WEEK - 6

1. Implement navigation using react router.

Components in React Router:

There are two types of router components:

- **<BrowserRouter>:** It is used for handling the dynamic URL.
- **<HashRouter>:** It is used for handling the static request.

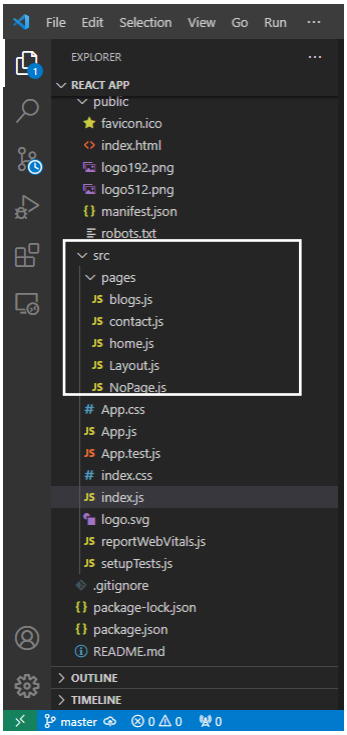
Step 1: Open the VS Code and launch the React on the local server.

- **Add React Router:**

To add React Router in your application, run this command **'npm i -D react-router-dom'** in the terminal from the root directory of the application. (Go to Terminal □ New Terminal □ Command Prompt).

Step 2: Open the folder in the VS Code and in the **'src'** folder, create a new folder named **'Pages'**.

Step 3: Now within the **'Pages'** folder, create the new files named **'blogs.js'**, **'contact.js'**, **'home.js'**, **'Layout.js'** & **'NoPage.js'** as shown below.



Step 4: Then add code inside the **'index.js'** file as shown here.

index.js

router > src > index.js > ...

```
import ReactDOM from "react-dom/client";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Layout from "./pages/Layout";
import Home from "./pages/home";
import Blogs from "./pages/blogs";
import Contact from "./pages/contact";
import NoPage from "./pages/NoPage";
export default function App() {
    return (
        <BrowserRouter>
            <Routes>
                <Route path="/" element={<Layout />}>
                    <Route index element={<Home />} />
                    <Route path="blogs" element={<Blogs />} />
                    <Route path="contact" element={<Contact />} />
                    <Route path="*" element={<NoPage />} />
                </Route>
            </Routes>
        </BrowserRouter>
    );
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<App />);
```

Step 5: Now add code to **'App.css'** file to style your application.

index.js | App.css

router > src > App.css > li a:hover:not(.active)

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #04AA6D;
}

li {
  float: left;
  border-right:1px solid #bbb;
}

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

li a:hover:not(.active) {
  background-color: #111;
}
```

Step 6: Write the below code to **'blogs.js'** file inside the **'Pages'** folder.

index.js | blogs.js

router > src > pages > blogs.js > default

```
const Blogs = () => {
    return <h1>Blog Articles</h1>
};
export default Blogs;
```

Step 7: Write the below code to **'contact.js'** file inside the **'Pages'** folder.

index.js | contact.js

router > src > pages > contact.js > default

```
const contact =() => {
    return <h1>contact me</h1>
};
export default contact;
```

Step 8: Write the below code to **'home.js'** file inside the **'Pages'** folder.

index.js | home.js

router > src > pages > home.js > default

```
const Home =() => {
    return <h1>Home</h1>
};
export default Home;
```

Step 9: Write the below code to **'Layout.js'** file inside the **'Pages'** folder.

index.js | Layout.js

router > src > pages > Layout.js > default

```
import { Outlet, Link } from "react-router-dom";
const Layout = () => {
    return (
        <>
        <nav>
            <ul>
                <li>
                    <Link to="/">Home</Link>
                </li>
                <li>
                <Link to="/blogs">Blogs</Link>
                </li>
                <li>
                <Link to="/contact">contact</Link>
                </li>
            </ul>
        </nav>
        <Outlet />
        </>
    )
    };

    export default Layout;
```

Step 10: Write the below code to **'NoPages.js'** file inside the **'Pages'** folder.

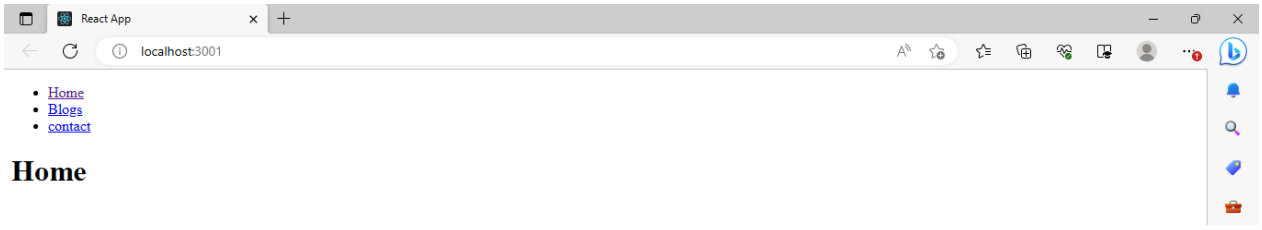
index.js | NoPage.js

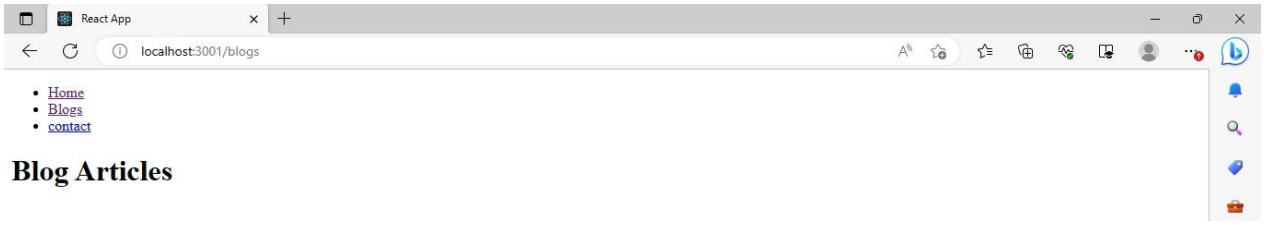
router > src > pages > NoPage.js > default

```
const NoPage =() => {
    return <h1>404</h1>;
};
export default NoPage;
```

Now run the **'index.js'** file to view the final output.

Output:





2. React Hooks-demonstrating useState, useEffect, useContext, useReducer.

React Hooks: Hooks (Receive or Steal) are the new feature introduced in the React 16.8 version. It allows you to use state and other React features without writing a class.

- **Types of Hooks in React:**

1) **UseState** - The React useState Hook allows us to track state in a function component. State generally refers to data or properties that need to be tracking in an application. To use the useState Hook, we first need to import it into our component.

   Syntax: import { useState } from "react";

2) **UseEffect** - The useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, directly updating the DOM, and timers. useEffect accepts two arguments. The second argument is optional.

   Syntax: useEffect(<function>, <dependency>)

3) **UseContext** - React Context is a way to manage state globally. It can be used together with the useState Hook to share state between deeply nested components more easily than with useState alone. To create context, you must Import createContext and initialize it.

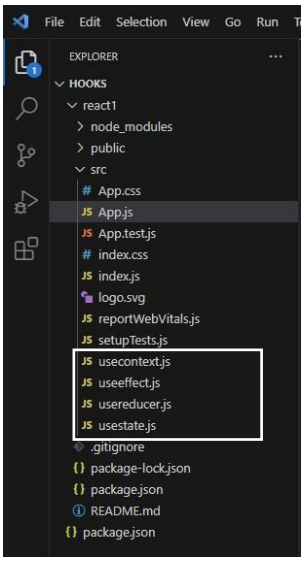
   Syntax: const myValue = useContext(MyContext);

4) **UseReducer -** The useReducer Hook is similar to the useState Hook. It allows for custom state logic.The useReducer Hook accepts two arguments.

   Syntax: useReducer(<reducer>, <initialState>)

Step 1: Open the VS Code and launch the React on the local server.

Step 2: To run the React Hooks, create the files (**usestate.js, useeffect.js, usecontext.js, usereducer.js**) as shown below in the **'src'** folder.



- **Example of UseState:**

Step 3: Write the below code in the **'App.js'** file inside the **'src'** folder.

Step 4: Then add code in **'usestate.js'** file within the **'src'** folder.

Output:

My name is VVP_CS_FSD

Click me

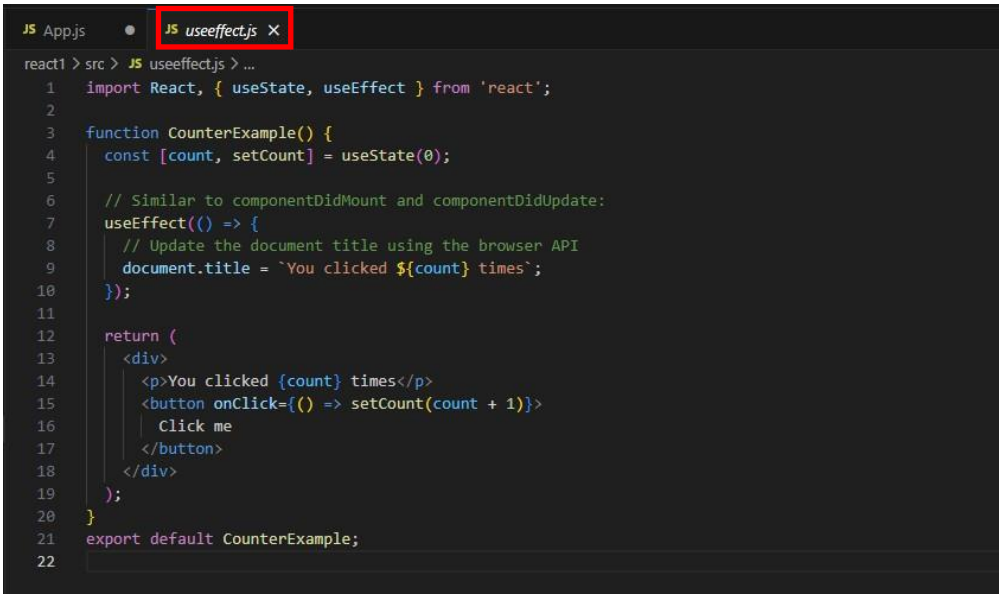
My name is Full stack development

Click me

- **Example of UseEffect:**

Step 5: Write the below code in the **'App.js'** file inside the **'src'** folder.

Step 6: Then add code in **'useeffect.js'** file within the **'src'** folder.



Output:

You clicked 0 times

Click me

You clicked 6 times

Click me

- **Example of UseContext:**

Step 7: Write the below code in the **'App.js'** file inside the **'src'** folder.

App.js
```
import Component1 from './usecontext';

function App() {
  return (
    <div>
    <Component1/>
    </div>
  );
}

export default App
```

Step 8: Then add code in the **'usecontext.js'** file within the **'src'** folder.

usecontext.js
```
import { useState, createContext, useContext } from "react";

const UserContext = createContext();

function Component1() {
  const [user, setUser] = useState("FSD");

  return (
    <UserContext.Provider value={user}>
      <h1>Hello {user}!</h1>
      <Component2 />
    </UserContext.Provider>
  );
}

function Component2() {
  return (
    <>
      <h1>Component 2</h1>
      <Component3 />
    </>
  );
}

function Component3() {
  return (
    <>
      <h1>Component 3</h1>
      <Component4 />
    </>
  );
}

function Component4() {
  return (
    <>
      <h1>Component 4</h1>
      <Component5 />
    </>
```

```
<Component5 />
</>
);
}

function Component5() {
  const user = useContext(UserContext);

  return (
    <>
      <h1>Component 5</h1>
      <h2>Hello {user} again!</h2>
    </>
  );
}

/*const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Component1 />);*/
export default Component1;
```

Output:

Hello FSD!

Component 2

Component 3

Component 4

Component 5

Hello FSD again!

- **Example of UseReducer:**

Step 9: Write the below code in the **'App.js'** file inside the **'src'** folder.

```
import AppReducer from './usereducer';

function App() {
  return (
    <div>

    <AppReducer/>
    </div>
  );
}

export default App;
```

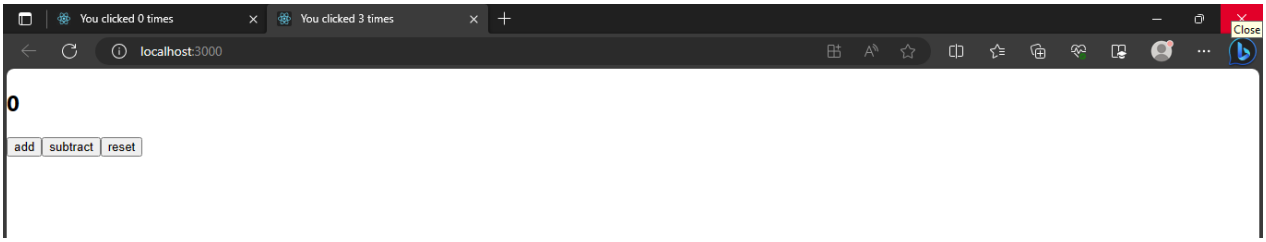
Step 10: Then add code in the **'usereducer.js'** file within the **'src'** folder.

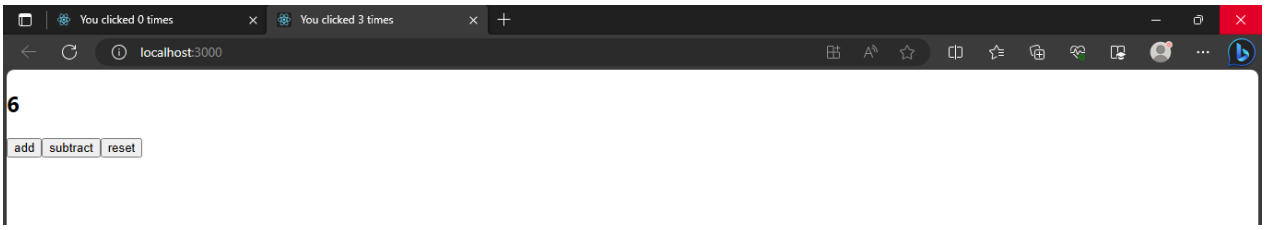
App.js | usereducer.js

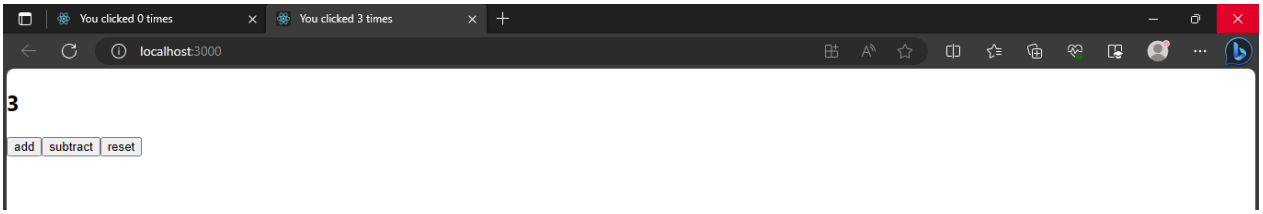
react1 > src > usereducer.js > ...

```
import React, { useReducer } from "react";

// Defining the initial state and the reducer
const initialState = 0;
const reducer = (state, action) => {
  switch (action) {
    case "add":
      return state + 1;
    case "subtract":
      return state - 1;
    case "reset":
      return 0;
    default:
      throw new Error("Unexpected action");
  }
};

const AppReducer = () => {
    // Initialising useReducer hook
  const [count, dispatch] = useReducer(reducer, initialState);
  return (
    <div>
      <h2>{count}</h2>
      <button onClick={() => dispatch("add")}>
        add
      </button>
      <button onClick={() => dispatch("subtract")}>
        subtract
      </button>
      <button onClick={() => dispatch("reset")}>
        reset
      </button>
    </div>
  );
};
export default AppReducer;

```

Output:



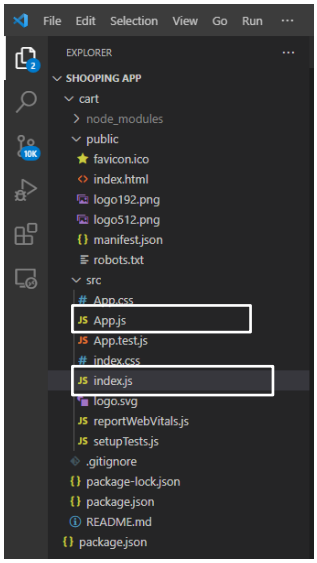




3. Build single page application - Shopping cart.

Step 1: Open the VS Code and launch the React on the local server.

- **React folder structure:**



Step 2: Add code to **'index.js'** file in the **'src'** folder.

App.js index.js

cart > src > index.js > ...

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App/>
  </React.StrictMode>
);
```

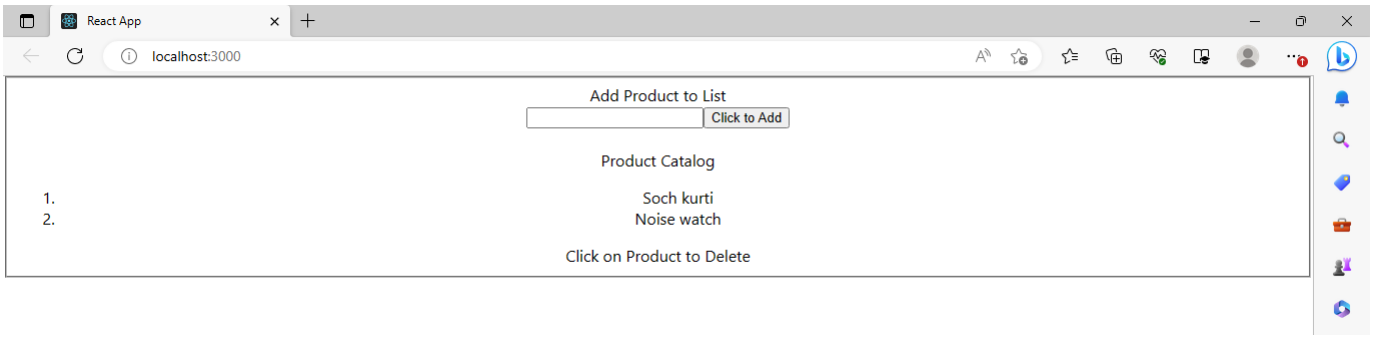
Step 3: Then write the below code in the **'App.js'** within the **'src'** folder.

App.js index.js

cart > src > App.js > ...

```
import { useState } from "react";
function App() {
  const [list, setList] = useState([]);
  const [value, setValue] = useState("");
  const addToList = () => {
    let tempArr = list;
    tempArr.push(value);
    setList(tempArr);
    setValue("");
  };
  const deleteItem = (index) => {
    let temp = list.filter((item, i) => i !== index);
    setList(temp);
  };
  return (
    <div className="App">
      <fieldset>
        <h>Add Product to List</h><br></br>
        <input type="text" value={value} onChange={(e) => setValue(e.target.value)} />
        <button onClick={addToList}> Click to Add </button><br></br><br></br>
        <h>Product Catalog</h><br></br>
        <ol>
          {list.map((item, i) => <li onClick={() => deleteItem(i)}>{item} </li>)}
        </ol>
        <h>Click on Product to Delete</h><br></br>
      </fieldset></div>
  );
} export default App;
```

Output:

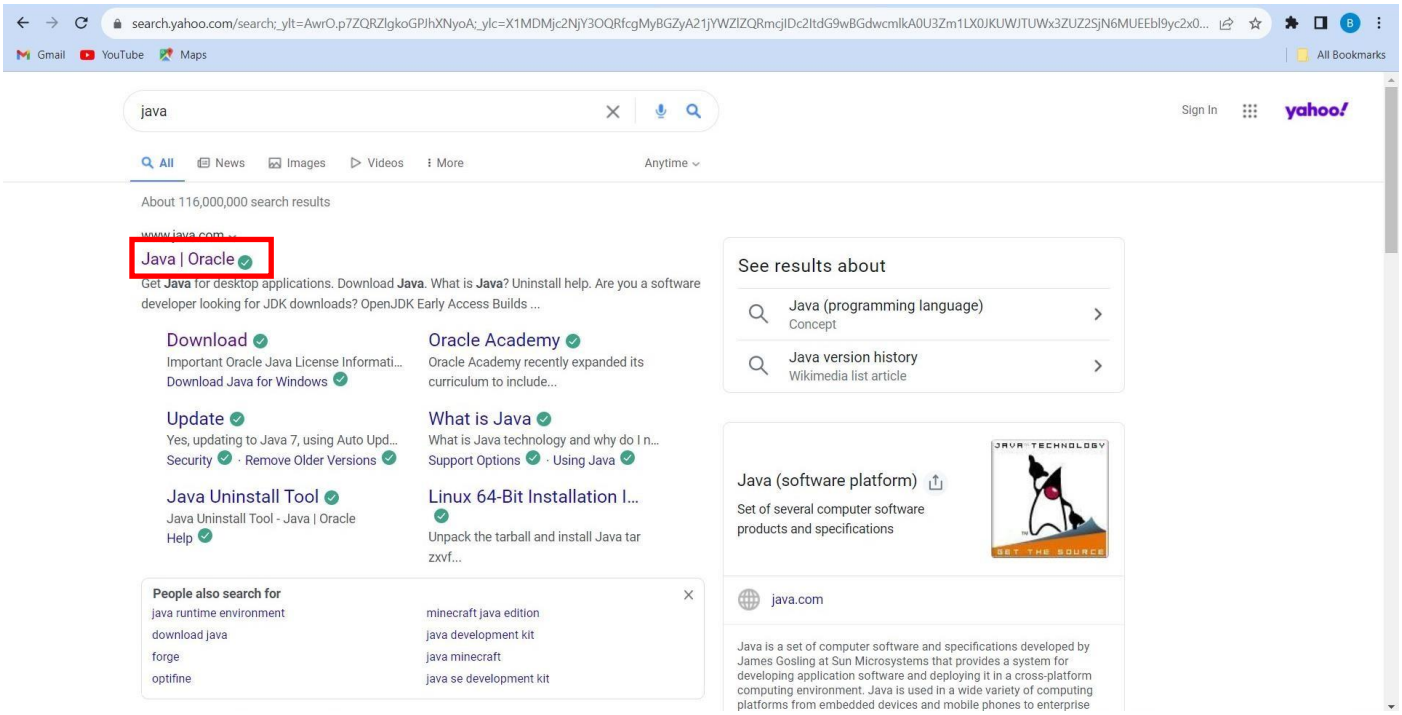


4. Setting up the environment and tools to install Java (latest stable version) and add environment variable to it.

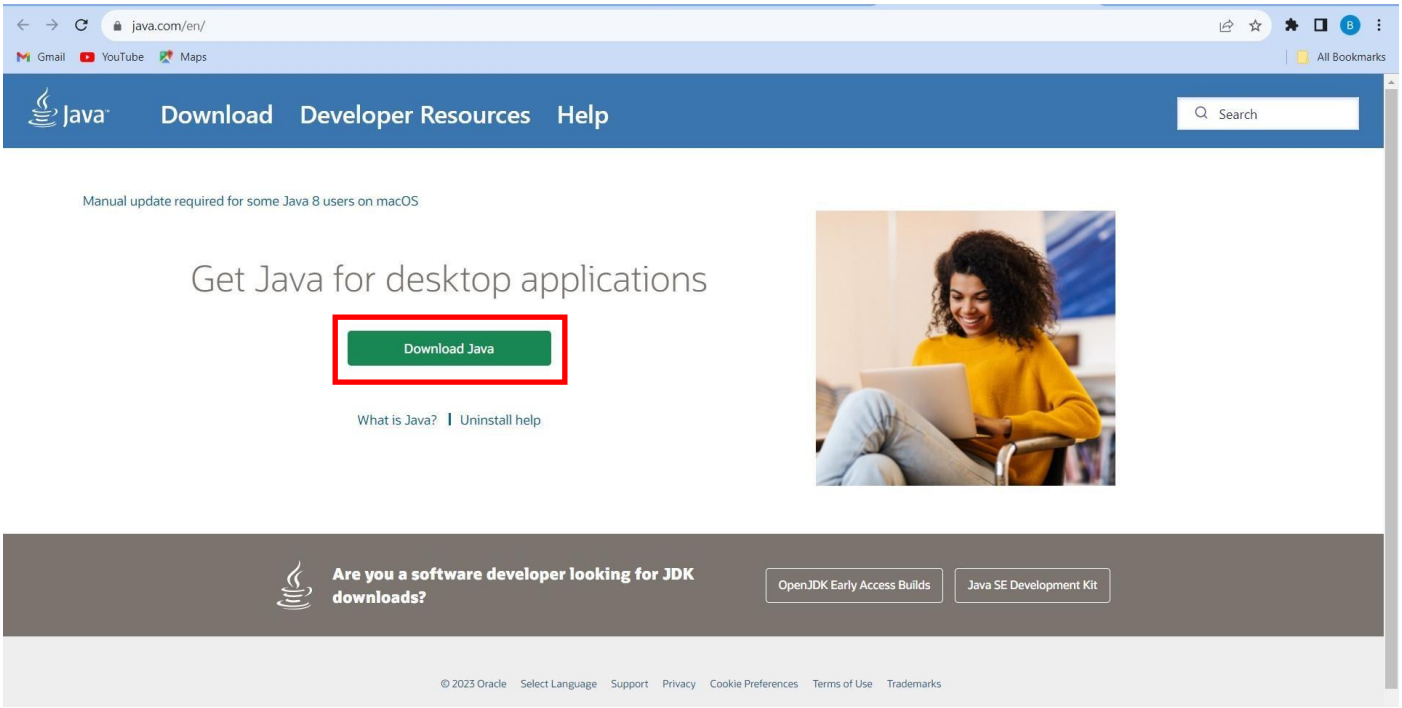
JAVA: Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let programmers *write once, run anywhere* (WORA), meaning that compiled Java code can run on all platforms that support Java without the need to recompile.

Installation of Java on Windows:

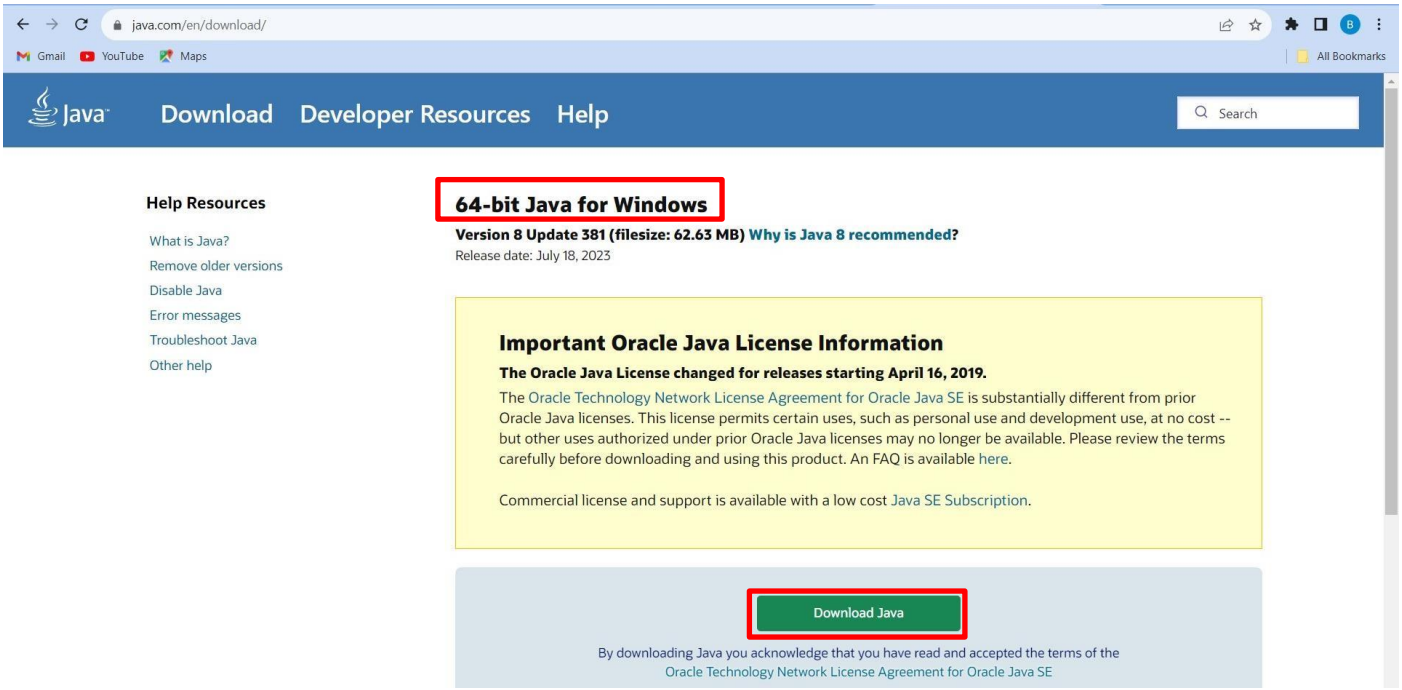
Step 1: Go to Web Browser → Search for Java → Click on the first link here.



Step 2: Now click on **'Download Java'**.



Step 3: Make sure to download the stable version for Windows. Here **64-bit** system type is recommended for download. Now click on **'Download Java'**.



Step 4: Open the downloaded file & click on **'Install'**.

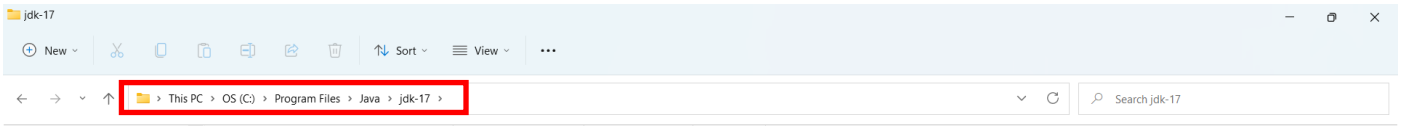
Step 5: Wait until the installation is completed

Step 6: Click on **'close'**.

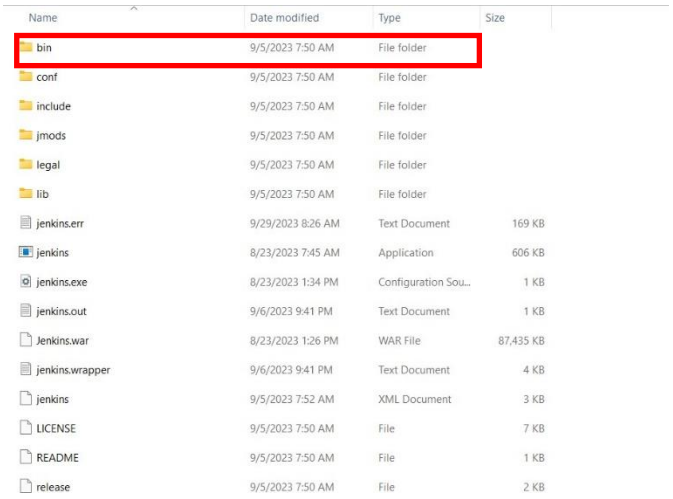
- **<u>Setting up of Environment Variables for Java.</u>**

Note: To run Java program, make sure to setup the Environment variables for Java on your windows system.

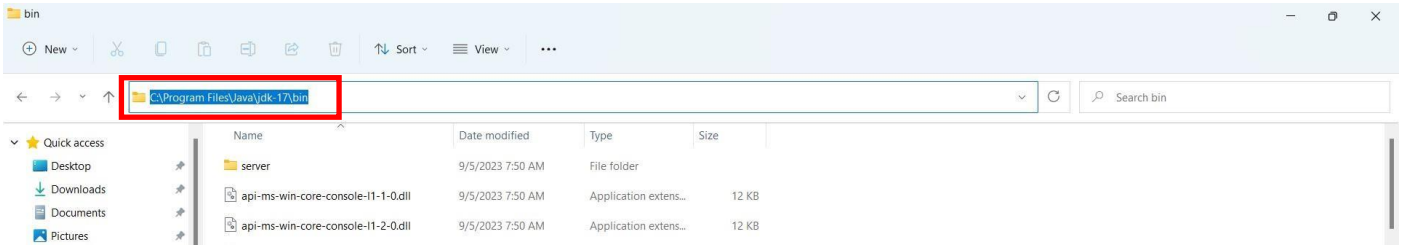
Step 1: After the installation of Java, go to This PC → C drive → Program files → Java → jdk17 as shown below.



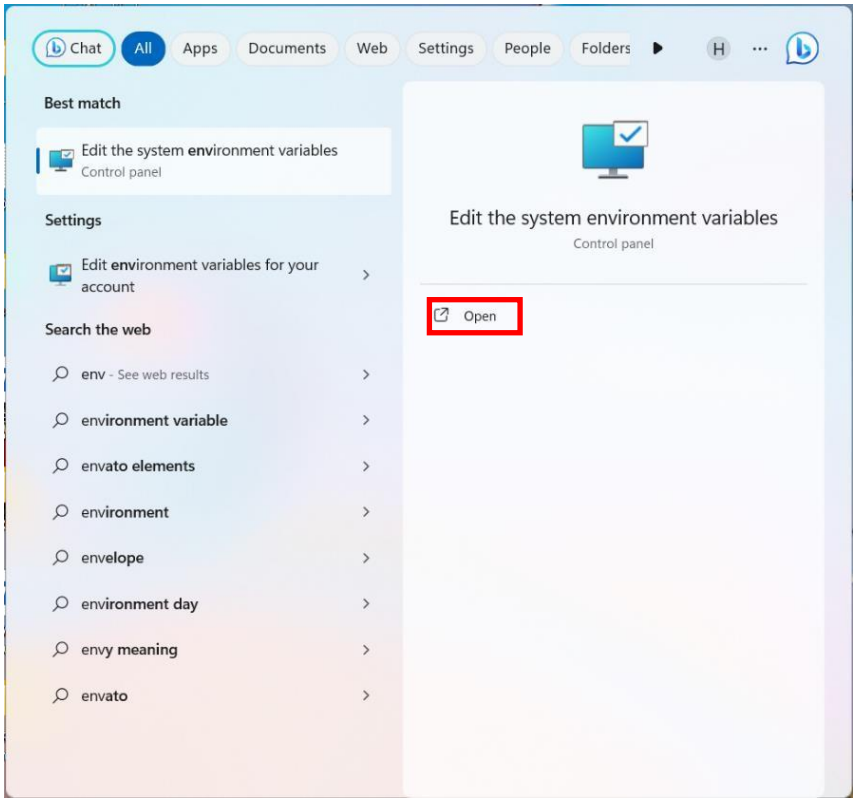
Step 2: Now click on **'bin'** folder.



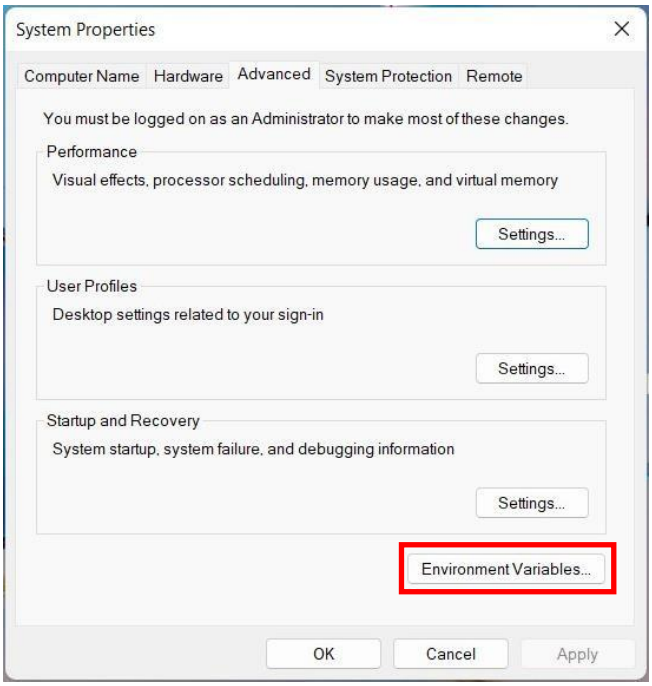
Step 3: Then copy the copy the bin location here.



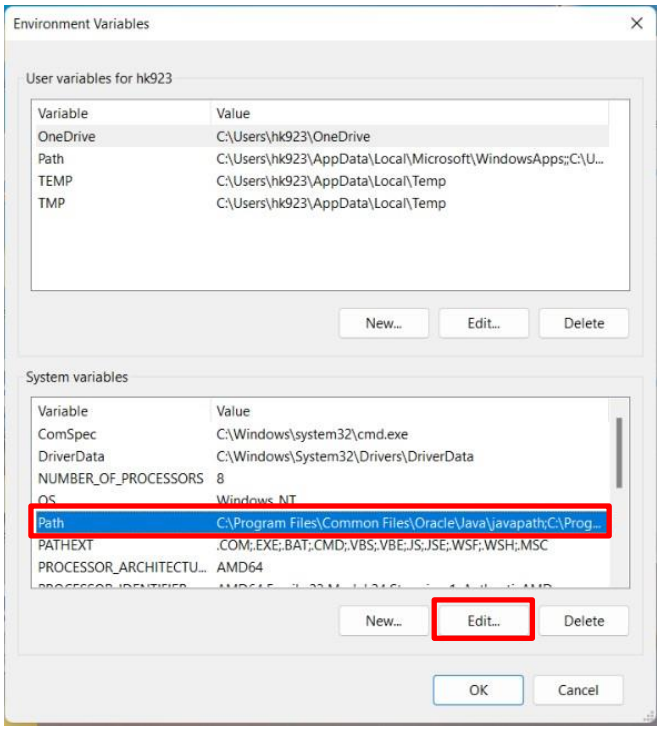
Step 4: Go to windows search bar Search for **'Environment Variables'** & click on **Open**



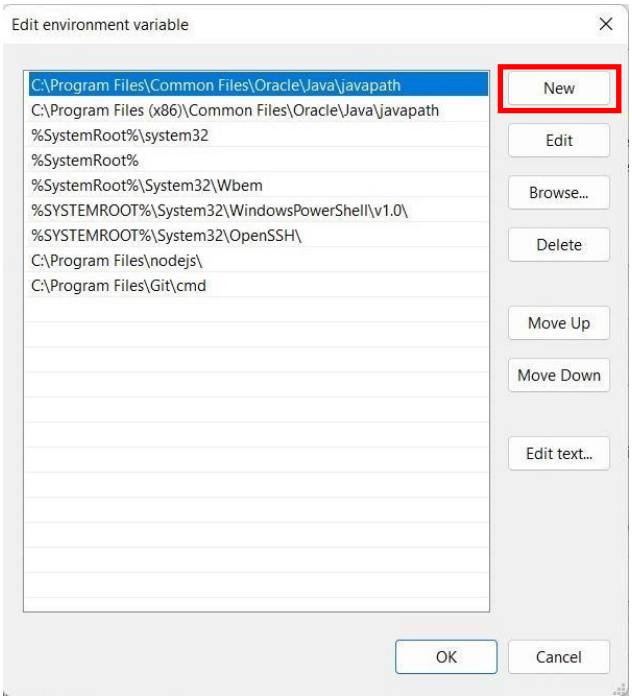
Step 5: Now click on **'Environment Variables'** here.



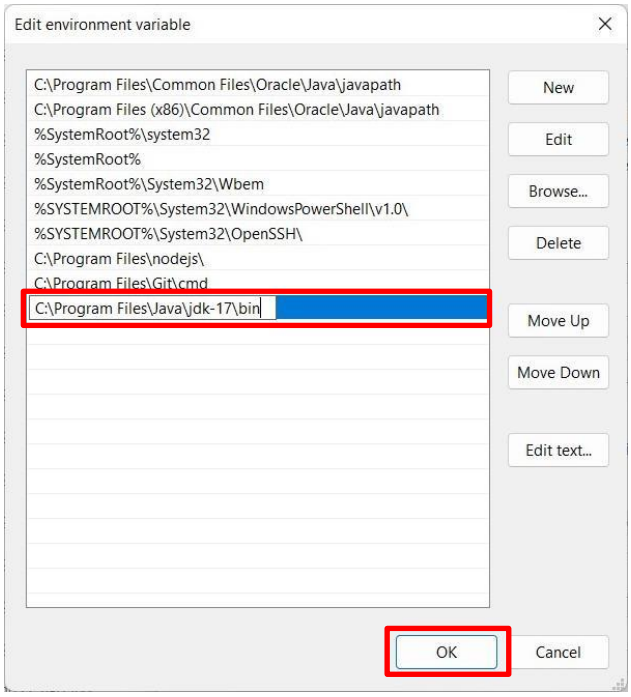
Step 6: Under System Variables, select Path location & click on **'Edit'**.



Step 7: Click on **'New'**.



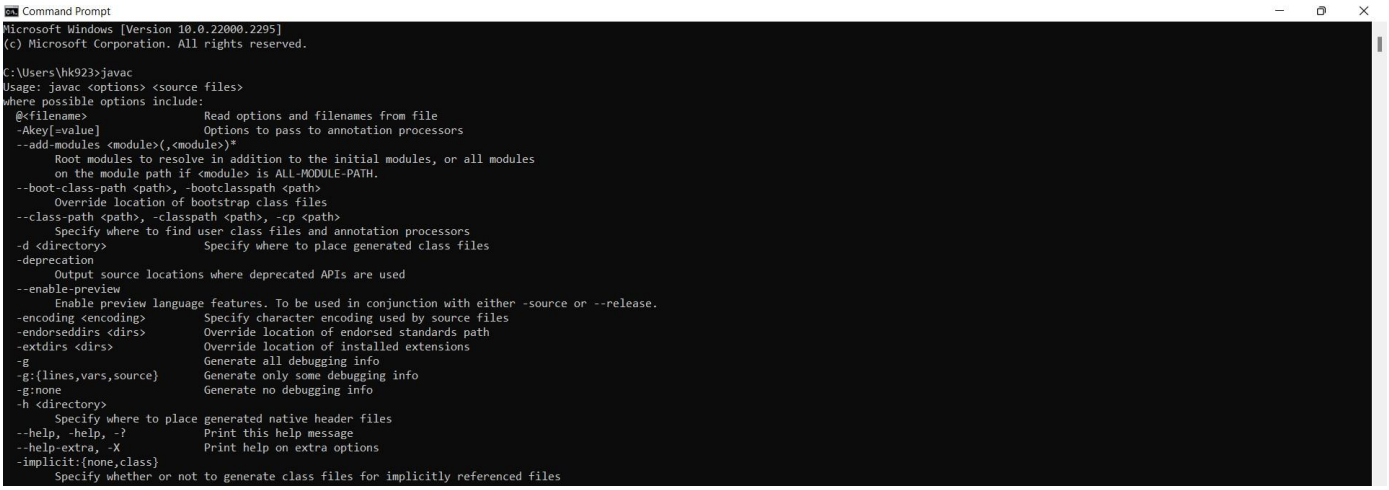
Step 8: Now paste the bin location which was copied in step 3.



Step 9: Use Command Prompt to check whether the Java Program is working or not.

Give the command as **'java -version'** to check the Version and other details as shown below.

Step 10: Now again enter the command as **'javac'** to check the working of Java compiler on your Windows.

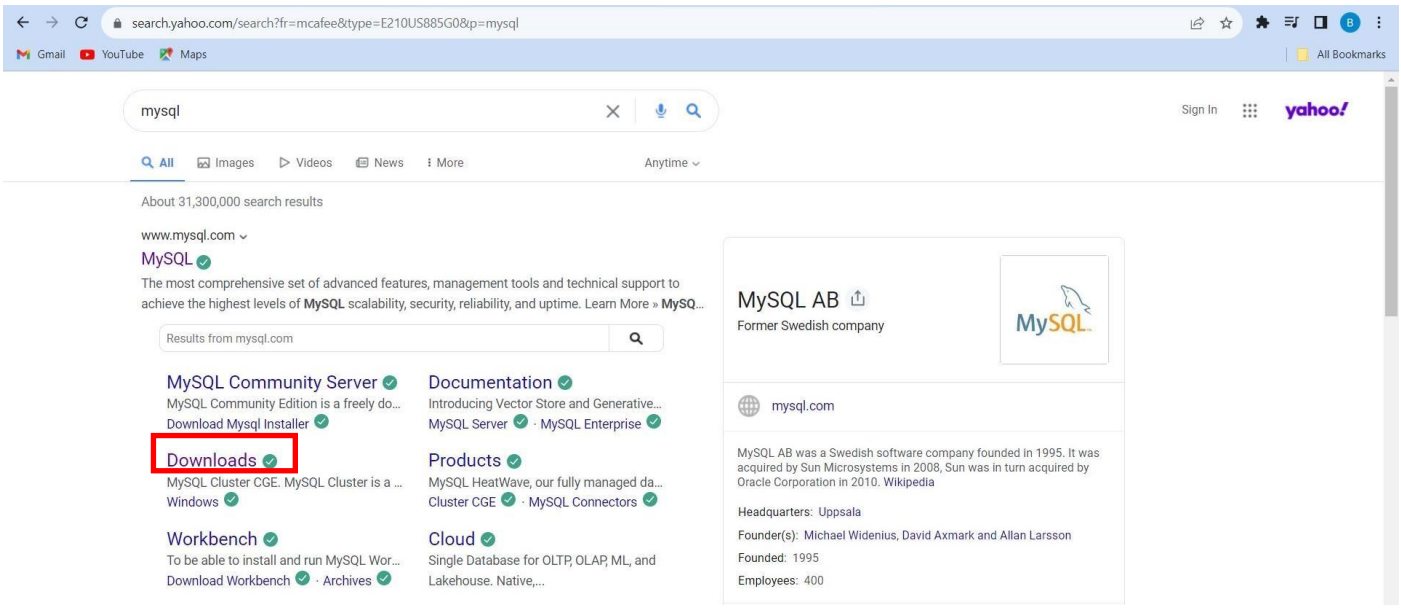


5. Install DBMS (MySQL, PostgreSQL or any other) tools.

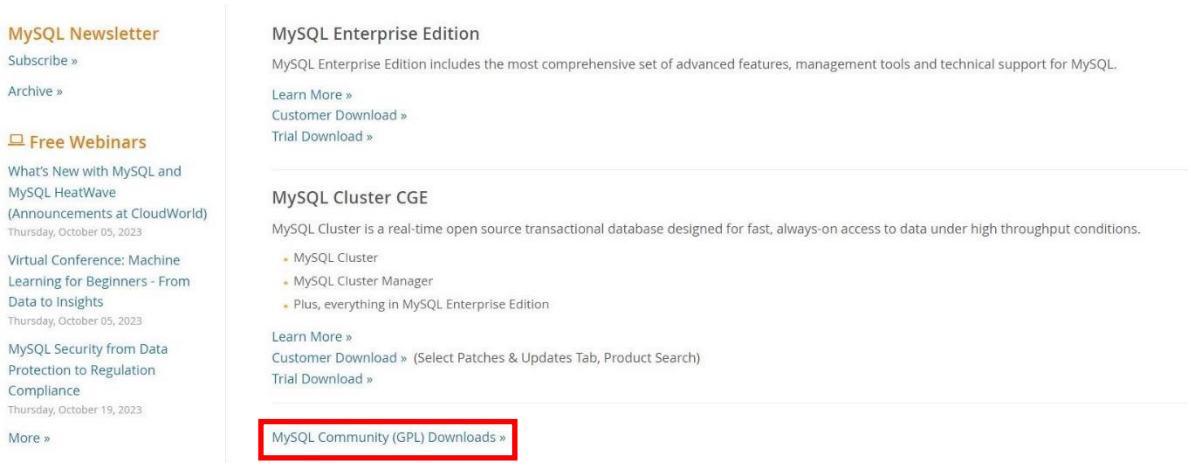
MySQL: **MySQL** is an open-source relational database management system (RDBMS). SQL is a language that programmers use to create, modify and extract data from the relational database, as well as control user access to the database. It manages the users, allows for network access and facilitates testing database integrity and creation of backups.

- **Installation procedures for MySQL:**

Step 1: Go to Web browser Search for MySQL Click on **'Downloads'**.

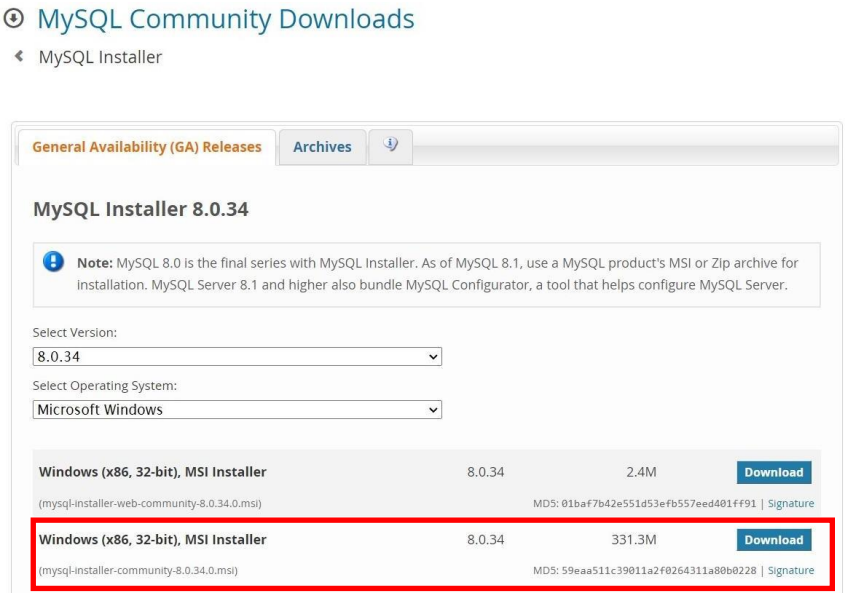


Step 2: Click on **'MySQL Community (GPL) Downloads'**.

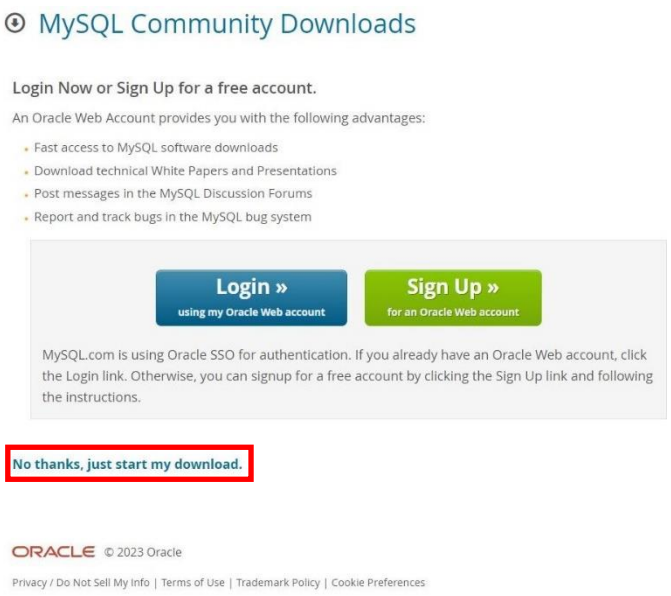


Step 3: Now click on **'MySQL Installer for Windows'**.

Step 4: Then select the second option & click on **'Download'**.



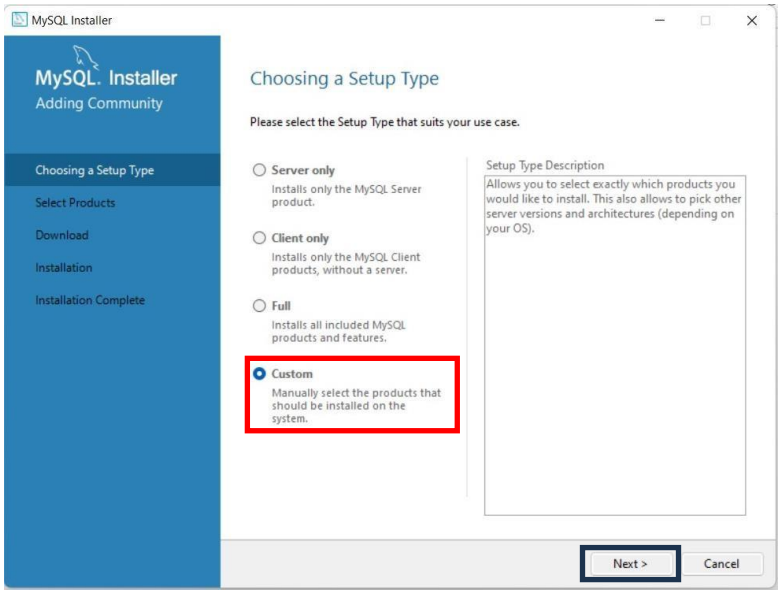
Step 5: Now you can Login or Sign up with your Oracle account or simply click on **'No thanks, just start my download'**.



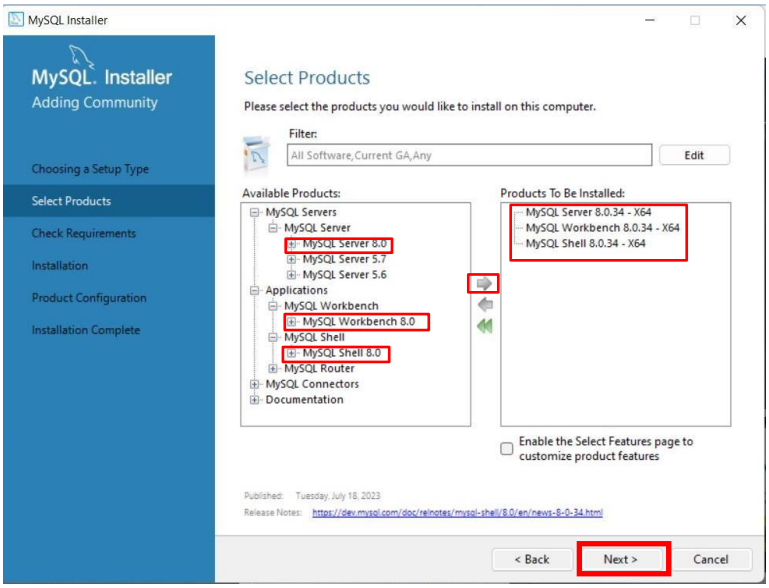
Step 6: Validating the internet connections.

Installing the MySQL on both client & server side interfaces.

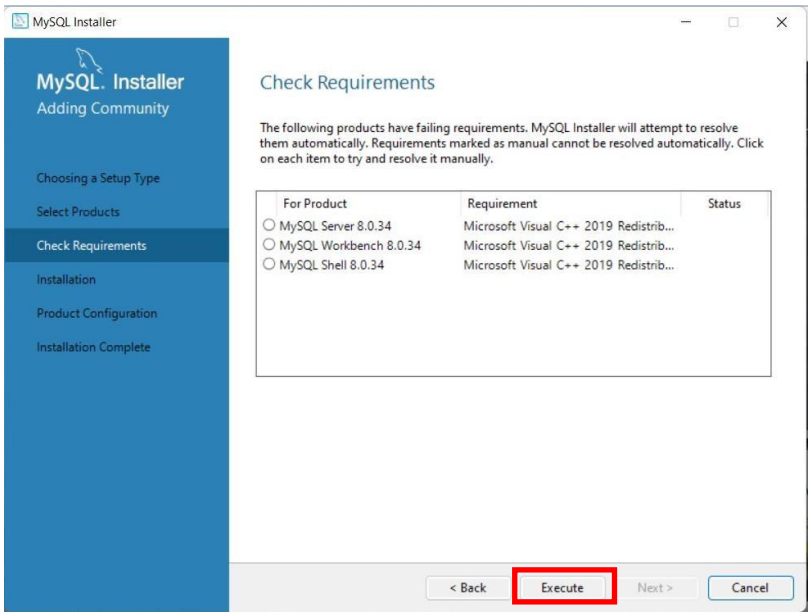
Step 7: Select **'Custom'** & click on **'Next'**.



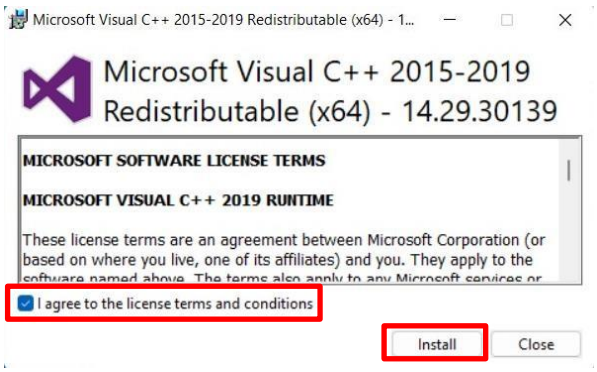
Step 8: Select the MySQL products from **Available products** to **Products to be installed** stage & click on **'Next'**.



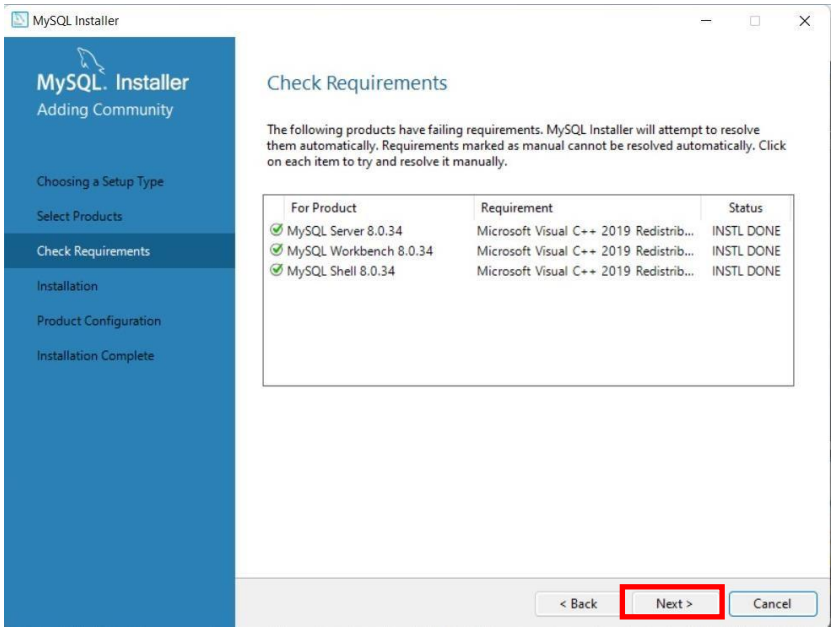
Step 9: Click on **'Execute'**.



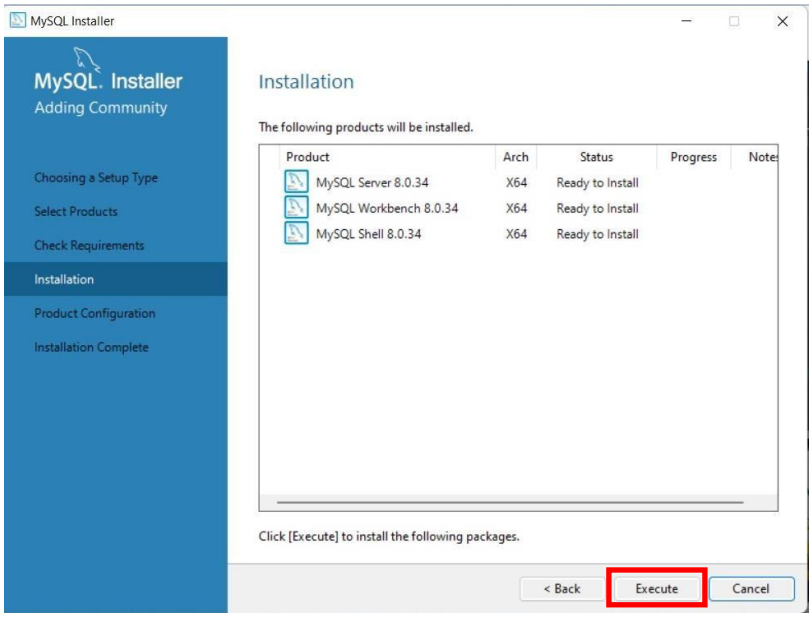
Step 10: Enable the option below & click on **'Install'**.



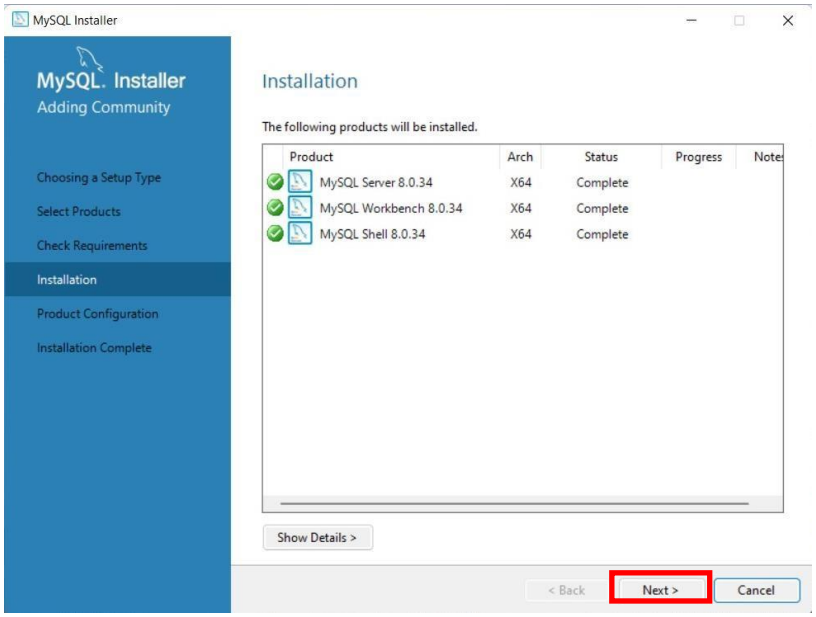
Step 11: After the installation of the products are complete, click on **'Next'**.



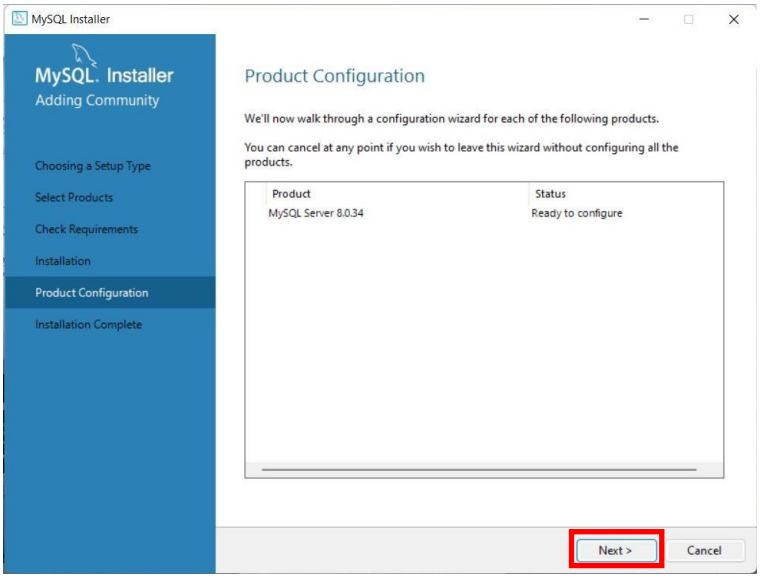
Step 12: Click on **'Execute'**.



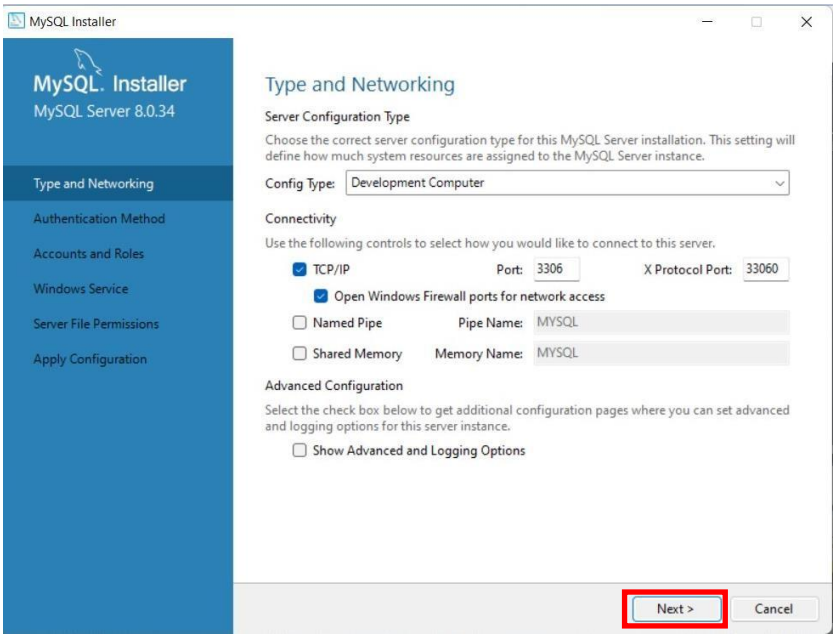
Step 13: After the complete execution, click on **'Next'**.



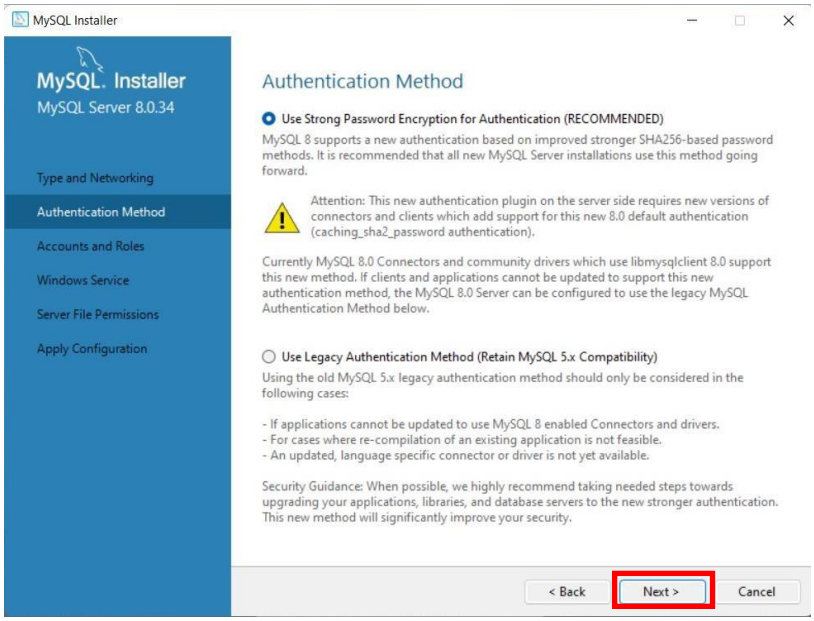
Step 14: Click on **'Next'**.



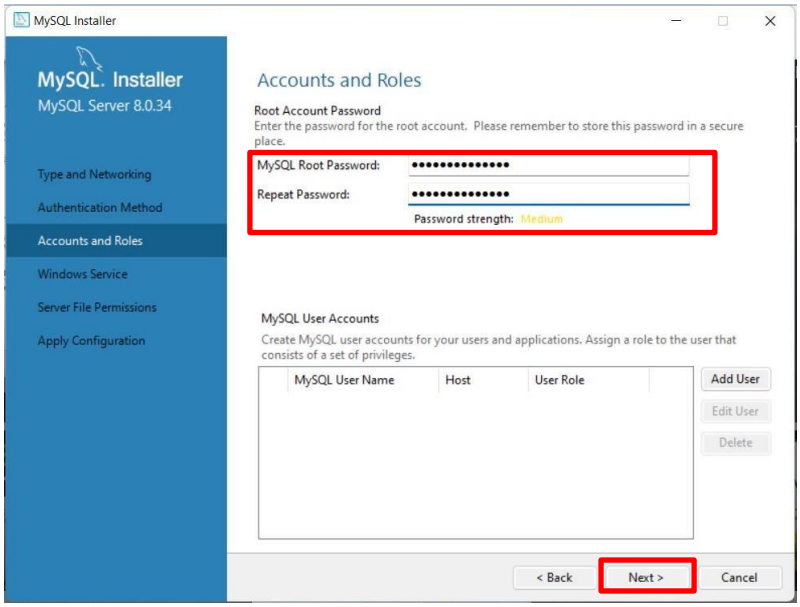
Step 15: Don't change the default selections, simply click on **'Next'**.



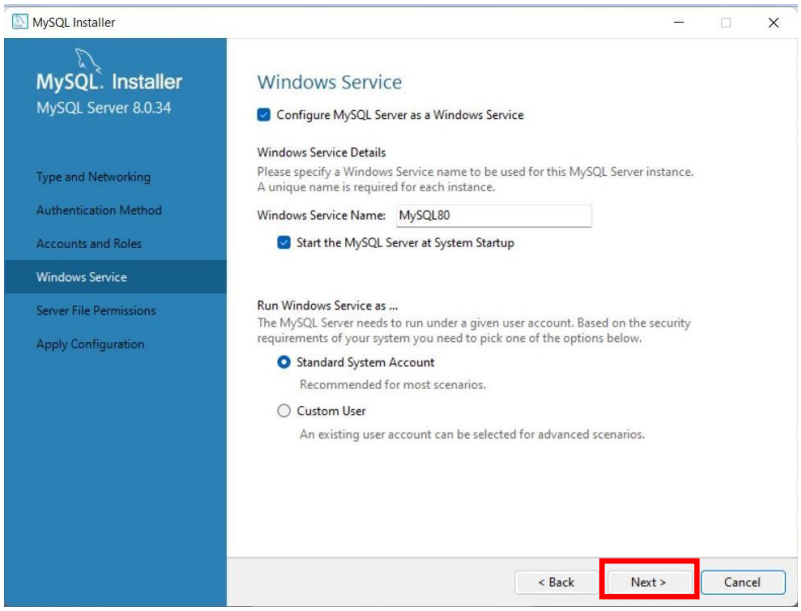
Step 16: Click on **'Next'**.



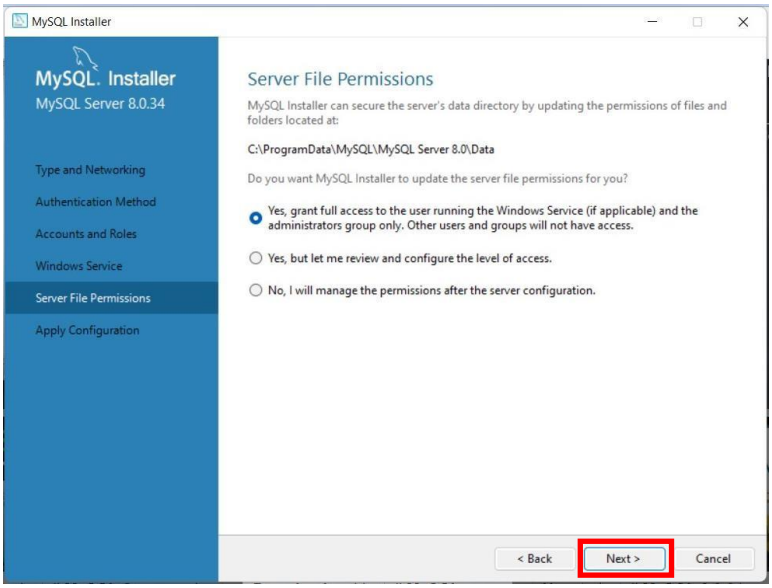
Step 17: Create a strong password for MySQL server & click on **'Next'.**



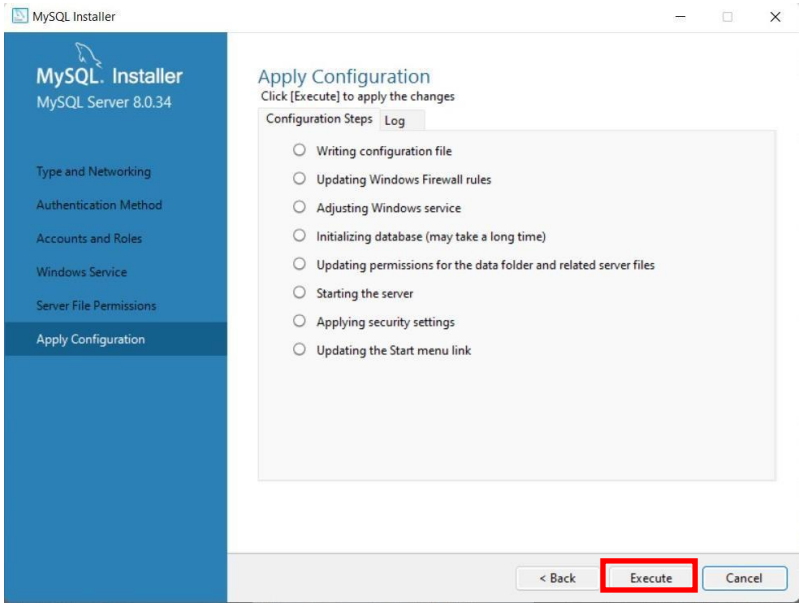
Step 18: Then click on **'Next'**.



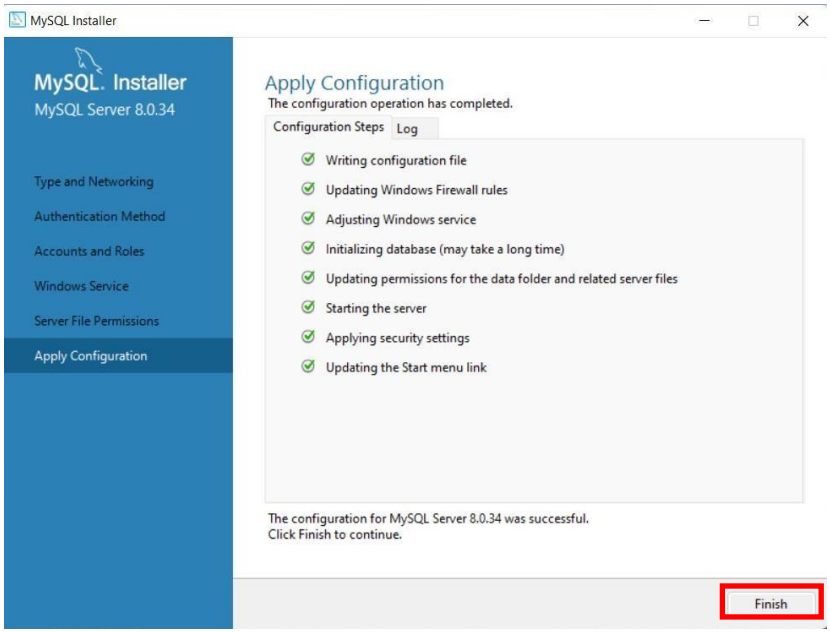
Step 19: Now click on **'Next'**.



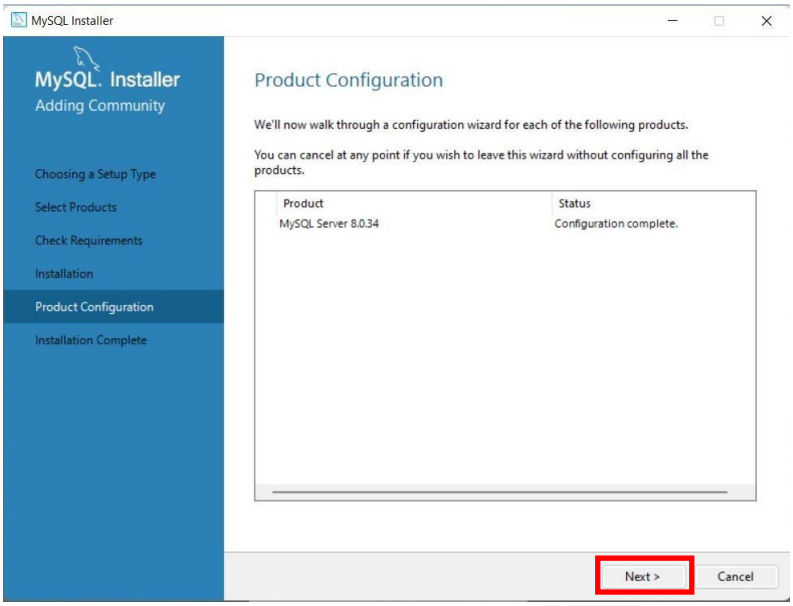
Step 20: Click on **'Execute'**.



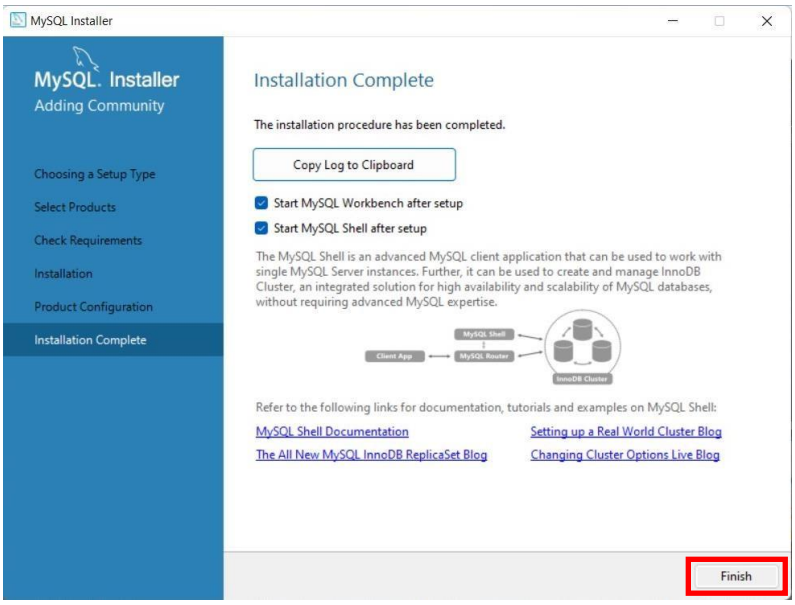
Step 21: Simply click on **'Finish'**.



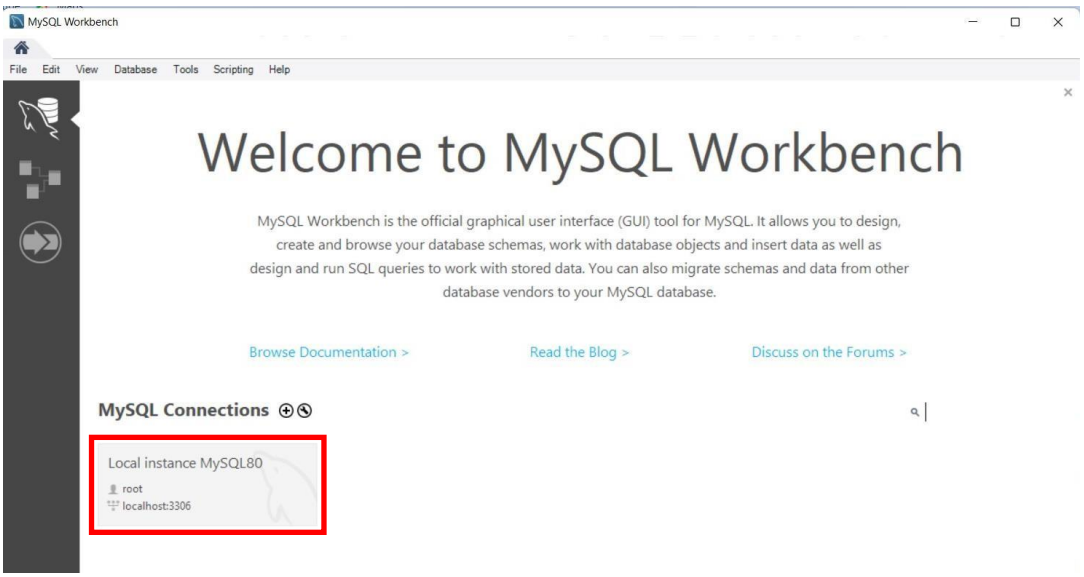
Step 22: Click on **'Next'**.



Step 23: Click on **'Finish'**.

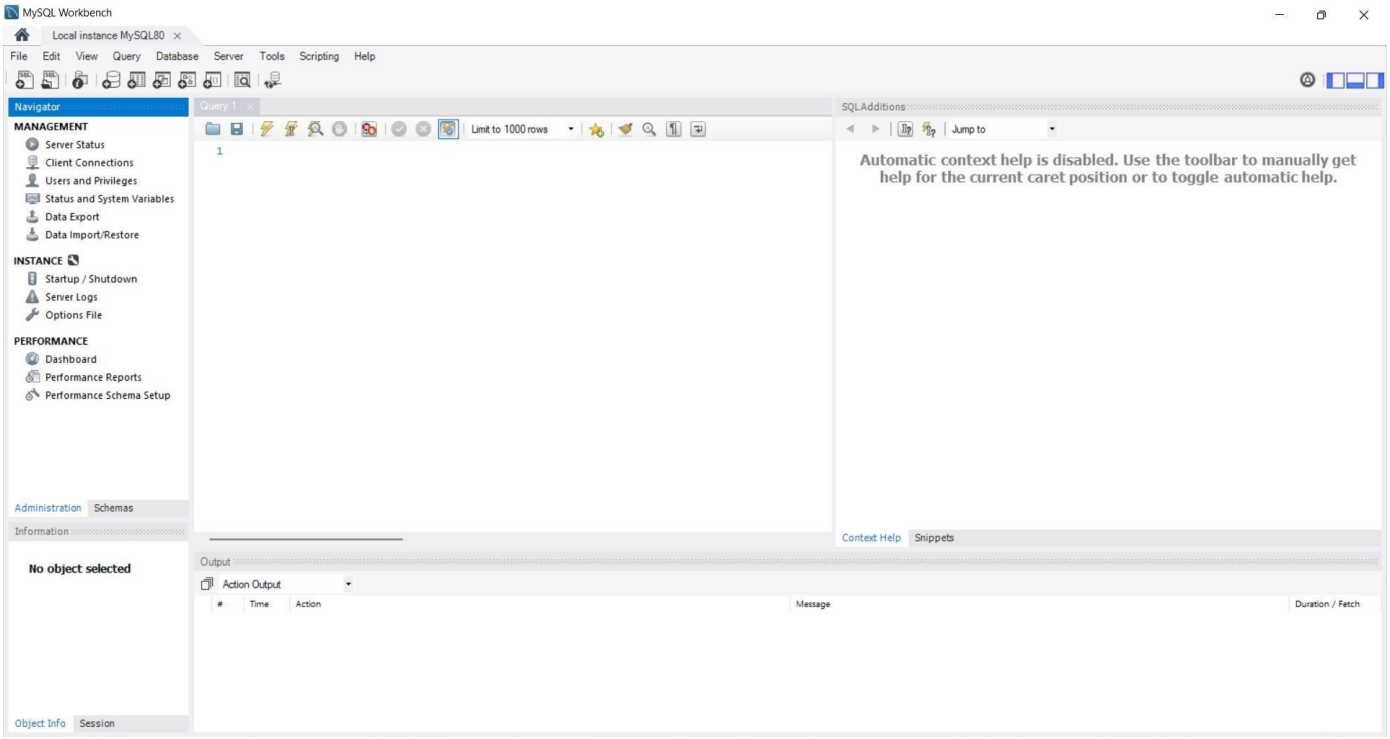


Step 24: Now click on **'Local instance MySQLBO'** here.



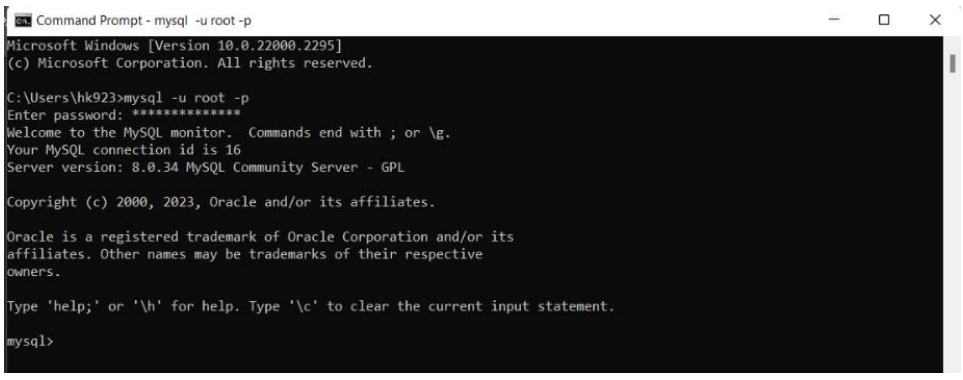
Step 25: Now enter your MySQL Server password here & click on **'OK'**.

Step 26: Here is your MySQL Workbench is ready to use.



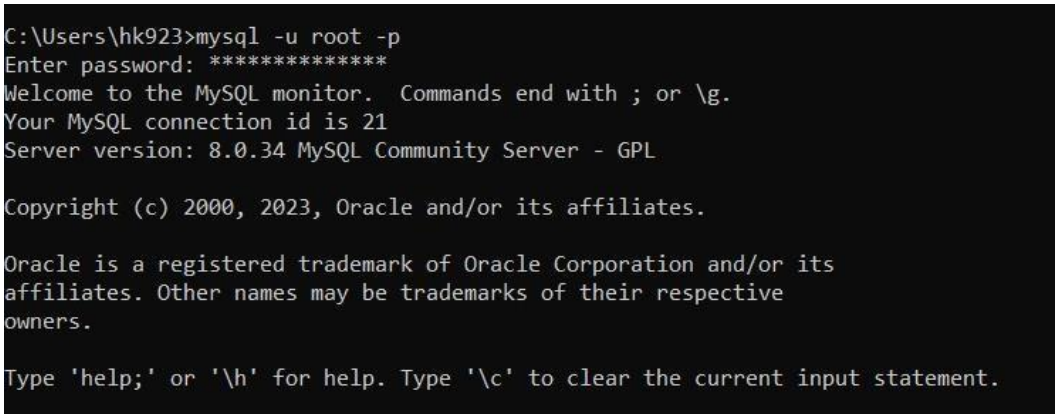
To run MySQL server:

Setup the Environment Variables for MySQL server & enter the command as **'mysql -u root -p'**. Also enter your password to use the MySQL Monitor.

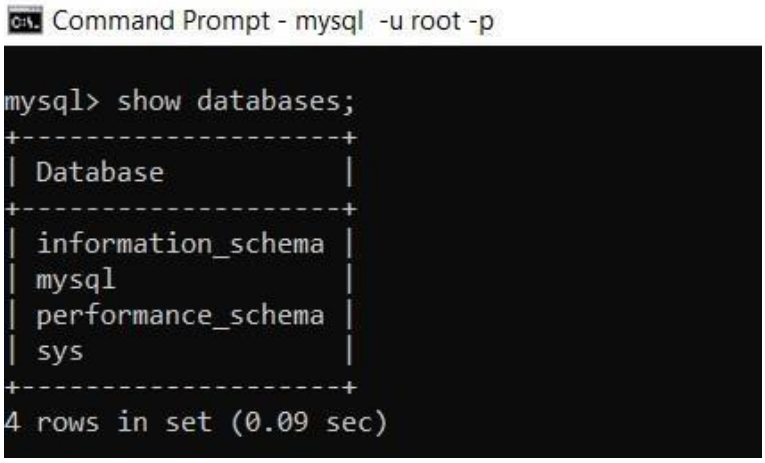


Step 1: Go to command Prompt → give the command as **'mysql –version'** → to check the current version of MySQL.

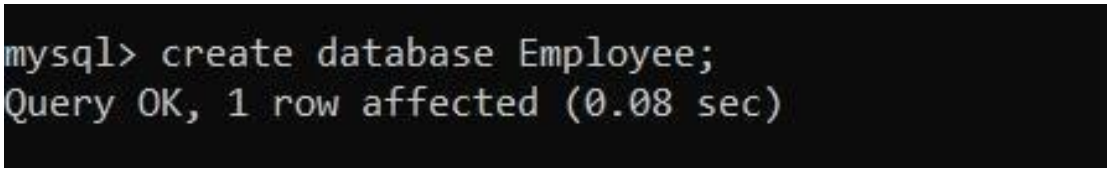
Step 2: Now give the command as **'mysql -u root -p'** & enter the password to enter into the MySQL server.



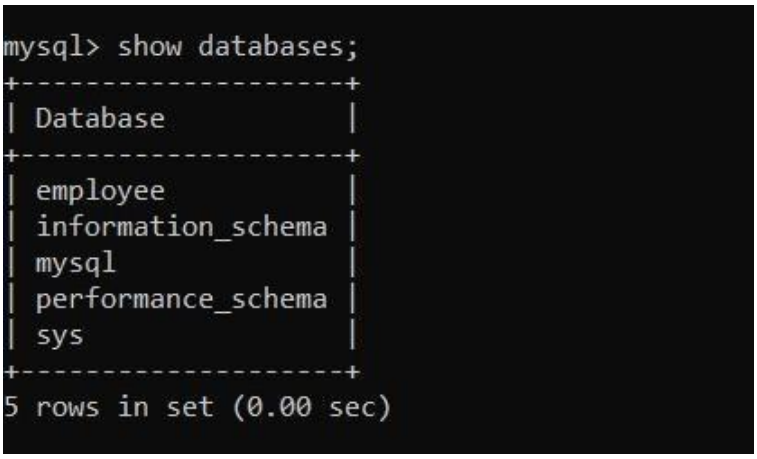
Step 3: Then enter **'show databases;'** to view the embedded databases.



Step 4: Enter the command **'create database Employee'** to create a new database.



Step 5: Again enter the command 'show databases' to view the created databases as shown below.


mysql> show databases;
+--------------------+
| Database |
+--------------------+
| employee |
| information_schema |
| mysql |
| performance_schema |
| sys |
+--------------------+
5 rows in set (0.00 sec)