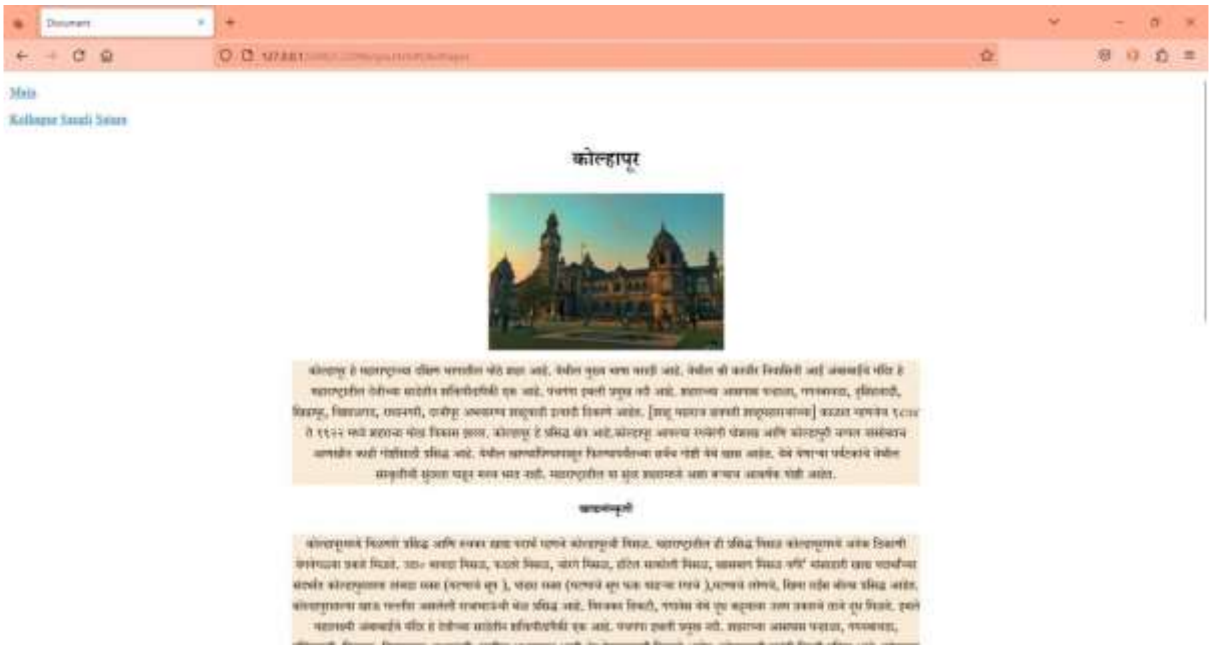
आहेत. मराठी िा्काचे हे उगमस्थाि अिल्यािे िांगलीला ‘िाट्टयपंढरी’ अिे िंबोधले जाते. िांगलीचे अचधपती िर चचतं ामणराव अप्पािाहेब प्वधिव यांिी राजाश्रय ददल्यामळेचु ववष्ट्णदािु भावे यांिी इ.ि १८४३ िाली सलदहलेल्या िीता स्वयंवर या िा्काचा प्रथम प्रयोग िांगली येथे झाला. िांगली क्जल्हा िाखरपट्टट्टयात येत अिल्यामळेु येथे अिेक िाखर-कारखािे आहेत. विंतदादा पाील शेतकरी िहकारी िाखर कारखािा हा आसशया खंडातील क्र.१ चा िहकारी िाखर कारखािा आहे. िांगली शहर हे पदहलवािांिाठी िद्धािु प्रसिद्ध आहे. दहदं केिरी मारुती मािे तिच त्रबजली या िावािे िंपणवू पक्वचम महाराष्ट्रात ओळखले जाणारे माजी आमदार कै. िंभाजी पवार हेही याच मातीतले. कुस्तीगीरांचे शहर अशीही िांगलीची ओळख आहे.</p1>

```
  </center>
  </body> </html>
```

satara.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
   <style>
p1
     {
     background-color: antiquewhite;
     display: block;
margin-top: 10px;       width:
800px;
     font-size: 20px;
     }
   </style>
</head>
<body >
  <center>
  <h1>िातारा</h1>
   <img src="satara.jpg"/>
  <p1>िातारा हे महाराष्ट्र राज्याच्या िातारा क्जल््यात कृष्ट्णा िदी व त्याच्या
उपिदी, वेण्णा यांच्या िंगमाजवळ क्स्थत एक शहर आहे. हे शहर १६ व्या शतकात स्थावपत
झाले होते आखण ते चक्रवती िम्रा् श्रीमंत छत्रपती शाहूमहाराज यांच्यापािि्ू मराठा
िाम्राज्याच्या राजधािीचे शहर होते. हे िातारा तहील तिच िातारा क्जल्हा यांचे
मख्यालयु       आहे. शहराचे िाव शहराभोवती अिलेल्या िात फकल्ल्या (िात-तारा)
पािि्ू आहे.िातारा क्जल् हा मराठी राज् याची राजधािी होती.भारतातील महाराष्ट्र
राज्यातील पणु ववभागातील एक क्जल्हा आहे. िातारा क्जल््याला मोठी ऐनतहासिक,
िामाक्जक व शैिखणक पाववभवमी्ू लाभली आहे. िातारा क्जल््यािे भारताच्या
स्वातंत्र्य लढ्यात व िामाक्जक जडणघडणीत मोठे योगदाि ददले आहे.
आधनिकु        भारतातिद्धाु        िातारा क्जल््यािे आपला वेगळा ठिा उम्वला
आहे. ्या क्जल््याच्या बहुतांश गावांमध्ये यवकु भारतीय िििे े मध्ये भरती होण्याची
परंपरा जपतात. त्यामळेु शरवीरांचा्ू  क्जल्हा अशी ओळख ्या क्जल््यािे निमावण
केली आहे.</p1>
  </center>
  </body>
</html>
```

Output:

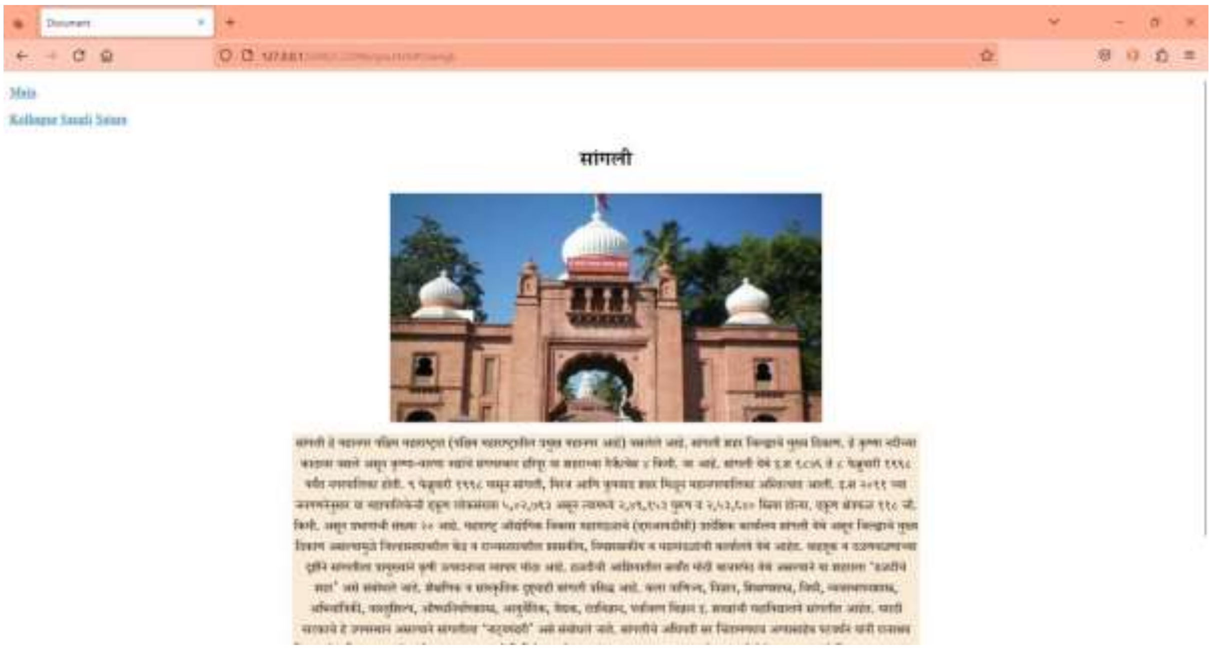

Document
Main
Kolhapur Sangli Satara

कोल्हापूर

[illegible]

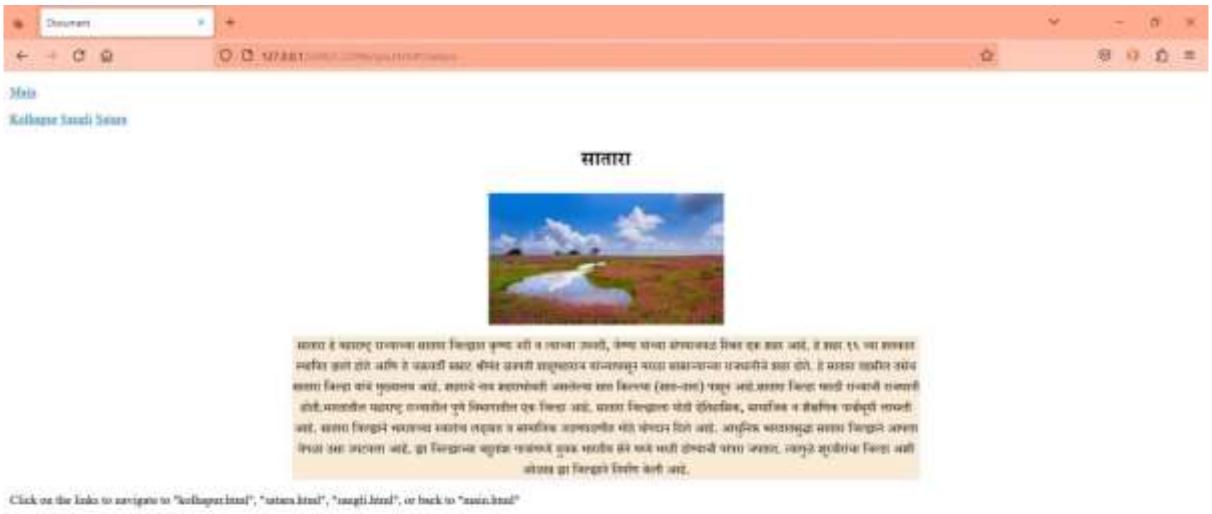
[illegible]

[illegible]


Document
Main
Kolhapur Sangli Satara

सांगली

[illegible]



[Main](#)

[Kolhapur](#) [Sangli](#) [Satara](#)

सातारा

सातारा हे महाराष्ट्र राज्याच्या सातारा जिल्ह्यात कृष्णा नदी व त्याच्या उपनदी, वेण्णा यांच्या संगमाजवळ स्थित एक शहर आहे. हे शहर १६ व्या शतकात स्थापित झाले होते आणि हे [illegible] छत्रपती शाहूमहाराज यांच्यापासून मराठा साम्राज्याच्या राजधानीचे शहर होते. हे सातारा [illegible] सातारा जिल्हा याचे मुख्यालय आहे. साताऱ्याचे नाव शहराभोवती असलेल्या सात किल्ल्या (सात-तारा) वरून आले.सातारा जिल्हा मराठी राज्याची राजधानी होती.भारतातील महाराष्ट्र राज्यातील पुणे विभागातील एक जिल्हा आहे. सातारा जिल्ह्याला मोठी ऐतिहासिक, धार्मिक व शैक्षणिक पार्श्वभूमी लाभली आहे. सातारा जिल्ह्याने भारताच्या स्वातंत्र्य लढ्यात व सामाजिक उभारणीत मोठे योगदान दिले आहे. आधुनिक भारतामध्ये सातारा जिल्ह्याने आपला ठसा उमटवला आहे. ह्या जिल्ह्याच्या बहुतेक गावांमधून एक भारतीय सैन्यात भरती होण्याची परंपरा आहे. त्यामुळे शूरवीरांचा जिल्हा अशी ओळख ह्या जिल्ह्याने निर्माण केली आहे.

Click on the links to navigate to "kolhapur.html", "satara.html", "sangli.html", or back to "main.html"