1.How to Install ReactJS on Windows?

AIM: installation of reactjs on windows.

Description:

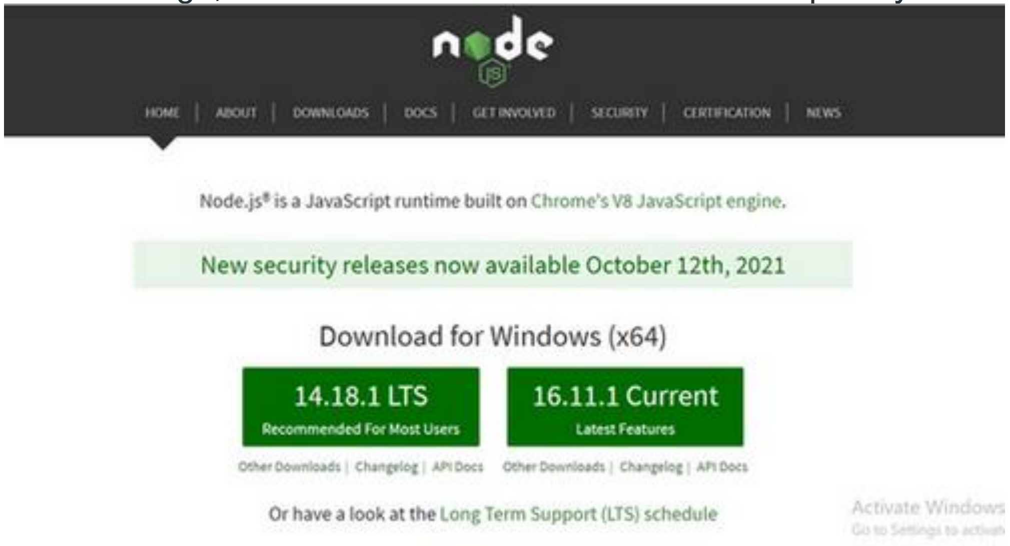
React JS -React is an open-source component-based front-end JavaScript library. It is used to create fast and interactive user interfaces for web and mobile applications. It is easy to create a dynamic application in React because it requires less coding and offer more functionality. It is used by big MNC and fresh new startups

Features of React:

1. **Reusable Components**: A single React app consists of many components each component have their own logic and code but we can easily reuse components any number of time hence reducing the developers time and increasing the efficiency of work
2. **Debugging**: React app can be easily debug using "react developer tools".It's a browser extension that can be used for both chrome as well as Firefox.

Installation Reactjs on Windows:

Step 1: Install Node.js installer for windows. Click on this link. Here install the LTS version (the one present on the left). Once downloaded open NodeJS without disturbing other settings, click on the **Next** button until it's completely installed.


node
HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS
14.18.1 LTS
Recommended For Most Users
16.11.1 Current
Latest Features
Other Downloads | Changelog | API Docs
Other Downloads | Changelog | API Docs
Activate Windows
Go to Settings to activate

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

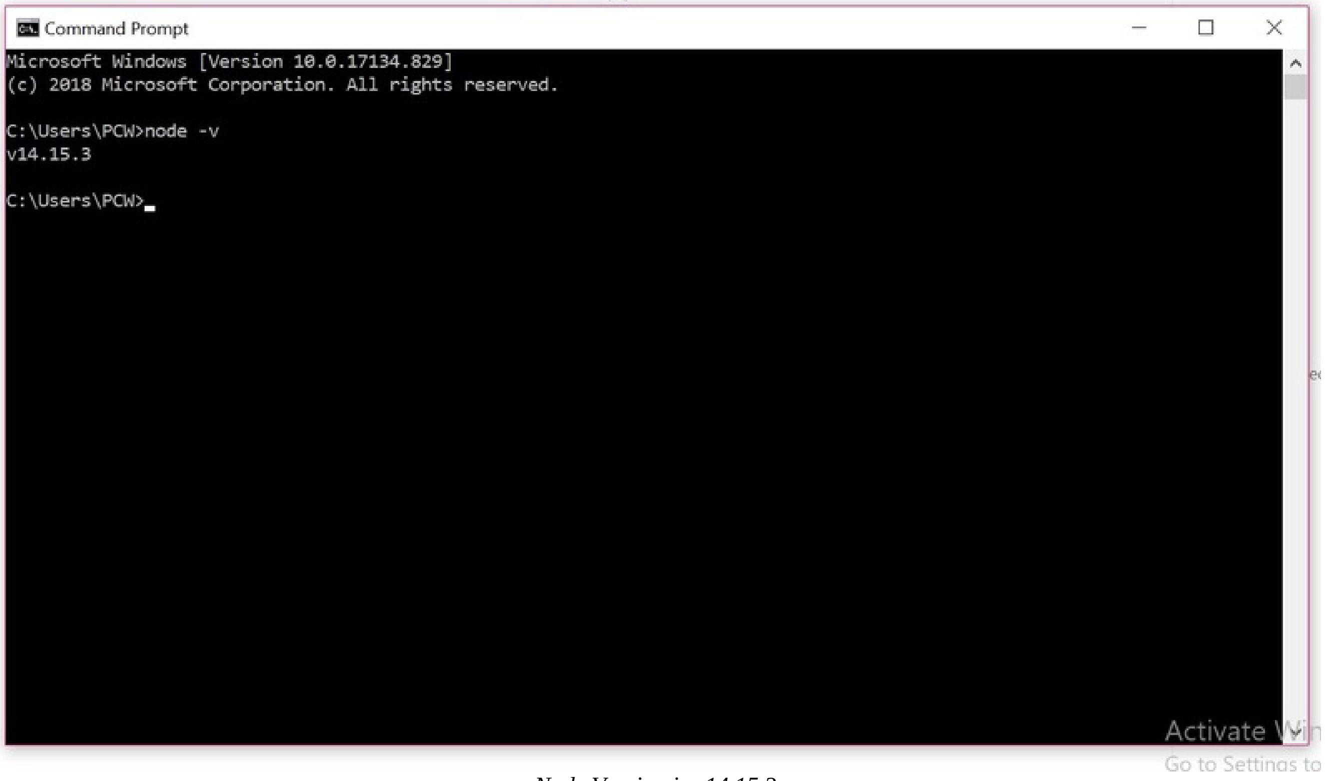
New security releases now available October 12th, 2021

Download for Windows (x64)

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

Step 2: Open command prompt to check whether it is completely installed or not type the command –>

```
node -v
```



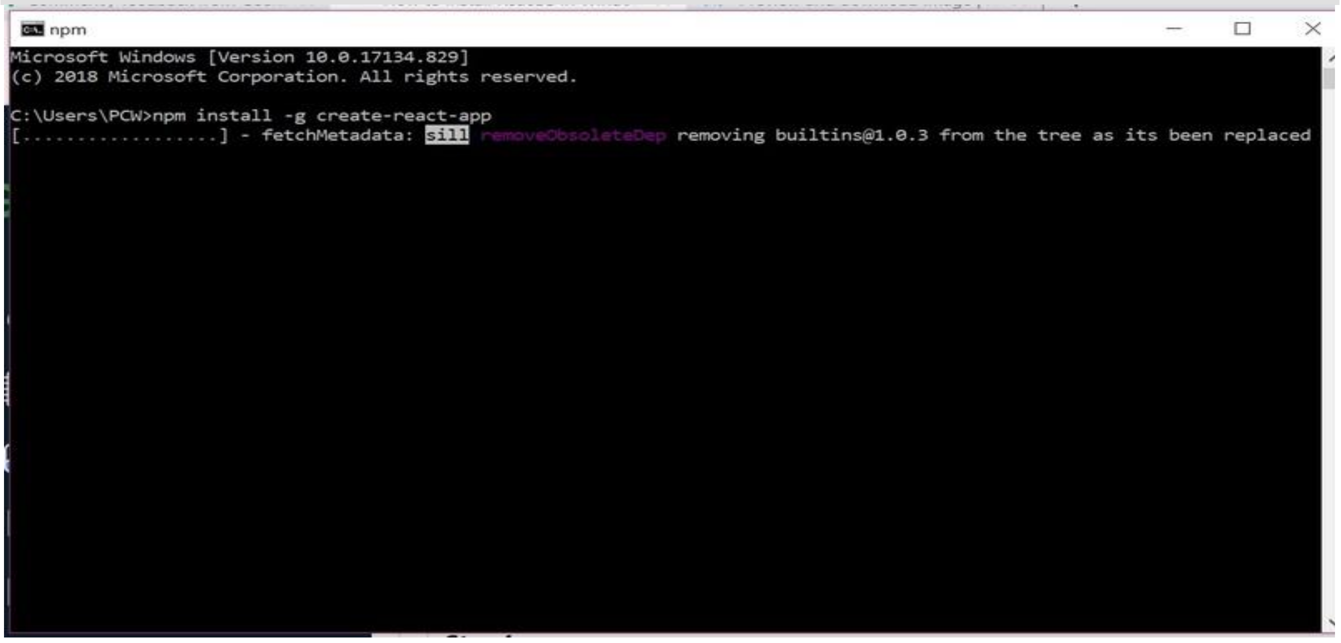
```
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PCW>node -v
v14.15.3

C:\Users\PCW>
```

Node Version is v14.15.3

If the installation went well it will give you the version you have installed

Step 3: Now in the terminal run the below command:

```
npm install -g create-react-app
```

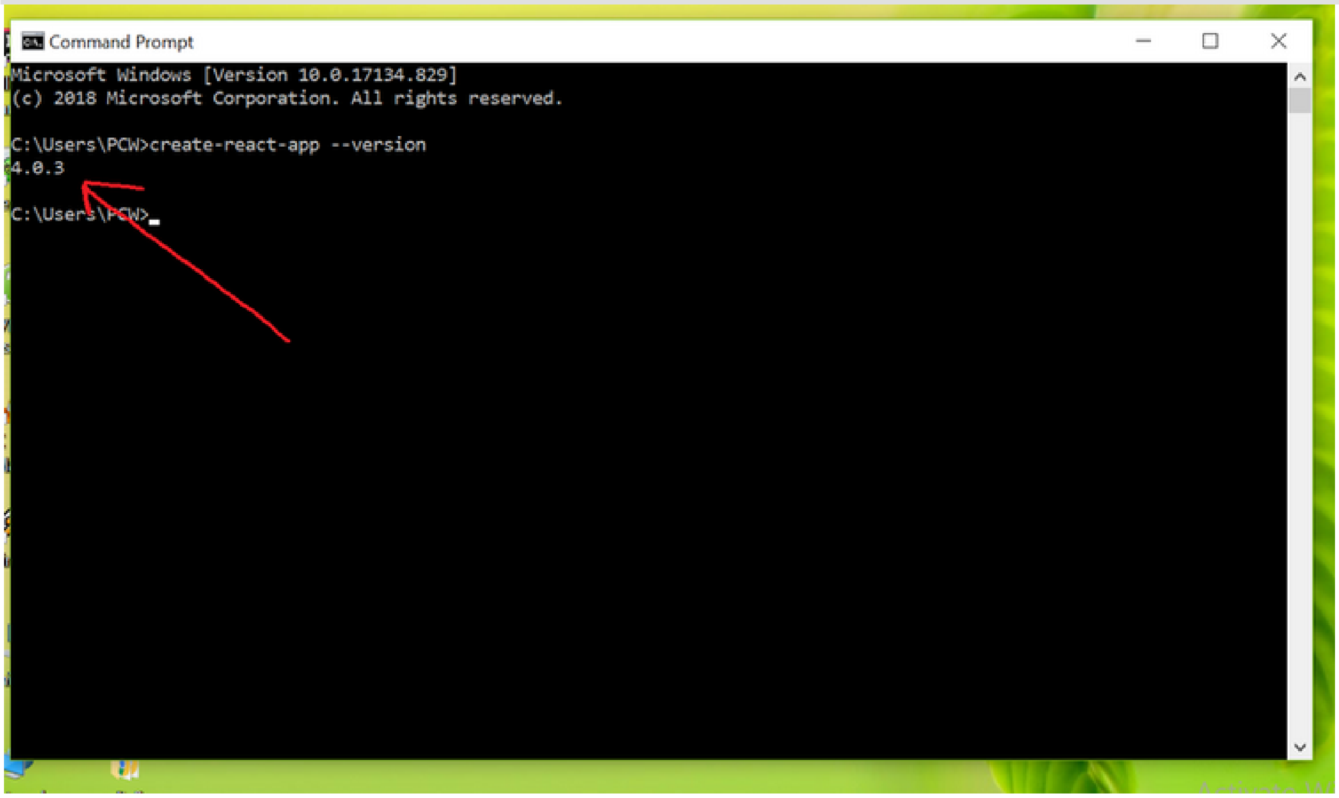


```
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PCW>npm install -g create-react-app
[..................] - fetchMetadata: sill removeObsoleteDep removing builtins@1.0.3 from the tree as its been replaced
```

Installation will take few seconds

It will globally install react app for you. To check everything went well run the command

```
create-react-app --version
```



```
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PCW>create-react-app --version
4.0.3

C:\Users\PCW>_
```

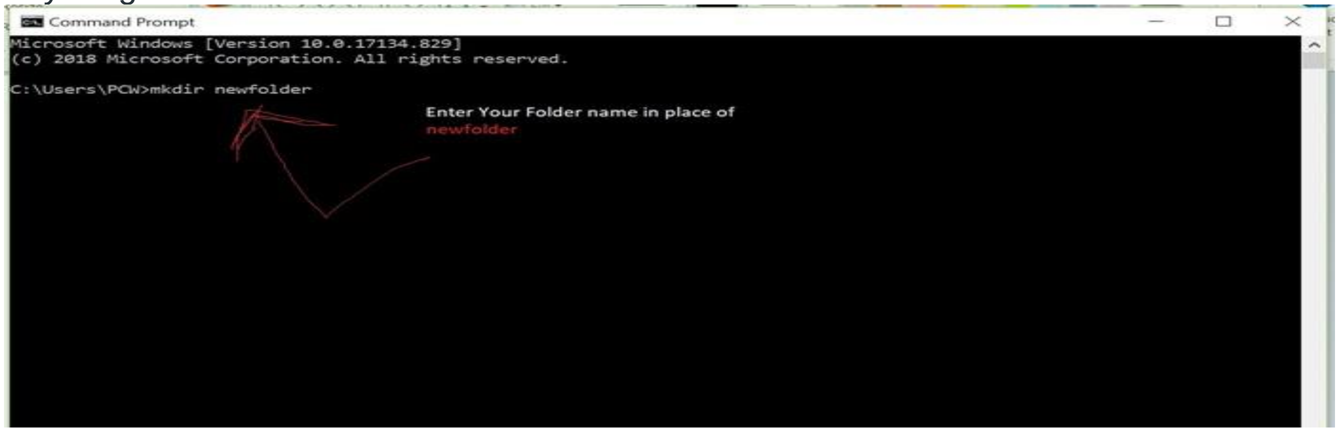
version 4.0.3

If everything went well it will give you the installed version of react app

Step 4: Now Create a new folder where you want to make your react app using the below command:

```
mkdir newfolder
```

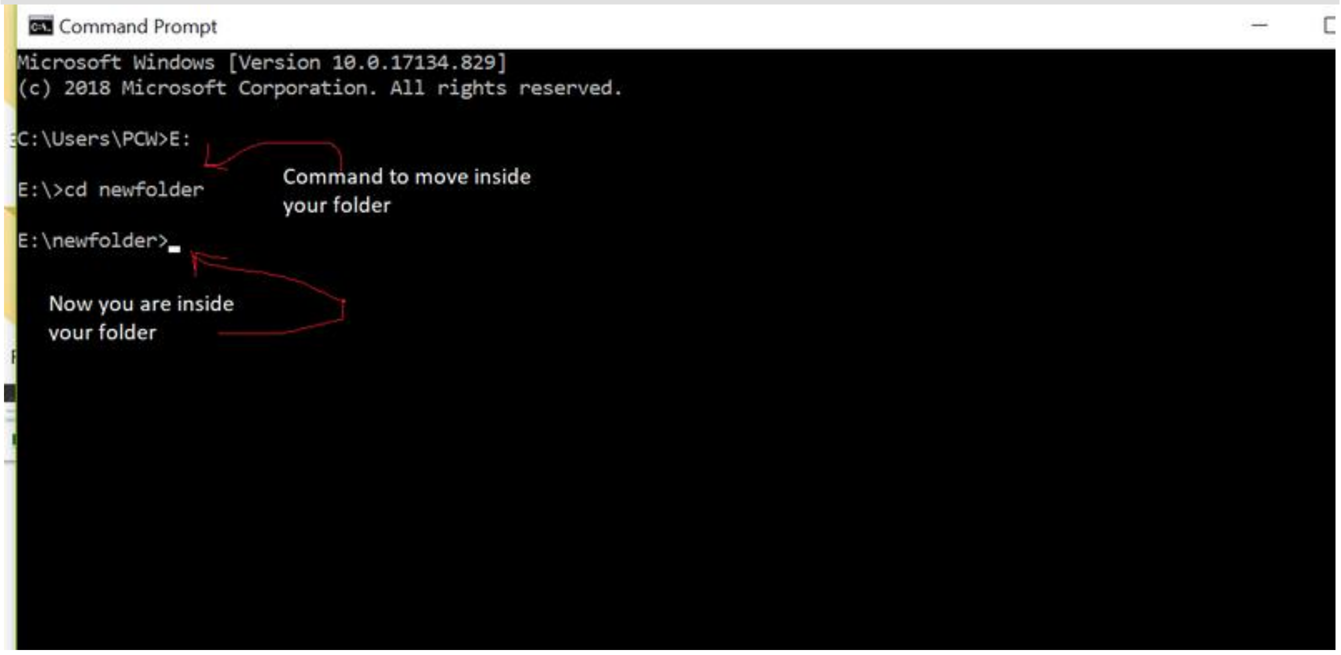
Note: The *newfolder* in the above command is the name of the folder and can be anything.



```
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PCW>mkdir newfolder
```

Move inside the same folder using the below command:

```
cd newfolder (your folder name)
```

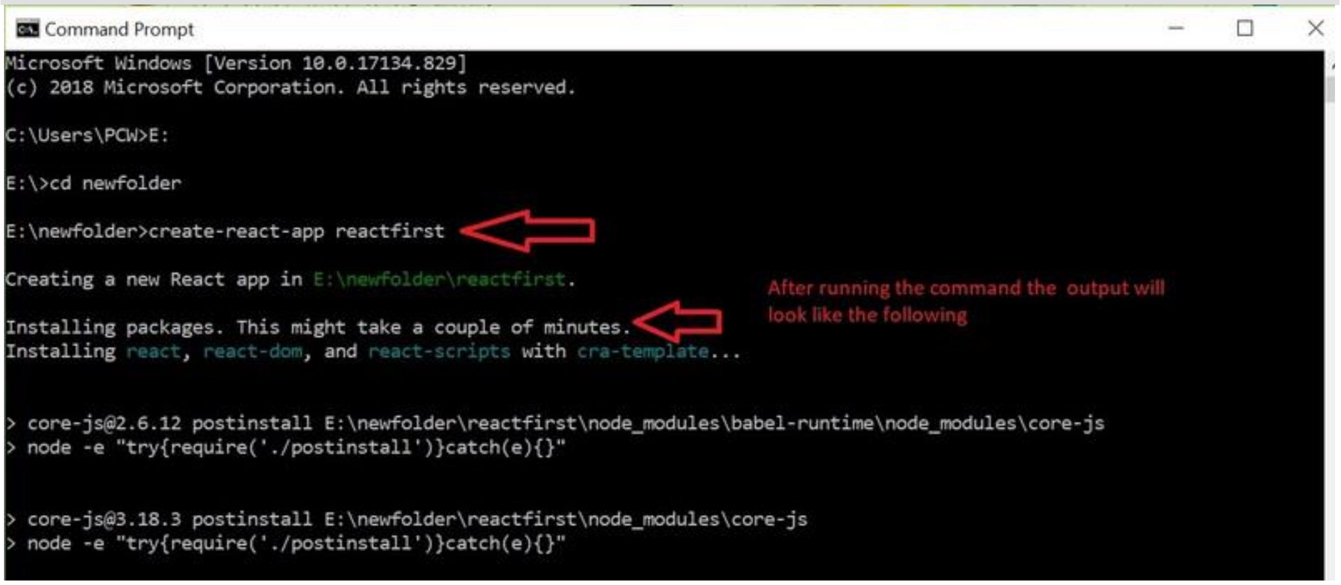


```
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PCW>E:

E:\>cd newfolder

E:\newfolder>
```

Command to move inside your folder
Now you are inside your folder

Step 5: Now inside this folder run the command –>

```
create-react-app reactfirst YOUR_APP_NAME
```

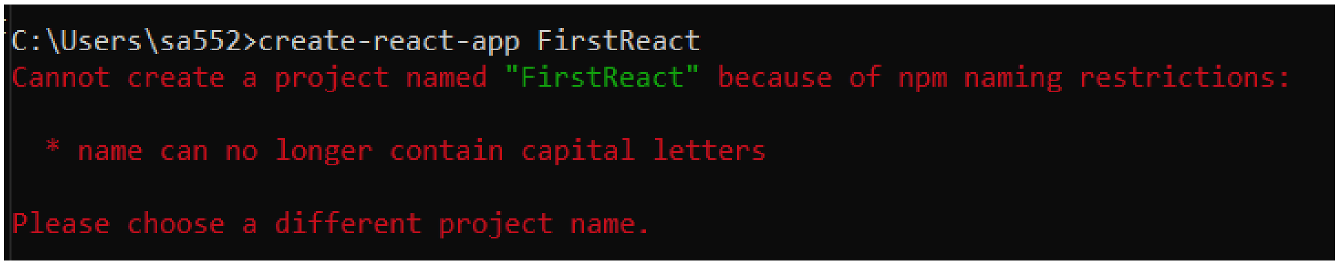


```
Microsoft Windows [Version 10.0.17134.829]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\PCW>E:

E:\>cd newfolder

E:\newfolder>create-react-app reactfirst

Creating a new React app in E:\newfolder\reactfirst.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...


> core-js@2.6.12 postinstall E:\newfolder\reactfirst\node_modules\babel-runtime\node_modules\core-js
> node -e "try{require('./postinstall')}catch(e){}"


> core-js@3.18.3 postinstall E:\newfolder\reactfirst\node_modules\core-js
> node -e "try{require('./postinstall')}catch(e){}"
```

After running the command the output will look like the following

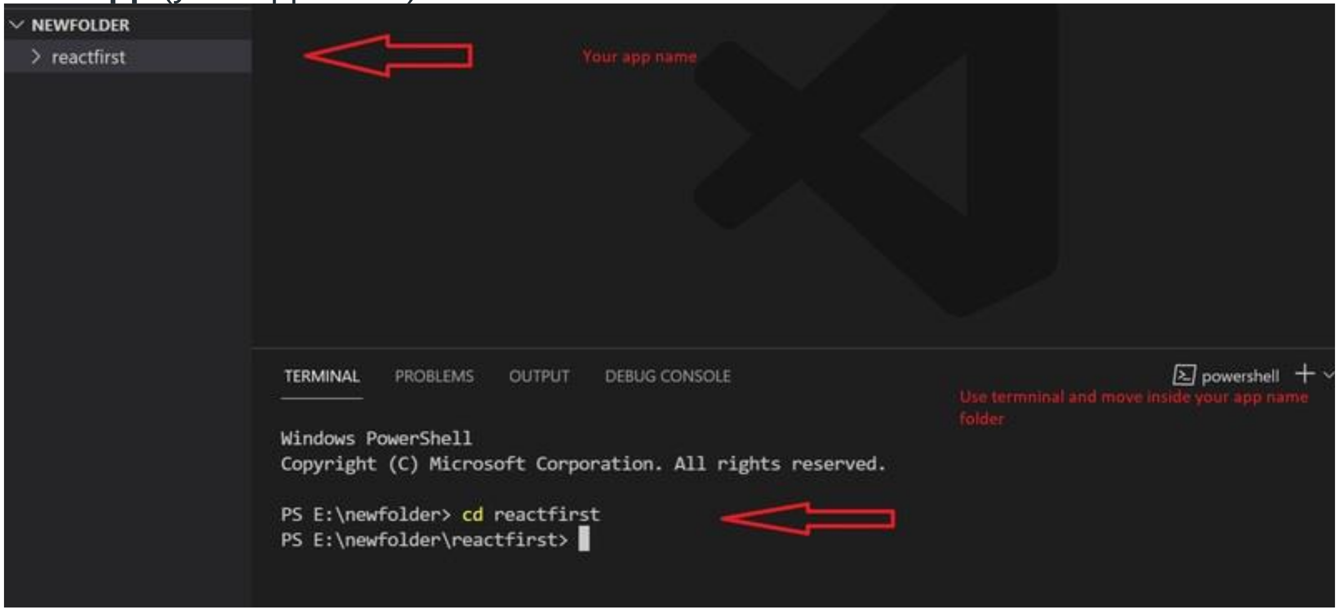
It will take some time to install the required dependencies

NOTE: Due to npm naming restrictions, names can no longer contain capital letters, thus type your app’s name in lowercase.



```
C:\Users\sa552>create-react-app FirstReact
Cannot create a project named "FirstReact" because of npm naming restrictions:

  * name can no longer contain capital letters

Please choose a different project name.
```

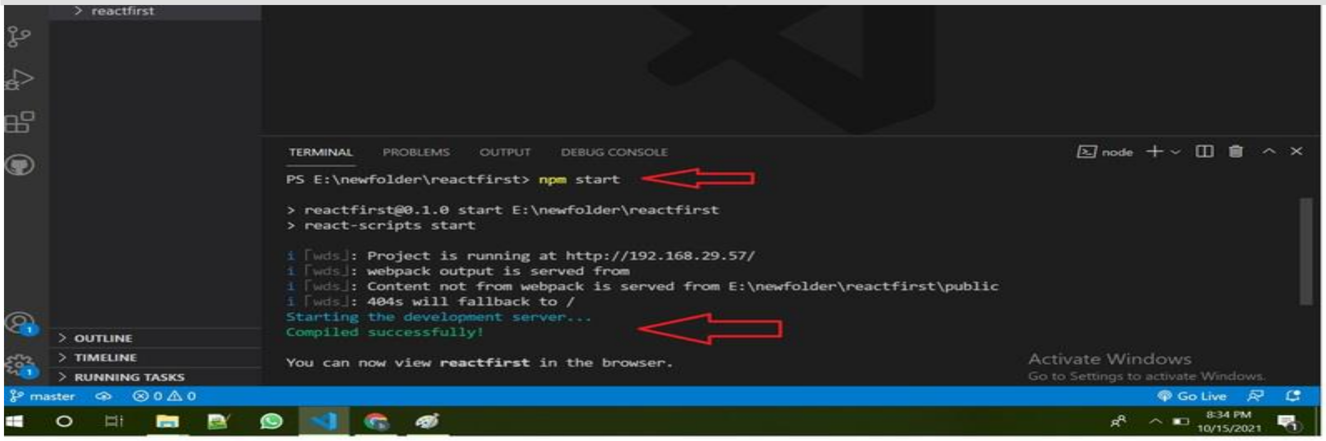
Step 6: Now open the IDE of your choice for eg. Visual studio code and open the folder where you have installed the react app **newfolder** (in the above example) inside the folder you will see your app's name **reactapp** (In our example). Use the terminal and move inside your app name folder.Use command **cd reactapp** (your app name)



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS E:\newfolder> cd reactfirst
PS E:\newfolder\reactfirst>
```

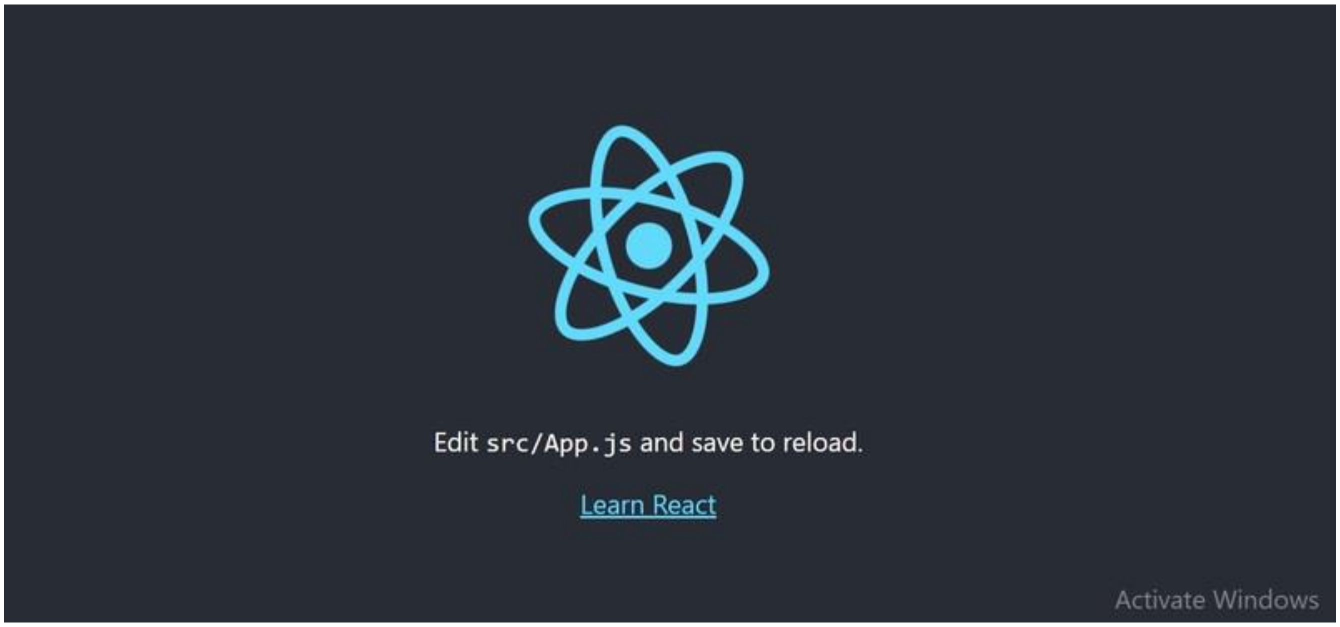
Step 7: To start your app run the below command :

```
npm start
```



```
PS E:\newfolder\reactfirst> npm start

> reactfirst@0.1.0 start E:\newfolder\reactfirst
> react-scripts start

i [wds]: Project is running at http://192.168.29.57/
i [wds]: webpack output is served from
i [wds]: Content not from webpack is served from E:\newfolder\reactfirst\public
i [wds]: 404s will fallback to /
Starting the development server...
Compiled successfully!

You can now view reactfirst in the browser.
```

Once you run the above command a new tab will open in your browser showing React logo as shown below :


Edit src/App.js and save to reload.

Learn React

Congratulation you have successfully installed the react-app and are ready to build awesome websites and app

2. WRITE A PROGRAM TO CREATE REACT APPLICATION HELLO-WORLD .

AIM :

Create a React Application hello-world. Implement a functional component FunctionClick.js and import it in App.js as a <FunctionClick/> tag which will display a button .Clicking on the button log a message to the console as “ButtonClicked”.(use event handling concept)

DESCRIPTION :

React is a popular JavaScript library used for building web applications. It was developed by Facebook and has become a widely adopted technology for front-end development due to its simplicity, efficiency, and flexibility.

Functional components are simpler to write and understand than class components because they are just plain functions that take in props as input and return JSX as output. They are also more lightweight and faster to render than class components because they don't have any additional overhead of maintaining state or lifecycle methods.

PROGRAM :

1. First, make sure you have Node.js installed on your computer.
2. Open your terminal/command prompt and navigate to the folder where you want to create your React application.
3. Run the following command to create a new React application: (lua)

```
npx create-react-app hello-world
```

4. Once the application is created, navigate into the project folder: (bash)

```
cd hello-world
```

5. Now, open the project in your favorite code editor. I'll assume you're using VSCode.
6. Open the App.js file from the src folder and replace its contents with the following code:(javascript)

```
import React from 'react';
import FunctionClick from './FunctionClick';

function App() {
  return (
<div className="App">
<FunctionClick />
</div>
  );
}

export default App;
```

7. Now, create a new file called FunctionClick.js inside the src folder and add the following code to it: (javascript)

```
import React from 'react';

function FunctionClick() {
  const handleClick = () => {
    console.log('ButtonClicked');
  }

  return (
<div>
<button onClick={handleClick}>Click me</button>
</div>
  )
}

export default FunctionClick;
```

8. Save the files and start the development server by running the following command in the terminal: (sql)

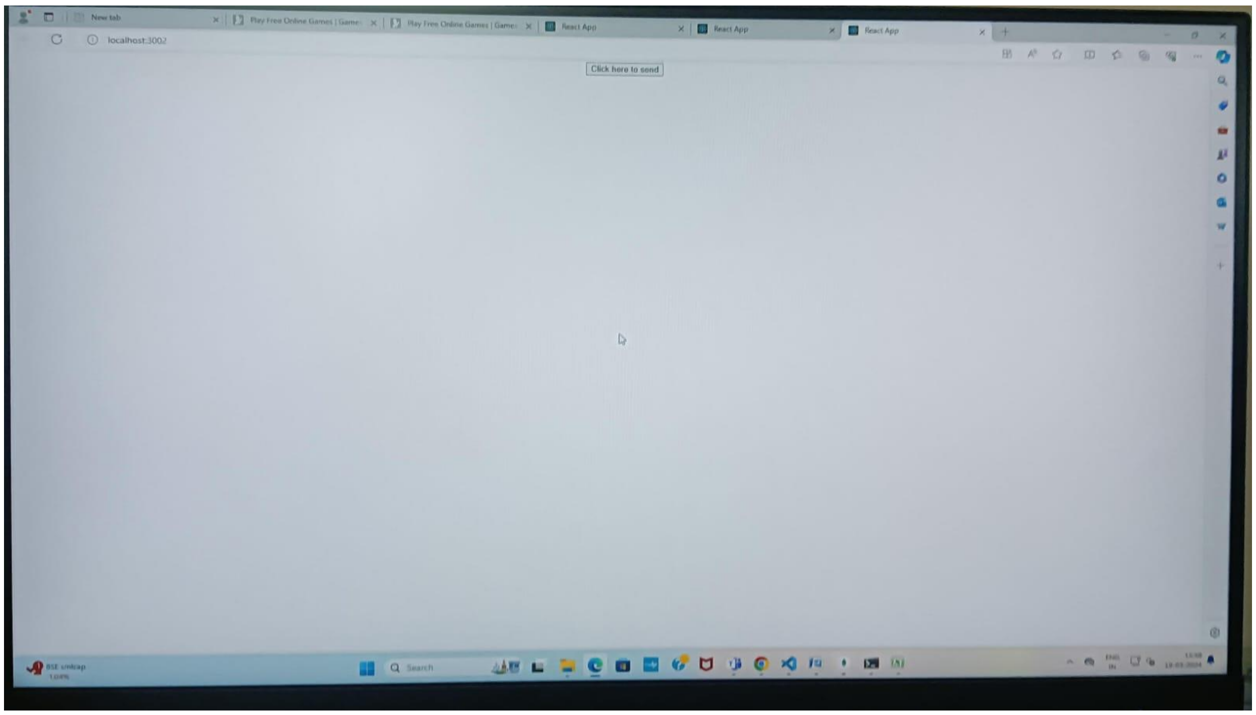
```
npm start
```

9. Now, open your browser and navigate to http://localhost:3000 to see your React application in action.
10. Click the button on the page and check the console in your browser's developer tools. You should see the message "ButtonClicked" logged to the console.

That's it! You have successfully created a React application called "hello-world" and implemented a functional component FunctionClick.js that logs a message to the console when a button is clicked.

OUTPUT :

ButtonClicked



3.WRITE A PROGRAM TO CREATING FORMS USING REACT JS .

AIM: Creating forms using ReactJs

DESCRIPTION:

The. form has the default HTML form behavior of browsing to a new page when the user submits the form. If you want this behavior in React, it just works. But in most cases, it's convenient to have a JavaScript function that handles the submission of the form and has access to the data that the user entered into the form. The standard way to achieve this is with a technique called "controlled components".

Controlled Components

In HTML, form elements such as <input>, <textarea>, and <select> typically maintain their own state and update it based on user input. In React, mutable state is typically kept in the state property of components, and only updated with setState().

We can combine the two by making the React state be the "single source of truth". Then the React component that renders a form also controls what happens in that form on subsequent user input. An input form element whose value is controlled by React in this way is called a "controlled component".

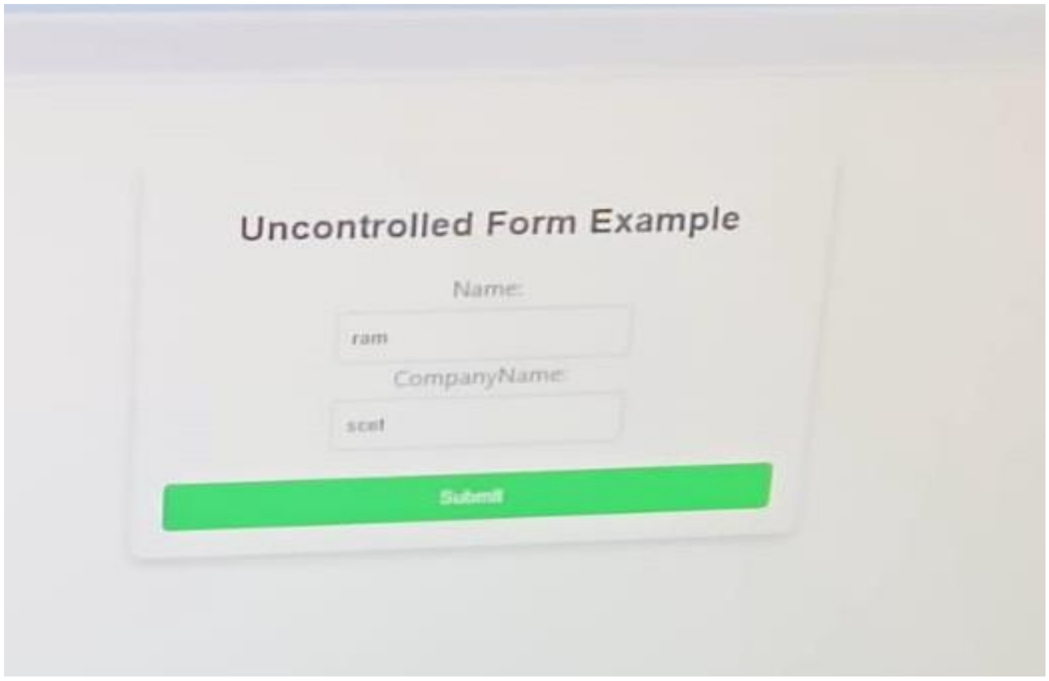
CODE:

APP.js:

```
importReact, { Component } from'react';

classAppextendsComponent {
constructor(props) {
super(props);
this.updateSubmit=this.updateSubmit.bind(this);
this.nameInput=React.createRef();
this.companyInput=React.createRef();
  }

updateSubmit(event) {
alert('You have entered the UserName and CompanyName successfully.');
event.preventDefault();
  }

render() {
return (
<divclassName="card"style={styles.card}>
<formonSubmit={this.updateSubmit}style={styles.form}>
<h1style={styles.heading}>Uncontrolled Form Example</h1>
<divstyle={styles.inputContainer}>
<labelstyle={styles.label}>Name:</label>
<inputtype="text"ref={this.nameInput}style={styles.input}/>
</div>
<divstyle={styles.inputContainer}>
<labelstyle={styles.label}>CompanyName:</label>
<inputtype="text"ref={this.companyInput}style={styles.input}/>
</div>
<inputtype="submit"value="Submit"style={styles.submitButton}/>
</form>
</div>
    );
  }
}

conststyles= {
```

```
card: {
width:'400px',
margin:'50px auto',
padding:'20px',
borderRadius:'10px',
boxShadow:'0 4px 8px 0 rgba(0, 0, 0, 0.2)',
backgroundColor:'#fff',
textAlign:'center'
  },
form: {
display:'flex',
flexDirection:'column',
alignItems:'center'
  },
heading: {
fontFamily:'Arial, sans-serif',
fontSize:'24px',
fontWeight:'bold',
marginBottom:'20px',
color:'#333'
  },
inputcontainer: {
marginBottom:'20px'
  },
label: {
display:'block',
marginBottom:'5px',
color:'#555'
  },
input: {
width:'100%',
padding:'10px',
borderRadius:'5px',
border:'1px solid #ccc',
boxSizing:'border-box'
  },
submitButton: {
width:'100%',
padding:'10px',
borderRadius:'5px',
border:'none',
backgroundColor:'#4CAF50',
color:'white',
cursor:'pointer',
transition:'background-color 0.3s',
marginTop:'20px'
  },
'submitButton:hover': {
backgroundColor:'#45a049'
  }
};

exportdefaultApp;
```

Output:


Uncontrolled Form Example
Name:
ram
CompanyName:
scet
Submit

4. Creating reactjs application for displaying details of bank accountName, accountnumber, ifsc code, bank branch, city.

AIM: write a react js application to display bank details.

Description:

In the BankAccount.js, we create a class with account name, account number, IFSC code, branch name and city.

In the App.js, we export the BankAccount.js.

Code:

Step1: goto react app and open src folder

create BankAccount.js .

<u>BankAccount.js:</u>

```
import React from 'react'

class BankAccount extends React.Component

{

  constructor(  )

  {

    super()

    this.state={

      accountName:'abc',

      accountNumber:10000,

      ifscCode:'SBI0987',

      branchName:'nsp',

      city:'nsp'

    }

  }

  render()

  {

    return(<div>

    <h1>{this.state.accountName}</h1>;

    <h1>{this.state.accountNumber}</h1>;
```

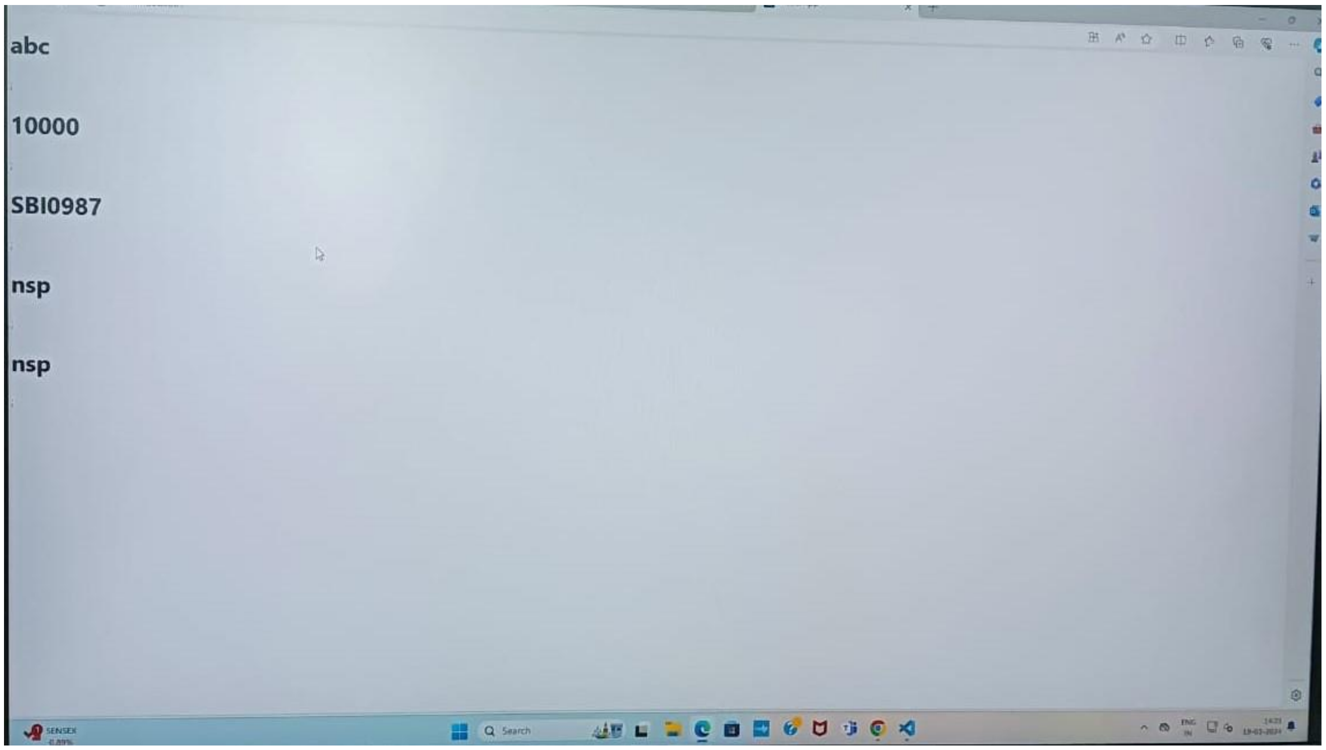
```
        <h1>{this.state.ifscCode}</h1>;
        <h1>{this.state.branchName}</h1>;
        <h1>{this.state.city}</h1>;
        </div>);
    }

}
export default BankAccount;
```

APP.JS:

Modify the app.js

```
import BankAccount from './BankAccount';
function App()
{
  return(
    <div className="App">
      <BankAccount></BankAccount>
    </div>
  );
}
export default App;
```

Output:



```
abc

10000

SBI0987

nsp

nsp
```

MONGO DB

5.Demonstrate how to create, drop a database and creation of collection in mongo DB

Aim: Demonstrate how to

a) create a database in mongo DB

b) drop a database in mongo DB

c) create a collection in mongo DB

Description:

MongoDB is a popular NoSQL database that uses a document-oriented data model. It stores data in flexible, JSON-like documents, making it easy to work with dynamic schemas. MongoDB is known for its scalability, high performance, and ease of use, making it suitable for a wide range of applications, from small startups to large enterprises. It's commonly used in web development, mobile apps, real-time analytics, and more.

Program:

a) To create a database in mongo DB

The use Command

MongoDB **use DATABASE_NAME** is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.

Syntax

Basic syntax of **use DATABASE** statement is as follows –

```
use DATABASE_NAME
```

Example

If you want to use a database with name **<mydb>**, then **use DATABASE** statement would be as follows –

```
>use mydb
switched to db mydb
```

To check your currently selected database, use the command **db**

```
>db
mydb
```

If you want to check your databases list, use the command **show dbs**.

```
>show dbs
local     0.78125GB
test     0.23012GB
```

Your created database (mydb) is not present in list. To display database, you need to insert at least one document into it.

```
>db.movie.insert({"name":"tutorials point"})
>show dbs
local      0.78125GB
mydb       0.23012GB
```

```
test          0.23012GB
```

b)create collection :

The createCollection() Method

MongoDB **db.createCollection(name, options)** is used to create collection.

Syntax

Basic syntax of **createCollection()** command is as follows −

db.createCollection(name, options)

In the command, **name** is name of collection to be created. **Options** is a document and is used to specify configuration of collection.

Parameter	Type	Description
Name	String	Name of the collection to be created
Options	Document	(Optional) Specify options about memory size and indexin

Basic syntax of **createCollection()** method without options is as follows −

```
>use test
switched to db test
>db.createCollection("mycollection")
{ "ok" : 1 }
>
```

You can check the created collection by using the command **show collections.**

```
>show collections
mycollection
system.indexes
```

The following example shows the syntax of **createCollection()** method with few important options –

```
> db.createCollection("mycol", { capped : true, autoIndexID :
true, size : 6142800, max : 10000 } ){
"ok" : 0,
"errmsg" : "BSON field 'create.autoIndexID' is an unknown
field.",
"code" : 40415,
"codeName" : "Location40415"
}
>
```

c)Drop data base

The dropDatabase() Method

MongoDB **db.dropDatabase()** command is used to drop a existing database.

Syntax

Basic syntax of **dropDatabase()** command is as follows –

```
db.dropDatabase()
```

This will delete the selected database. If you have not selected any database, then it will delete default 'test' database.

Example

First, check the list of available databases by using the command, **show dbs**.

```
>show dbs
local      0.78125GB
mydb       0.23012GB
test       0.23012GB
>
```

If you want to delete new database **<mydb>**,
then **dropDatabase()** command would be as follows –

```
>use mydb
switched to db mydb
>db.dropDatabase()
>{ "dropped" : "mydb", "ok" : 1 }
>
```

Now check list of databases.

```
>show dbs
local      0.78125GB
test       0.23012GB
>
```

6. Insert a single document, multiple documents, update and remove values in a document.

Aim: Insert the following:

a) single document

b) multiple documents

c) update

d) remove

program

a) inserting single document

The insert() Method

To insert data into MongoDB collection, you need to use MongoDB's **insert()** or **save()** method.

Syntax

The basic syntax of **insert()** command is as follows −

```
>db.COLLECTION_NAME.insert(document)
```

Example

```
> db.users.insert({
... _id : ObjectId("507f191e810c19729de860ea"),
... title: "MongoDB Overview",
... description: "MongoDB is no sql database",
... by: "tutorials point",
... url: "http://www.tutorialspoint.com",
... tags: ['mongodb', 'database', 'NoSQL'],
... likes: 100
... })
WriteResult({ "nInserted" : 1 })
>
```

b) inserting multiple documents:

The insertMany() method

You can insert multiple documents using the insertMany() method. To this method you need to pass an array of documents.

Example

Following example inserts three different documents into the empDetails collection using the insertMany() method.

```
> db.empDetails.insertMany(
	[
		{
			First_Name: "Radhika",
			Last_Name: "Sharma",
			Date_Of_Birth: "1995-09-26",
			e_mail: "radhika_sharma.123@gmail.com",
			phone: "9000012345"
		},
		{
			First_Name: "Rachel",
			Last_Name: "Christopher",
			Date_Of_Birth: "1990-02-16",
			e_mail: "Rachel_Christopher.123@gmail.com",
			phone: "9000054321"
		},
		{
			First_Name: "Fathima",
			Last_Name: "Sheik",
			Date_Of_Birth: "1990-02-16",
			e_mail: "Fathima_Sheik.123@gmail.com",
			phone: "9000054321"
		}
	]
)
```

```
{
      "acknowledged" : true,
      "insertedIds" : [
            ObjectId("5dd631f270fb13eec3963bed"),
            ObjectId("5dd631f270fb13eec3963bee"),
            ObjectId("5dd631f270fb13eec3963bef")
      ]
}
>
```

MongoDB Update() Method

The update() method updates the values in the existing document.

Syntax

The basic syntax of **update()** method is as follows –

```
>db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA)
```

Example

Consider the mycol collection has the following data.

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB
Overview"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL
Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials
Point Overview"}
```

Following example will set the new title 'New MongoDB Tutorial' of the documents whose title is 'MongoDB Overview'.

```
>db.mycol.update({'title':'MongoDB
Overview'},{$set:{'title':'New MongoDB Tutorial'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" :
1 })
>db.mycol.find()
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"New
MongoDB Tutorial"}
```

```
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL 
Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials 
Point Overview"}
>
```

By default, MongoDB will update only a single document. To update multiple documents, you need to set a parameter 'multi' to true.

```
>db.mycol.update({'title':'MongoDB Overview'},
   {$set:{'title':'New MongoDB Tutorial'}},{multi:true})
```

The remove() Method

Syntax

Basic syntax of **remove()** method is as follows −

>db.COLLECTION_NAME.remove(DELLETION_CRITTERIA)

Example

Consider the mycol collection has the following data.

```
{_id : ObjectId("507f191e810c19729de860e1"), title: "MongoDB 
Overview"},
{_id : ObjectId("507f191e810c19729de860e2"), title: "NoSQL 
Overview"},
{_id : ObjectId("507f191e810c19729de860e3"), title: 
"Tutorials Point Overview"}
```

Following example will remove all the documents whose title is 'MongoDB Overview'.

```
>db.mycol.remove({'title':'MongoDB Overview'})
WriteResult({"nRemoved" : 1})
> db.mycol.find()
{"_id" : ObjectId("507f191e810c19729de860e2"), "title" : 
"NoSQL Overview" }
{"_id" : ObjectId("507f191e810c19729de860e3"), "title" : 
"Tutorials Point Overview" }
```

Remove Only One

If there are multiple records and you want to delete only the first record, then set **justOne** parameter in **remove()** method.

```
>db.COLLECTION_NAME.remove(DELETION_CRITERIA,1)
```

Remove All Documents

If you don't specify deletion criteria, then MongoDB will delete whole documents from the collection. **This is equivalent of SQL's truncate command.**

```
> db.mycol.remove({})
WriteResult({ "nRemoved" : 2 })
> db.mycol.find()
>
```

7.Implement aggregation functions- $group,$match,$sort,distinct,count.

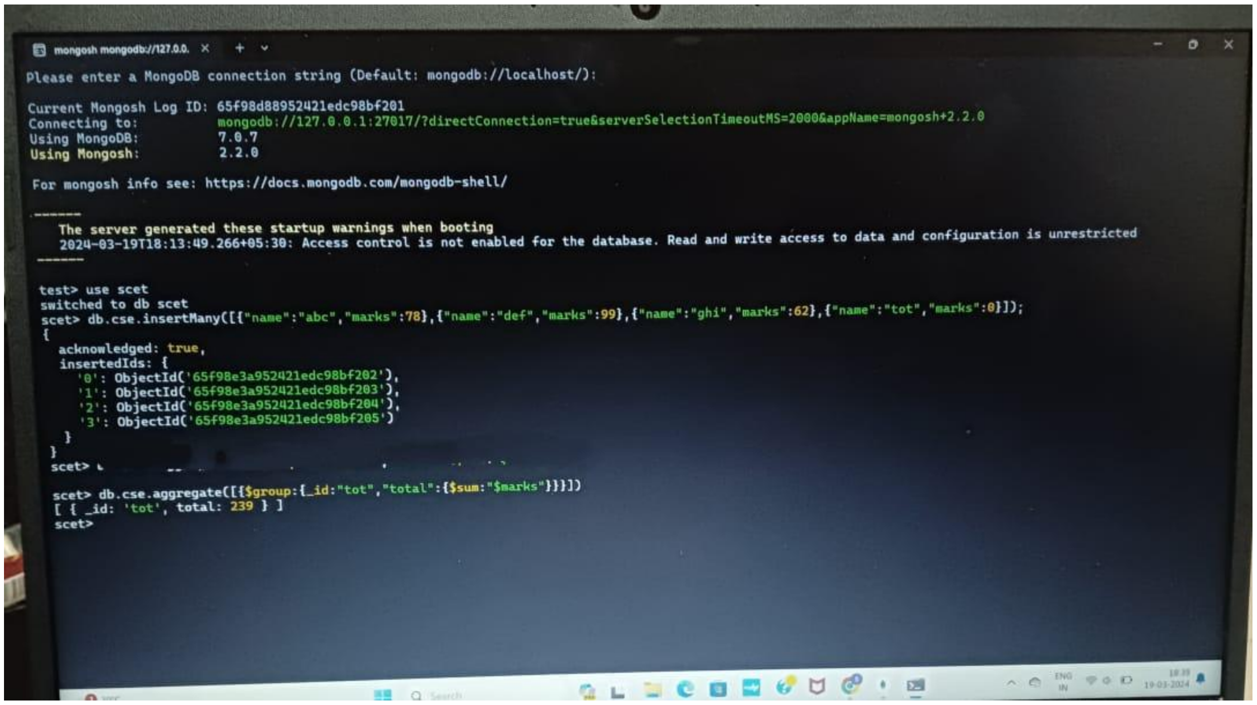
Aim: Implement Aggregation functions

Description:

Aggregation operations process the data records/documents and return computed results. It collects values from various documents and groups them together and then performs different types of operations on that grouped data.

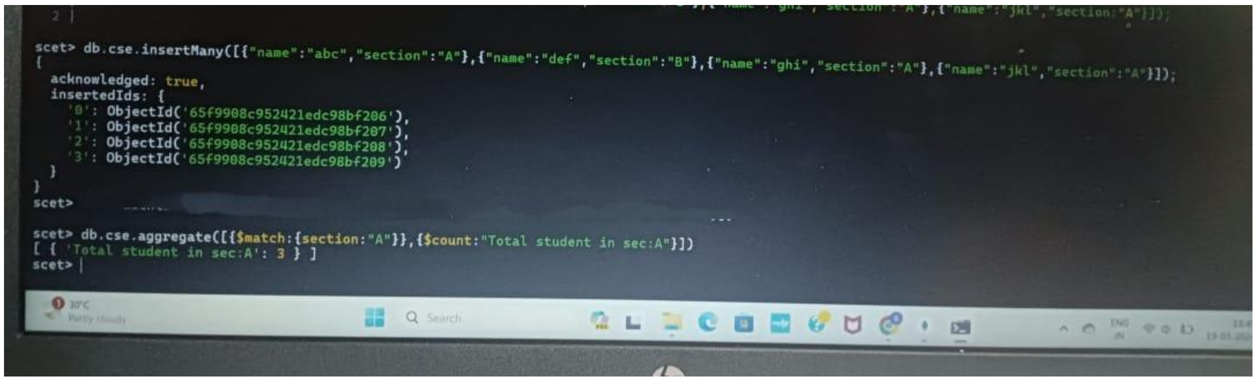
Program:

a) $group:



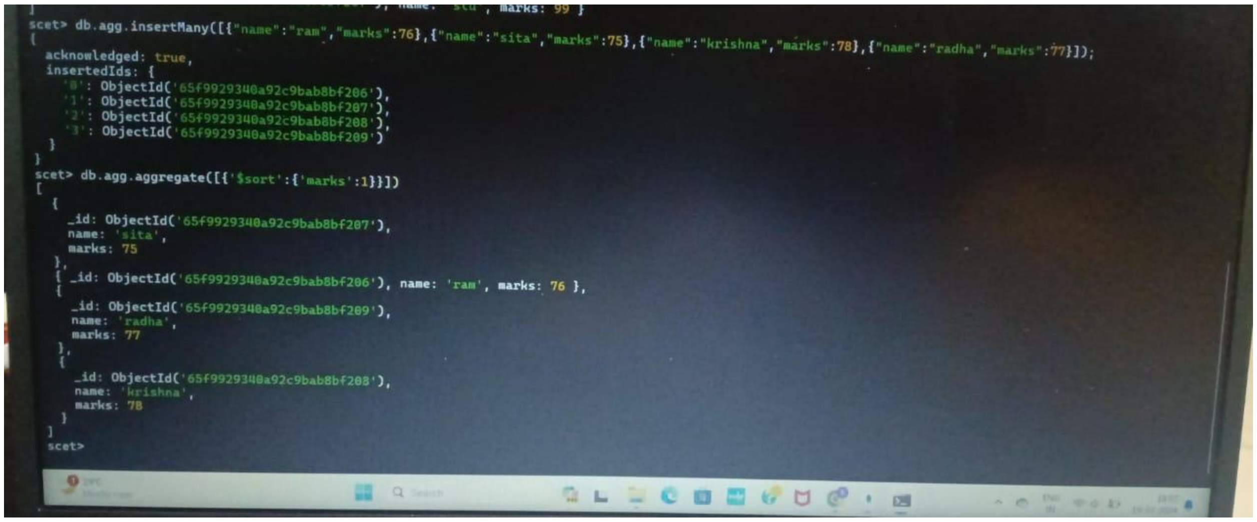
```
Please enter a MongoDB connection string (Default: mongodb://localhost/):

Current Mongosh Log ID: 65f98d88952421edc98bf201
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.2.0
Using MongoDB:          7.0.7
Using Mongosh:          2.2.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-03-19T18:13:49.266+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use scet
switched to db scet
scet> db.cse.insertMany([{"name":"abc","marks":78},{"name":"def","marks":99},{"name":"ghi","marks":62},{"name":"tot","marks":0}]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('65f98e3a952421edc98bf202'),
    '1': ObjectId('65f98e3a952421edc98bf203'),
    '2': ObjectId('65f98e3a952421edc98bf204'),
    '3': ObjectId('65f98e3a952421edc98bf205')
  }
}
scet>

scet> db.cse.aggregate([{$group:{_id:"tot","total":{$sum:"$marks"}}}])
[ { _id: 'tot', total: 239 } ]
scet>
```

b) $match: to filter out the documents.



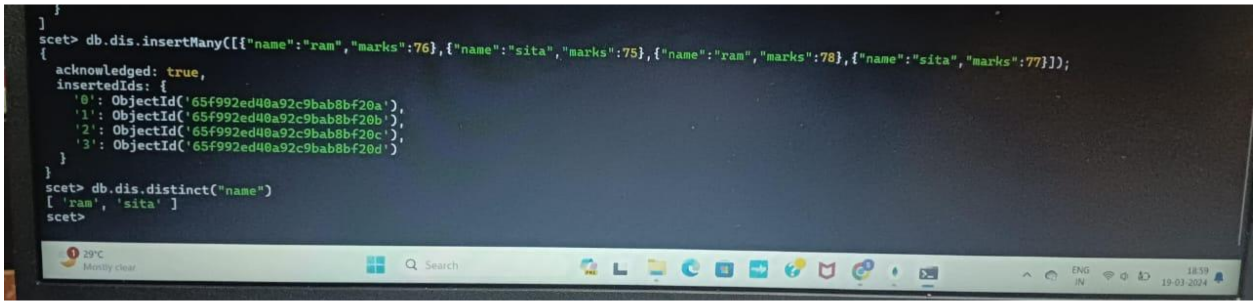
```
scet> db.cse.insertMany([{"name":"abc","section":"A"},{"name":"def","section":"B"},{"name":"ghi","section":"A"},{"name":"jkl","section":"A"}]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('65f9908c952421edc98bf206'),
    '1': ObjectId('65f9908c952421edc98bf207'),
    '2': ObjectId('65f9908c952421edc98bf208'),
    '3': ObjectId('65f9908c952421edc98bf209')
  }
}
scet>

scet> db.cse.aggregate([{$match:{section:"A"}},{$count:"Total student in sec:A"}])
[ { 'Total student in sec:A': 3 } ]
scet>
```

c) $sort: to sort in ascending order



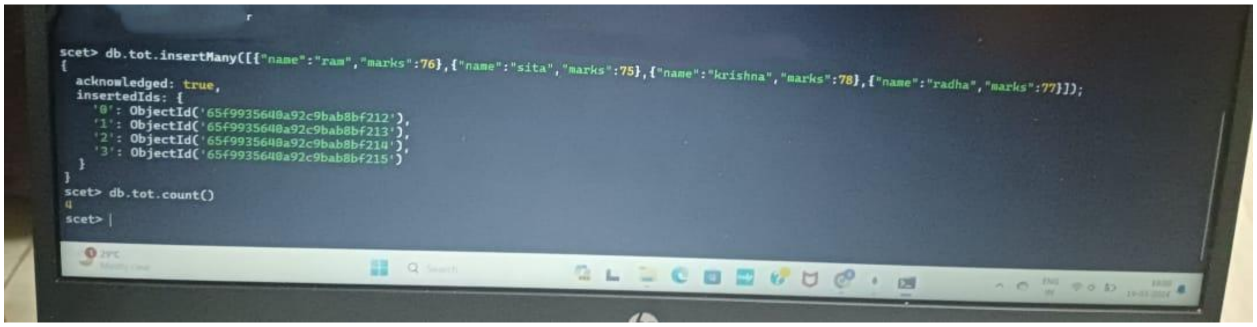
```
                                    name: 'stu', marks: 99 }
]
scet> db.agg.insertMany([{"name":"ram","marks":76},{"name":"sita","marks":75},{"name":"krishna","marks":78},{"name":"radha","marks":77}]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('65f9929340a92c9bab8bf206'),
    '1': ObjectId('65f9929340a92c9bab8bf207'),
    '2': ObjectId('65f9929340a92c9bab8bf208'),
    '3': ObjectId('65f9929340a92c9bab8bf209')
  }
}
scet> db.agg.aggregate([{'$sort':{'marks':1}}])
[
  {
    _id: ObjectId('65f9929340a92c9bab8bf207'),
    name: 'sita',
    marks: 75
  },
  { _id: ObjectId('65f9929340a92c9bab8bf206'), name: 'ram', marks: 76 },
  {
    _id: ObjectId('65f9929340a92c9bab8bf209'),
    name: 'radha',
    marks: 77
  },
  {
    _id: ObjectId('65f9929340a92c9bab8bf208'),
    name: 'krishna',
    marks: 78
  }
]
scet>
```

d) Distinct: finds distinct values of the specified field.



```
  }
]
scet> db.dis.insertMany([{"name":"ram","marks":76},{"name":"sita","marks":75},{"name":"ram","marks":78},{"name":"sita","marks":77}]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('65f992ed40a92c9bab8bf20a'),
    '1': ObjectId('65f992ed40a92c9bab8bf20b'),
    '2': ObjectId('65f992ed40a92c9bab8bf20c'),
    '3': ObjectId('65f992ed40a92c9bab8bf20d')
  }
}
scet> db.dis.distinct("name")
[ 'ram', 'sita' ]
scet>
```

e) Count: find the total number of the document, it counts them and return a number.



```
scet> db.tot.insertMany([{"name":"ram","marks":76},{"name":"sita","marks":75},{"name":"krishna","marks":78},{"name":"radha","marks":77}]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('65f99356408a92c9bab8bf212'),
    '1': ObjectId('65f99356408a92c9bab8bf213'),
    '2': ObjectId('65f99356408a92c9bab8bf214'),
    '3': ObjectId('65f99356408a92c9bab8bf215')
  }
}
scet> db.tot.count()
4
scet> |
```

8. Install TOMCAT web server and APACHE.

AIM: Install TOMCAT web server and APACHE.

While installation assign port number 8080 to APACHE. Make sure that these ports areavailable i.e., no other process is using this port.

DESCRIPTION:

- **Set the JAVA_HOME Variable**

You must set the `JAVA_HOME` environment variable to tell Tomcat where to find Java. Failing to properly set this variable prevents Tomcat from handling JSP pages. This variable should list the base JDK installation directory, not the bin subdirectory.

On Windows XP, you could also go to the Start menu, select Control Panel, choose System, click on the Advanced tab, press the Environment Variables button at the bottom, and enter the `JAVA_HOME` variable and value directly as:

Name: JAVA_HOME

Value: C:\jdk

- ### Set the CLASSPATH

Since servlets and JSP are not part of the Java 2 platform, standard edition, you have to identify the servlet classes to the compiler. The server already knows about the servlet classes, but the compiler (i.e., `javac`) you use for development probably doesn't. So, if you don't set your `CLASSPATH`, attempts to compile servlets, tag libraries, or other classes that use the servlet and JSP APIs will fail with error messages about unknown classes.

Name: JAVA_HOME

Value: *install_dir/common/lib/servlet-api.jar*

Turn on Servlet Reloading

next step is to tell Tomcat to check the modification dates of the class files of requested servlets and reload ones that have changed since they were loaded into the server's memory. This slightly degrades performance in deployment situations, so is turned off by default. However, if you fail to turn it on for your development server, you'll have to restart the server every time you recompile a servlet that has already been loaded into the server's memory.

To turn on servlet reloading, edit *install_dir/conf/server.xml* and add a `DefaultContext` subelement to the main `Host` element and supply `true` for the `reloadable` attribute. For example, in Tomcat 5.0.27, search for this entry:

<Host name="localhost" debug="0" appBase="webapps" ...>

and then insert the following immediately below it:

<DefaultContext reloadable="true"/>

Be sure to make a backup copy of *server.xml* before making the above change.

- ## Enable the Invoker Servlet

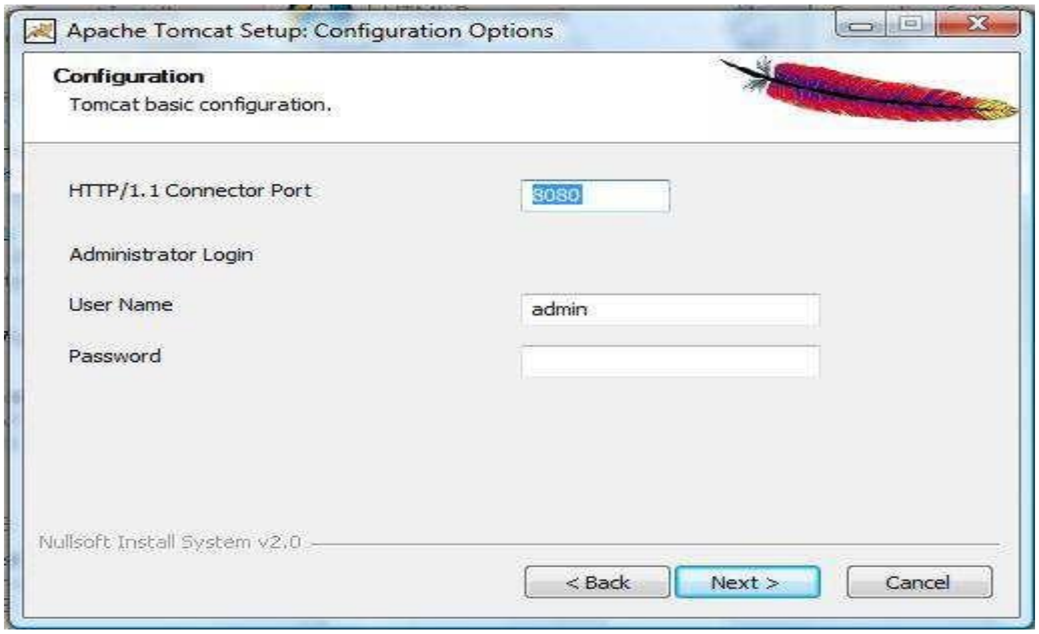
The invoker servlet lets you run servlets without first making changes to your Web application's deployment descriptor. Instead, you just drop your servlet into *WEB-INF/classes* and use the URL *http://host/servlet/ServletName*. The invoker servlet is extremely convenient when you are learning and even when you are doing your initial development.

To enable the invoker servlet, uncomment the following `servlet` and `servlet-mapping` elements in *install_dir/conf/web.xml*. Finally, remember to make a backup copy of the original version of this file before you make the changes.

```
<servlet>
```

```
    <servlet-name>invoker</servlet-name>
    <servlet-class>
      org.apache.catalina.servlets.InvokerServlet
    </servlet-class>
    ...
</servlet>
...
<servlet-mapping>
<servlet-name>invoker</servlet-name>
    <url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
```

OUTPUT :



Apache Tomcat Setup: Configuration Options

Configuration
Tomcat basic configuration.

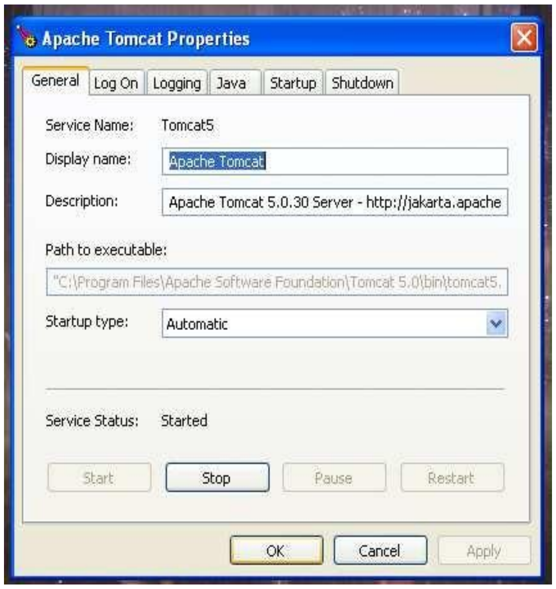
HTTP/1.1 Connector Port: 8080

Administrator Login

User Name: admin

Password:

Nullsoft Install System v2.0



Apache Tomcat Properties

Service Name: Tomcat5
Display name: Apache Tomcat
Description: Apache Tomcat 5.0.30 Server - http://jakarta.apache
Path to executable:
"C:\Program Files\Apache Software Foundation\Tomcat 5.0\bin\tomcat5.
Startup type: Automatic

Service Status: Started

RESULT: Thus TOMCAT web server was installed successfully.

9. Read the user id and passwords entered in the Login form (week1) and authenticatewith the values (user id and passwords) available in the cookies.

AIM: Read the user id and passwords entered in the Login form (week1) and authenticatewith the values (user id and passwords) available in the cookies.

I he is a valid user (i.e., user-name and password match) you should welcome him byname (user-name) else you should display “You are not an authenticated user “.

Use init-parameters to do this. Store the user-names and passwords in the webinf.xml andaccess them in the servlet by using the getInitParameters() method.

home.html:

```
<html>
<head>
 <title>Authentication</title>
</head>
<body>
<form action="ex1">
<label>Username </label>
<input type="text"size="20" name="user"><br><br>
password<input type="text" size="20"
name="pwd"><br><br>
<input type="submit" value="submit">
</form>
```

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

Example1.java

```
import
javax.servlet.*;
import java.io.*;
public class Example1 extends GenericServlet
{
  private String
  user1,pwd1,user2,pwd2,user3,pwd3,user4,pwd4,user5,pwd5; public
  void init(ServletConfig sc)
  {
    user1=sc.getInitParameter("username1");

pwd1=sc.getInitParameter("password1");

    user2=sc.getInitParameter("usernam
    e2");
    pwd2=sc.getInitParameter("passwor
    d2");

        user3=sc.getInitParameter("username3")
    ;
    pwd3=sc.getInitParameter("password3")
    ;
```

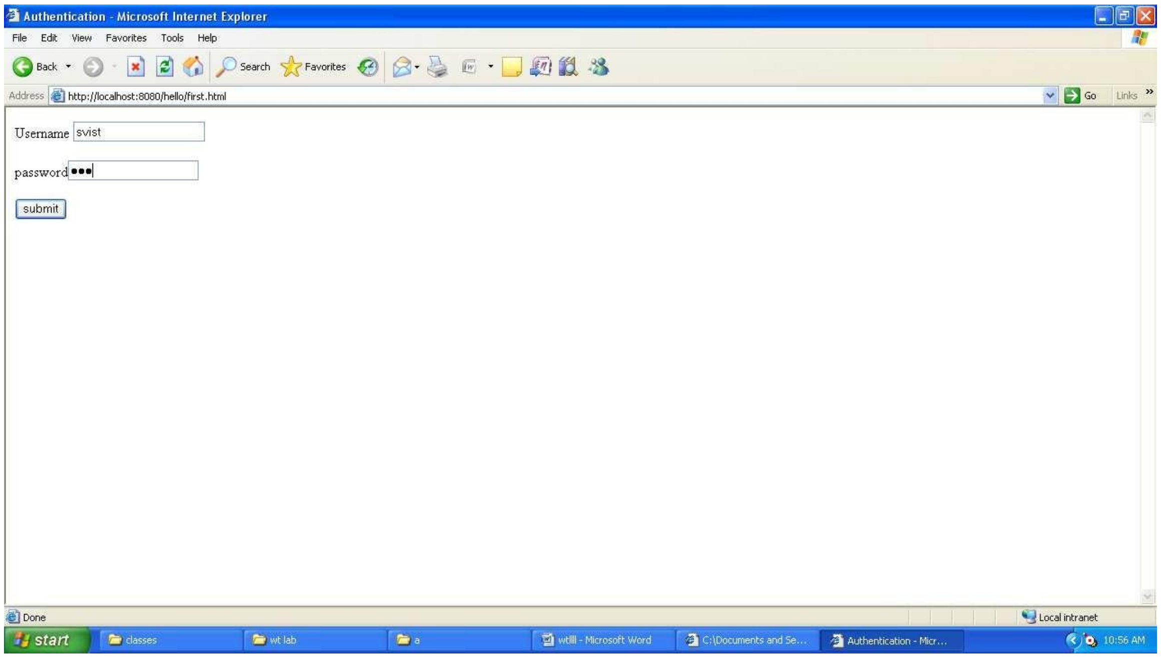
```
        user4=sc.getInitParameter("username4")
    ;
    pwd4=sc.getInitParameter("password4")
    ;
  }
  Public        void        service(ServletRequest      req,ServletResponse
                res)throwsServletException,IOException
 {
    res.setContentType("text/html");
    PrintWriter out=res.getWriter();
    user5=req.getParameter("user");
    pwd5=req.getParameter("pwd");

if((user5.equals(user1)&&pwd5.equals(pwd1))||(user5.equals(user2)&&pwd5.equals(pwd2)
)||(user5
.equals(user3)&&pwd5.equals(pwd3))||(user5.equals(user4)&&pwd5.equals(pwd4)))
      out.println("<p> welcome to"+user5.toUpperCase());
    else
      out.println("You are not authorized user");
 }
}
```

web.xml:

```
<web-app>

<servlet>

    <servlet-name>Example</servlet-name


<servlet-class>Example1</servlet-class>

<init-param>

<param-name>username1</param-name>

<param-value>svist</param-value>

</init-param>

<init-param>

<param-name>password1</param-name>

<param-value>cse</param-value>

</init-param>

<init-param>

<param-name>username2</param-name>

<param-value>1234</param-value>

</init-param>

<init-param>

<param-name>password2</param-name>

<param-value>4567</param-value>

</init-param>

<init-param>

<param-name>username3</param-name>
```

```
 <param-value>cse</param-value>
 </init-param>
 <init-param>
 <param-name>password3</param-name>
 <param-value>svist</param-value>
 </init-param>
 <init-param>
 <param-name>username4</param-name>
 <param-value>wt</param-value>
 </init-param>
 <init-param>
 <param-name>password4</param-name>
<param-value>lab</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>Example</servlet-name>
 <url-pattern>/ex1</url-pattern>
 </servlet-mapping>




 </web-app>
```

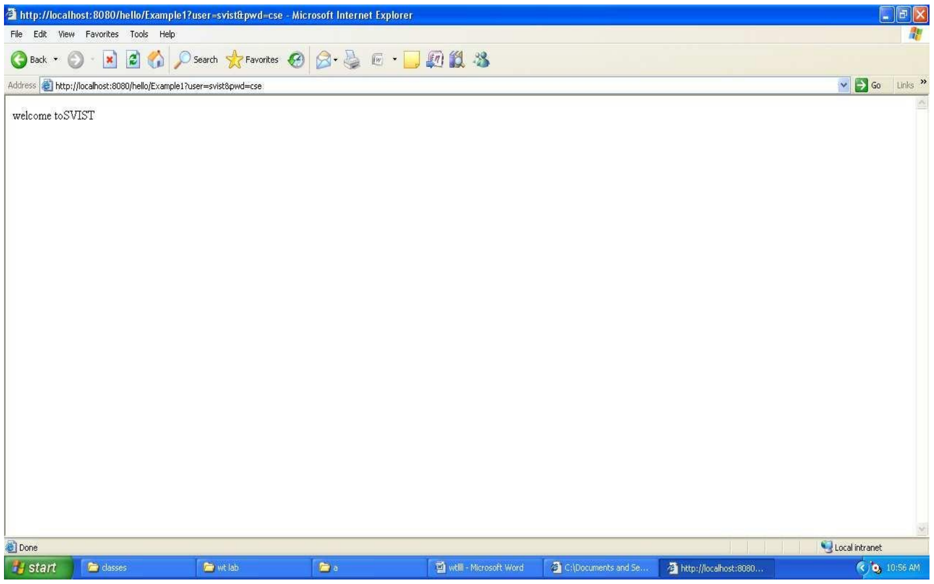
OUTPUT:


Authentication - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8080/hello/first.html

Username svist

password •••

submit


http://localhost:8080/hello/Example1?user=svist&pwd=cse - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8080/hello/Example1?user=svist&pwd=cse

welcome toSVIST

RESULT:

Thus the user authentication is carried out for four users by using both cookies andgetInitParameters successfully.

10.Extract data from the tables and display them in the catalogue page using JDBC.

AIM: Extract data from the tables and display them in the catalogue page using JDBC.

DESCRIPTION:

Create tables in the database which contain the details of items (books in our case like Book name, Price, Quantity, Amount)) of each category. Modify your catalogue page (week 2) in such a way that you should connect to the database and extract data from the tables and display them in the catalogue page using JDBC.

PROGRAM:

Retrieve.java:

```
import javax.servlet.*;
import
javax.servlet.http.*;
import java.sql.*;

import
java.io.*;
import
java.util.*;

public class Retrieve extends HttpServlet
{
public    void    service(HttpServletRequest req,HttpServletResponse res)
          throwsServletException,IOException
```

```
{
res.setContentType("text/htm
l"); PrintWriter
out=res.getWriter(); try{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@195.100.101.158:1521:cclab","sc
ott","tiger"); Statement s=con.createStatement();
ResultSet r=s.executeQuery("select * from cart");


out.println("<center> <table border=1>");
out.println("<thead> <th> Book name </th> <th> Price </th> <th> Quantity </th> <th> Amount </th>
</thead>");
while(r.next())
{
out.println("<tr> <td>
"+r.getString(1)+"</td> ");out.println("<td>
"+r.getString(2)+"</td> "); out.println("<td>
"+r.getInt(3)+"</td> "); out.println("<td>
"+r.getString(4)+"</td> </tr>");
}
out.println("</table></cente
r>"); con.close();
}
catch(SQLException sq)
```

```
  {
  out.println("sql exception"+sq);
  }
catch(ClassNotFoundException cl)
  {
  out.println("class not found"+cl);
  }
  }
}
```

web.xml:

```
  <web-app>
  <servlet>
  <servlet-name>set</servlet-name>
  <servlet-class>Cartenter</servlet-class>
  </servlet>
<servlet>
  <servlet-name>display</servlet-name>
  <servlet-class>Retrieve</servlet-class>
  </servlet>
  <servlet-mapping>
  <servlet-name>set</servlet-name>
  <url-pattern>/enterdata</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
```

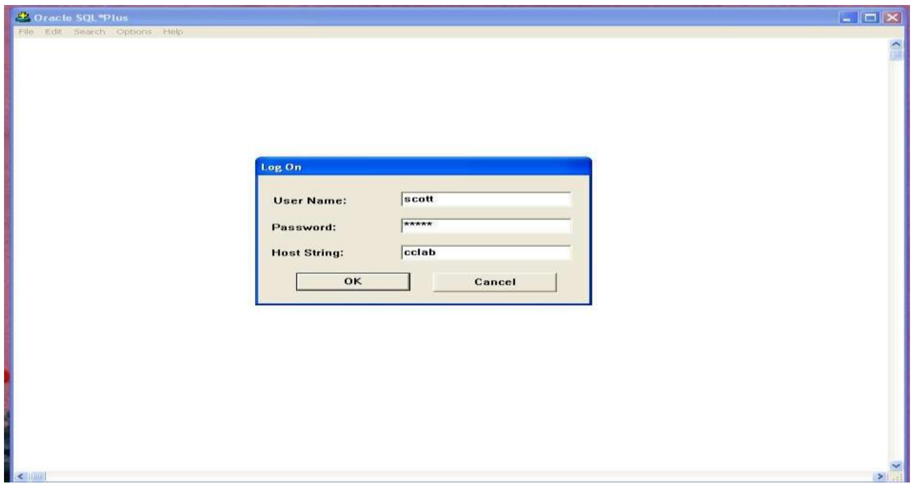
<servlet-name>display</servlet-name>

<url-pattern>/display1</url-pattern>

</servlet-mapping>

</web-app>

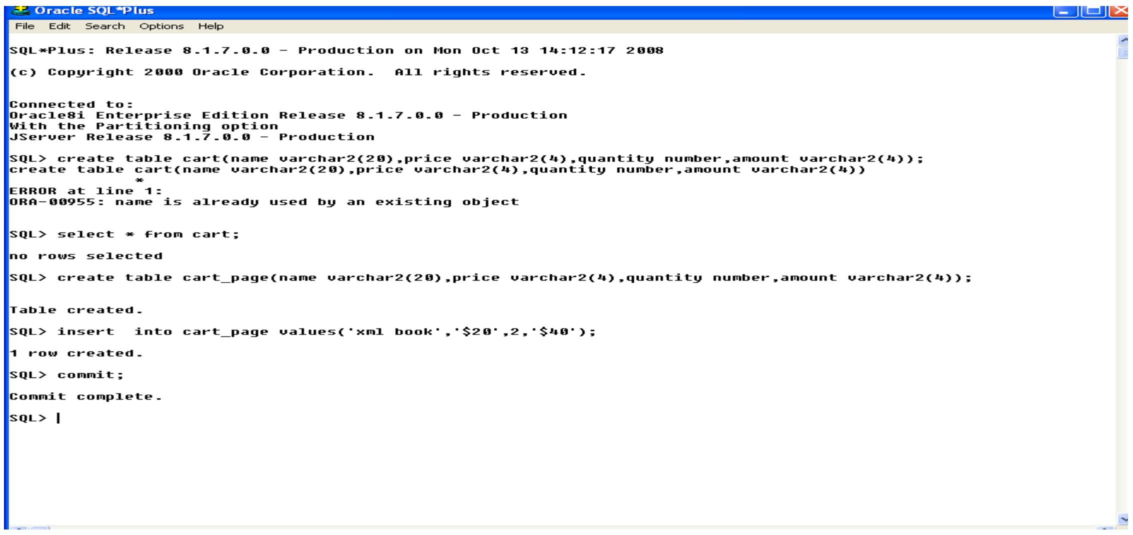
CREATE THE TABLE AND INSERT VALUES INTO THE TABLE:



User Name: scott

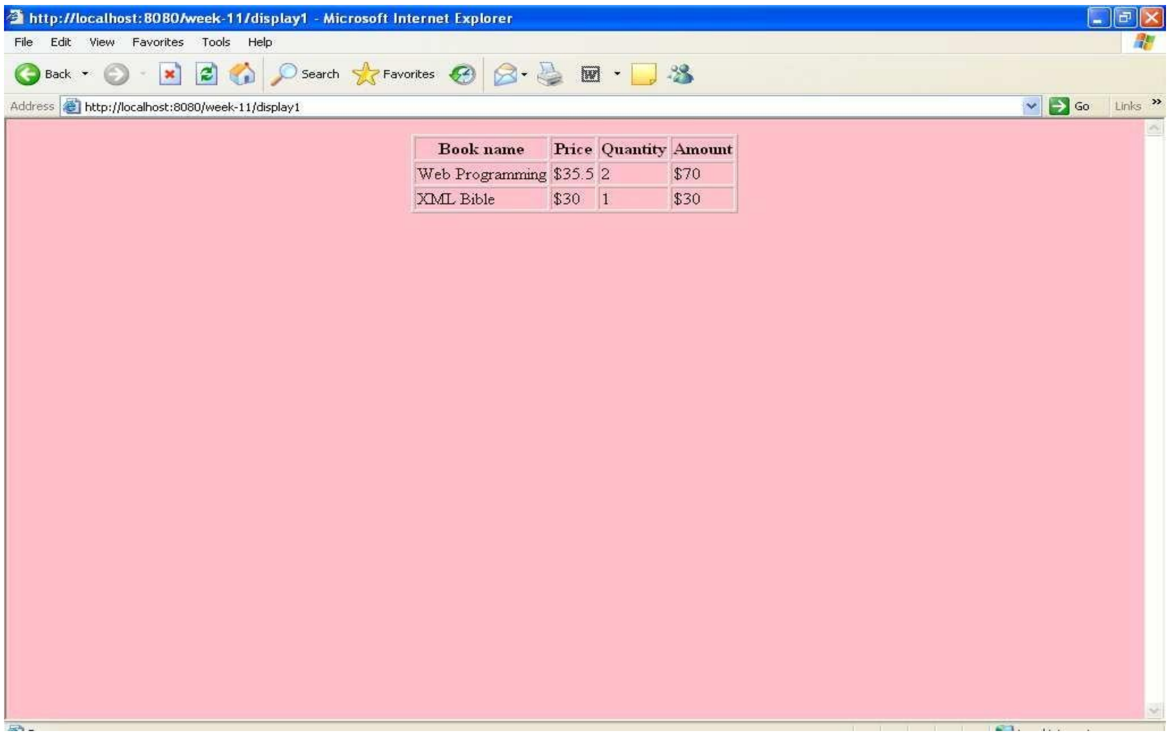
Password: *****

Host String: cclab



```
SQL*Plus: Release 8.1.7.0.0 - Production on Mon Oct 13 14:12:17 2008

(c) Copyright 2000 Oracle Corporation.  All rights reserved.

Connected to:
Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production

SQL> create table cart(name varchar2(20),price varchar2(4),quantity number,amount varchar2(4));
create table cart(name varchar2(20),price varchar2(4),quantity number,amount varchar2(4))
             *
ERROR at line 1:
ORA-00955: name is already used by an existing object


SQL> select * from cart;

no rows selected

SQL> create table cart_page(name varchar2(20),price varchar2(4),quantity number,amount varchar2(4));

Table created.

SQL> insert  into cart_page values('xml book','$20',2,'$40');

1 row created.

SQL> commit;

Commit complete.

SQL> |
```

OUTPUT:


http://localhost:8080/week-11/display1 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/week-11/display1
Go Links

Book name	Price	Quantity	Amount
Web Programming	$35.5	2	$70
XML Bible	$30	1	$30

RESULT:

The data is extracted from the tables and displayed in the catalogue page using JDBC

11.Write down the steps to create a Spring Boot App FirstProj. On running this app it will display “Welcome To Spring Boot” into the console.

Aim:

To create a Spring Boot App FirstProj. On running this app it will display “Welcome To Spring Boot” into the console.

Program:

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class FirstProjApplication {

	public static void main(String[] args) {
		SpringApplication.run(FirstProjApplication.class, args);
		System.out.println("Welcome to SpringBoot");
	}

}
```

Output:

```
Welcome To SpringBoot
```

12. Write down the steps to create a Spring Boot App Demo1.Here create a class Alien having 3 private member data aid(int),aname(String),tech(String) write down the steps to create all the getter and setters for these members. Now write a business method called show() in Alien class and within show method display a output “In Show”.Now in your main class under the main method create a object of Alien class through dependency injection.Now using this object set the Alien id as 1,name as “ABC” and technology as “JAVA”.Then using this object display the Alien id,name and tech by calling the getter method and at last called the show method. Use @Component annotation for Alien class.

Aim: steps to create a Spring Boot App Demo

Program:
Alien.java

```java
package com.example.demo;

import org.springframework.stereotype.Component;

@Component
public class Alien {
	private int aid;
	private String aname;
	private String tech;
	public int getAid() {
		return aid;
	}
	public void setAid(int aid) {
		this.aid = aid;
	}
	public String getAname() {
		return aname;
	}
	public void setAname(String aname) {
		this.aname = aname;
	}
	public String getTech() {
		return tech;
	}
	public void setTech(String tech) {
		this.tech = tech;
	}
	public void show()
	{
		System.out.println("In Show Function....");
	}

}
```

Demo1Application.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication
;
import
org.springframework.context.ConfigurableApplicationContext;

@SpringBootApplication
public class Demo1Application {

	public static void main(String[] args) {
		ConfigurableApplicationContext context=
SpringApplication.run(Demo1Application.class, args);
		Alien a=context.getBean(Alien.class);
		a.setAid(1);
		a.setAname("ABC");
		a.setTech("JAVA");
		System.out.println("Alien Id is:"+a.getAid());
		System.out.println("Alien Name is:"+a.getAname());
		System.out.println("Tech is:"+a.getTech());
		a.show(); }}
```

Output:

```
Alien Id is:1
Alien Name is:ABC
Tech is:JAVA
In Show Function....
         .
```

13. Write down steps to create a Spring Boot App Demo2. Create a singleton bean called Alien having 3 member aid, aname and tech. Write down the steps to create getters and setters. Create a zero argument constructor under Alien class and display “Object Created….” Under the constructor. Now under the main class the main method will have two object of Alien class through dependency injection. Write down on running the spring boot app how many times the “Object Created…” message will be displayed.(Use @Component annotation @Scope annotation in Alien class)

Aim: to create a Spring Boot App Demo2

Program:
Alien.java
```java
package com.example.demo;

import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

@Component
@Scope(value="singleton")
public class Alien {
	private int aid;
	private String aname;
	private String tech;

	public Alien() {
		super();
		System.out.println("Object Created...");
	}
	public int getAid() {
		return aid;
	}
	public void setAid(int aid) {
		this.aid = aid;
	}
	public String getAname() {
		return aname;
	}
	public void setAname(String aname) {
		this.aname = aname;
	}
	public String getTech() {
		return tech;
	}
	public void setTech(String tech) {
		this.tech = tech;
	}


}
```
Demo2Application.java
```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import 
org.springframework.boot.autoconfigure.SpringBootApplication
;
import 
org.springframework.context.ConfigurableApplicationContext;

@SpringBootApplication
public class Demo2Application {

	public static void main(String[] args) {
		ConfigurableApplicationContext 
context=SpringApplication.run(Demo2Application.class, args);
		Alien a1=context.getBean(Alien.class);
		
		Alien a2=context.getBean(Alien.class);
	}

}
```

Output:

```
Object Created...(Only One Time the message will be
displayed)
```

14. Write down steps to create a Spring Boot App Demo2. Create a prototype bean called Alien having 3 member aid, aname and tech. Write down the steps to create getters and setters. Create a zero argument constructor under Alien class and display "Object Created…." Under the constructor. Now under the main class the main method will have two object of Alien class through dependency injection. Write down on running the spring boot app how many times the "Object Created…" message will be displayed.(Use @Component annotation @Scope annotation in Alien class)

Aim: to create a Spring Boot App Demo2

Program:
Alien.java

```java
package com.example.demo;

import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

@Component
@Scope(value="prototype")
public class Alien {
	private int aid;
	private String aname;
	private String tech;

	public Alien() {
		super();
		System.out.println("Object Created...");
	}
	public int getAid() {
		return aid;
	}
	public void setAid(int aid) {
		this.aid = aid;
	}
	public String getAname() {
		return aname;
	}
	public void setAname(String aname) {
		this.aname = aname;
	}
	public String getTech() {
		return tech;
	}
	public void setTech(String tech) {
		this.tech = tech;
	}


}
```

Demo2Application.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication
;
import
org.springframework.context.ConfigurableApplicationContext;

@SpringBootApplication
public class Demo2Application {

	public static void main(String[] args) {
		ConfigurableApplicationContext
context=SpringApplication.run(Demo2Application.class, args);
		Alien a1=context.getBean(Alien.class);

		Alien a2=context.getBean(Alien.class);
	}

}
```

Output:

```
Object Created...
Object Created...
```

15. Write down steps to create a Spring Boot App Demo3. Create a bean called Laptop having 2 member lid(int), brand(String). Write down the steps to create getters and setters. Write down the steps to generate toString() method. Write down a method called compile() which will display “Compiling….”. Now create a bean Alien having aid,aname,tech and a Laptop variable laptop. Write down the steps to generate getters and setters for all the members. Write a method show() which will display “Within Show…” and using laptop object call the compile() method.
Here within the Alien class before private Laptop laptop; use @Autowired annotation and for both the Alien and Laptop class use @Component annotation. At last within the main class main method create Alien object through dependency injection and call the show() method by using Alien object. Which will display “Within Show…” and “Compilling….” message.

Aim: to create a Spring Boot App Demo3

Program:
Laptop.java
```
package com.example.demo;

import org.springframework.stereotype.Component;

@Component
public class Laptop {
	private int lid;
	private String brand;
	public int getLid() {
		return lid;
	}
	public void setLid(int lid) {
		this.lid = lid;
	}
	public String getBrand() {
		return brand;
	}
	public void setBrand(String brand) {
		this.brand = brand;
	}
	@Override
	public String toString() {
		return "Laptop [lid=" + lid + ", brand=" + brand + 
"]";
	}
	public void compile()
	{
		System.out.println("Compiling....");
	}

}
```
Alien.java
```
package com.example.demo;

import 
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
@Component
public class Alien {
	private int aid;
	private String aname;
	private String tech;
	@Autowired
	private Laptop laptop;
	public int getAid() {
		return aid;
	}
	public void setAid(int aid) {
		this.aid = aid;
	}
	public String getAname() {
		return aname;
	}
	public void setAname(String aname) {
		this.aname = aname;
	}
	public String getTech() {
		return tech;
	}
	public void setTech(String tech) {
		this.tech = tech;
	}
	public Laptop getLaptop() {
		return laptop;
	}
	public void setLaptop(Laptop laptop) {
		this.laptop = laptop;
	}
	public void show()
	{
		System.out.println("Within Show...");
		laptop.compile();
	}

}
```

Demo3Application.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication
;
import
org.springframework.context.ConfigurableApplicationContext;

@SpringBootApplication
public class Demo3Application {

	public static void main(String[] args) {
		ConfigurableApplicationContext
context=SpringApplication.run(Demo3Application.class, args);
		Alien obj=context.getBean(Alien.class);
		obj.show();
	}

}
```

Output:

```
Within Show...
Compiling....
```