Assignment 1

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

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>eCommerce Admin Dashboard</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" />
  <style>
    body {
      font-family: 'Segoe UI', sans-serif;
      background-color: #f8f9fa;
    }
    .sidebar {
      height: 100vh;
      background-color: #72899f;
      color: white;
    }
    .sidebar a {
      color: white;
      text-decoration: none;
      display: block;
      padding: 10px 20px;
    }
    .sidebar a:hover {
      background-color: #343a40;
    }
```

```
    .card-icon {
      font-size: 1.5rem;
    }
    .banner-img {
      width: 100%;
      height: 200px;
      object-fit: cover;
      border-radius: 10px;
      margin-bottom: 20px;
    }
    .product-img {
      height: 150px;
      object-fit: cover;
      border-radius: 10px;
    }
  </style>
</head>
<body>
<div class="container-fluid">
  <div class="row">
    <nav class="col-md-2 d-none d-md-block sidebar">
      <div class="p-3">
        <h4 class="text-white">eCommerce</h4>
        <hr>
        <a href="#">Dashboard</a>
        <a href="#">Products</a>
        <a href="#">Orders</a>
        <a href="#">Customers</a>
        <a href="#">Reports</a>
        <a href="#">Settings</a>
      </div>
    </nav>
```

```
<main class="col-md-10 ms-sm-auto col-lg-10 px-md-4">
 <div class="pt-3 pb-2 mb-3 border-bottom">
  <h2>Admin Dashboard</h2>
 </div>
 <img src="Banner.jpg" alt="Banner" class="banner-img" />
 <div class="row">
  <div class="col-md-3 mb-4">
   <div class="card text-white bg-primary">
    <div class="card-body d-flex justify-content-between">
     <div>
      <h5 class="card-title">Total Sales</h5>
      <p class="card-text fs-4">₹45,320</p>
     </div>
     <div class="card-icon">💰</div>
    </div>
   </div>
  </div>
  <div class="col-md-3 mb-4">
   <div class="card text-white bg-success">
    <div class="card-body d-flex justify-content-between">
     <div>
      <h5 class="card-title">Orders</h5>
      <p class="card-text fs-4">1,280</p>
     </div>
     <div class="card-icon">📦</div>
    </div>
   </div>
  </div>
  <div class="col-md-3 mb-4">
   <div class="card text-white bg-warning">
    <div class="card-body d-flex justify-content-between">
     <div>
```

```
        <h5 class="card-title">Customers</h5>
        <p class="card-text fs-4">860</p>
       </div>
       <div class="card-icon">👥</div>
     </div>
    </div>
   </div>
   <div class="col-md-3 mb-4">
    <div class="card text-white bg-danger">
     <div class="card-body d-flex justify-content-between">
      <div>
       <h5 class="card-title">Revenue</h5>
       <p class="card-text fs-4">₹1.2L</p>
      </div>
      <div class="card-icon">📈</div>
     </div>
    </div>
   </div>
  </div>
  <div class="card mb-4">
   <div class="card-body">
    <h5 class="card-title mb-3">Top Products</h5>
    <div class="row g-3">
     <div class="col-md-3">
      <div class="card h-100">
       <img src="Watch.jpg" class="card-img-top product-img" alt="Product 1">
       <div class="card-body">
        <h6 class="card-title">Smartwatch</h6>
        <p class="card-text">₹5,999</p>
       </div>
      </div>
     </div>
```

```
        <div class="col-md-3">
          <div class="card h-100">
            <img src="HeadPhones.jpg" class="card-img-top product-img" alt="Product 2">
            <div class="card-body">
              <h6 class="card-title">Headphones</h6>
              <p class="card-text">₹3,499</p>
            </div>
          </div>
        </div>
        <div class="col-md-3">
          <div class="card h-100">
            <img src="Sneakers" class="card-img-top product-img" alt="Product 3">
            <div class="card-body">
              <h6 class="card-title">Sneakers</h6>
              <p class="card-text">₹4,299</p>
            </div>
          </div>
        </div>
        <div class="col-md-3">
          <div class="card h-100">
            <img src="Bag.jpg" class="card-img-top product-img" alt="Product 4">
            <div class="card-body">
              <h6 class="card-title">Backpack</h6>
              <p class="card-text">₹2,199</p>
            </div>  </div>  </div>  </div>  </div>  </div>
    </main>
  </div>
</div>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

Assignment 1

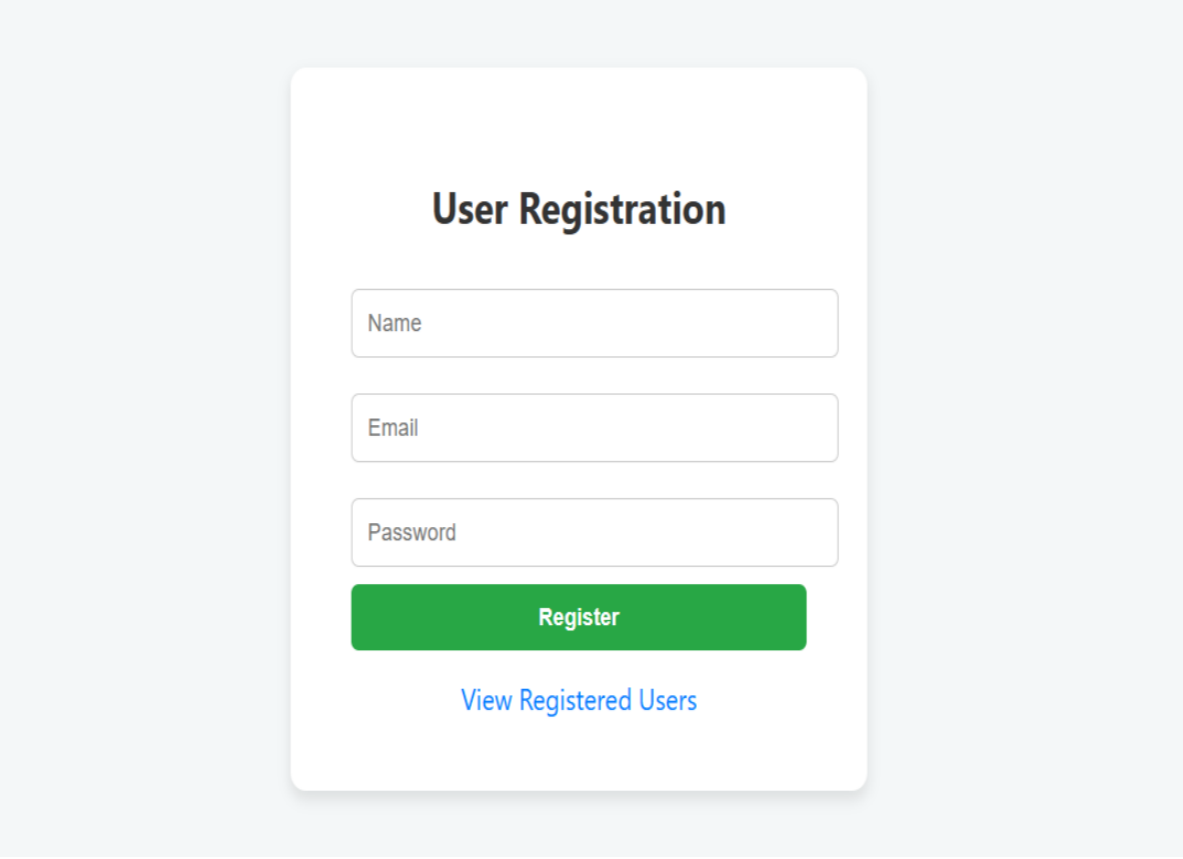
b. Write a JavaScript Program to get the user registration data and push to array/local storage with AJAX POST method and data list in new page.

```
<!DOCTYPE html>
<html>
<head>
 <title>Register</title>
 <style>
  body {
   font-family: 'Segoe UI', sans-serif;
   background: #f4f7f8;
   display: flex;
   justify-content: center;
   align-items: center;
   height: 100vh;
  }
  .container {
   background: white;
   padding: 40px;
   border-radius: 10px;
   box-shadow: 0 4px 8px rgba(0,0,0,0.1);
   width: 300px;
  }
  h2 {
   text-align: center;
   margin-bottom: 20px;
   color: #333;
  }

  input[type="text"],
  input[type="email"],
```

```
 input[type="password"] {
  width: 100%;
  padding: 10px;
  margin: 10px 0;
  border: 1px solid #ccc;
  border-radius: 5px;
 }
 button {
  width: 100%;
  background-color: #28a745;
  color: white;
  padding: 10px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  font-weight: bold;
 }
 button:hover {
  background-color: #218838;
 }
 a {
  display: block;
  text-align: center;
  margin-top: 15px;
  text-decoration: none;
  color: #007bff;
 }

 a:hover {
  text-decoration: underline;
 }
</style>
```

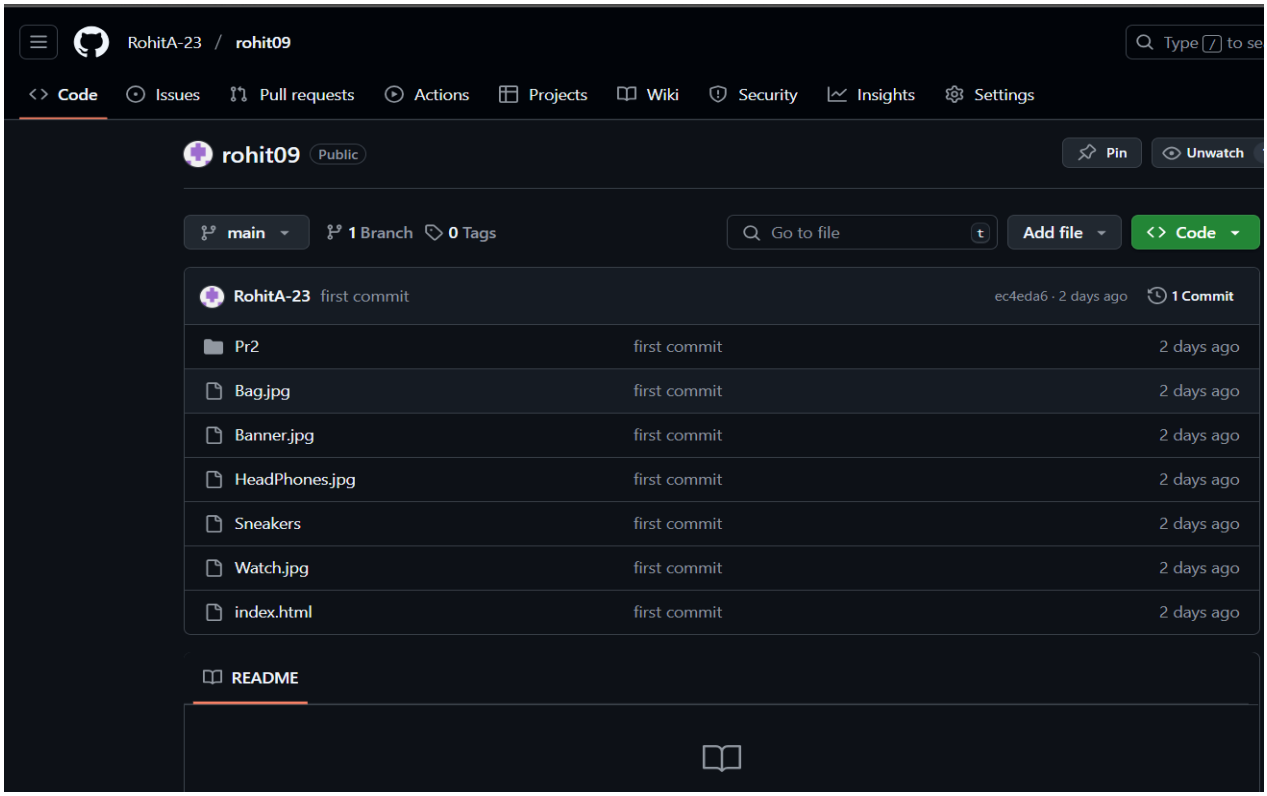
```
</head>
<body>
 <div class="container">
  <h2>User Registration</h2>
  <form id="registrationForm">
   <input type="text" id="name" placeholder="Name" required>
   <input type="email" id="email" placeholder="Email" required>
   <input type="password" id="password" placeholder="Password" required>
   <button type="submit">Register</button>
  </form>
  <a href="users.html">View Registered Users</a>
 </div>

 <script>
  const form = document.getElementById("registrationForm");
  form.addEventListener("submit", function (e) {
   e.preventDefault();
   const user = {
    name: document.getElementById("name").value,
    email: document.getElementById("email").value,
    password: document.getElementById("password").value
   };
   let users = JSON.parse(localStorage.getItem("users")) || [];
   users.push(user);
   localStorage.setItem("users", JSON.stringify(users));

   fetch("https://jsonplaceholder.typicode.com/posts", {
    method: "POST",
    headers: {
     "Content-type": "application/json"
    },
    body: JSON.stringify(user)
```

```
    })
      .then(response => response.json())
      .then(data => {
        alert("User registered successfully!");
        form.reset();
      })
      .catch(error => {
        console.error("Error:", error);
      });
  });
 </script>
</body>
</html>
```



Assignment 2

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



b. Create Docker Container Environment (NVIDEIA Docker or any other).

```
ubuntu@ubuntu-HP-Laptop-15s-du3xxx:~$ sudo docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Assignment 2

c). Create an Angular application which will do following actions: Register User,

Login User, Show User Data on Profile Component.

- **Index.html**

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>MyNewApp</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

- **Main.ts**

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app/app.module';
import { environment } from './environments/environment';
```

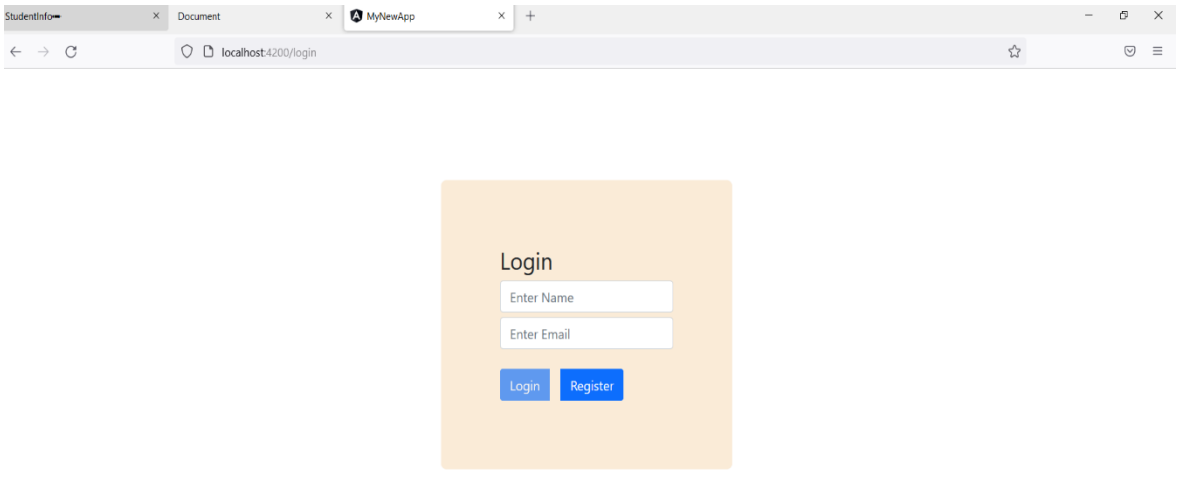
```
if (environment.production) {
  enableProdMode();
}
platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

- **Style.css**

```
  position: fixed;
  top: 20%;
  left: 40%;
  transform: translate(-50%, -50%);
  margin: 10%;
  padding: 5%;
  background-color: antiquewhite;
  font-size: 39px;
  width: max-content;
  border-radius: 2%;
} button {
  padding: 2%;
  background-color: skyblue;
  border: none;
  margin-right: 10%;
} .block {
  width: 50%;
  height: 100%;
} input {
  display: inline;
  border: none;
  margin-bottom: 5%;
  padding: 1%;
  width: 200% !important;
  border-radius: 5%; }
```

- **test.ts**

```
import 'zone.js/testing';
import { getTestBed } from '@angular/core/testing';
import {
  BrowserDynamicTestingModule,
  platformBrowserDynamicTesting
} from '@angular/platform-browser-dynamic/testing';
declare const require: {
  context(path: string, deep?: boolean, filter?: RegExp): {
    <T>(id: string): T;
    keys(): string[];
  };
};
getTestBed().initTestEnvironment(
  BrowserDynamicTestingModule,
  platformBrowserDynamicTesting(),
);
const context = require.context('./', true, /\.spec\.ts$/);
context.keys().map(context);
```



Assignment 3

a). Create a Node.JS Application which serves a static website.

- Index.js.

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

app.use(express.static('public'))

app.listen(4000,()=>{
    console.log("Server is Started");
})
```

- Package.json

```
{
  "name": "pr3a",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Rohit",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "express": "^5.1.0"
  }
}
```

Index.html:

Refer Assignment 1 A (Create a responsive web page which shows the ecommerce/college/exam admin dashboard with sidebar and statistics in cards using HTML, CSS and Bootstrap.)

Assignment 3

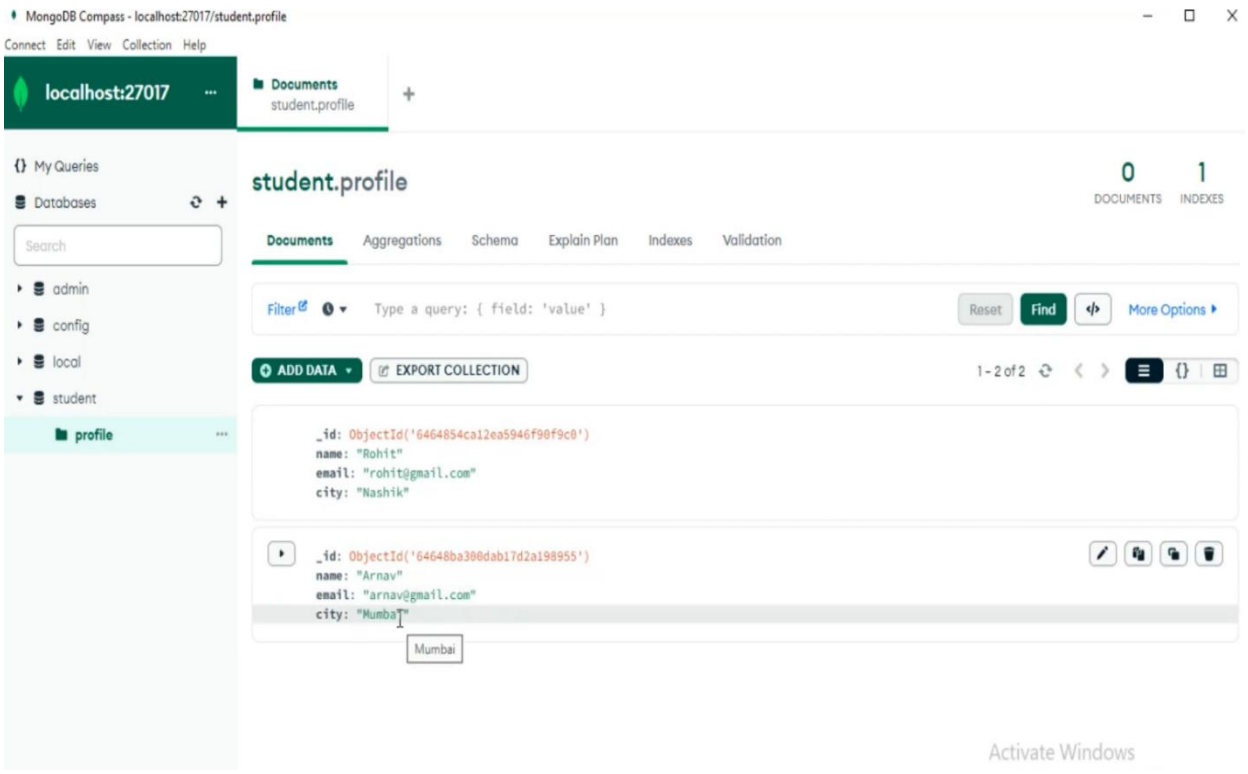
b). Create four API using Node.JS, ExpressJS and MongoDB for CURD Operations on assignment 2.C.

- Index.js

```
const dbConnect=require('./mongodb')
const express=require('express');
const {response}= require('express');
const app=express();
app.use(express.json())
app.get('/getData/', async (req,res)=>{
   let result=await dbConnect();
   result=await result.find().toArray();
   res.json(result)
})
app.post('/insertData/',async(req,res)=> {
   let result=await dbConnect();
   result=await result.insertOne(req.body);
   res.send("Data Inserted Successfully")
})
app.put('/updateData/:name',async(req,res)=> {
   let result=await dbConnect();
   result=await result.updateOneOne({name:req.params.name},{$set:req.body});
   res.send("Data Updated Successfully")
})
app.delete('/deleteData/:name',async(req,res)=> {
   let result=await dbConnect();
   result=await result.deleteOne({name:req.params.name});
   res.send("Data Deleted Successfully")
})
app.listen(3000, ()=>{
   console.log('Server runing on http://localhost:3000');  });
```

- **Mongodb.js**

```
const {MongoClient}=require('mongodb');
const url="mongodb://localhost:27017"
const database='student';
const client=new MongoClient(url);
const dbConnect=async()=>{
   const result=await client.connect();
   const db=await result.db(database);
   return db.collection('profile');
}
module.exports= dbConnect;
```



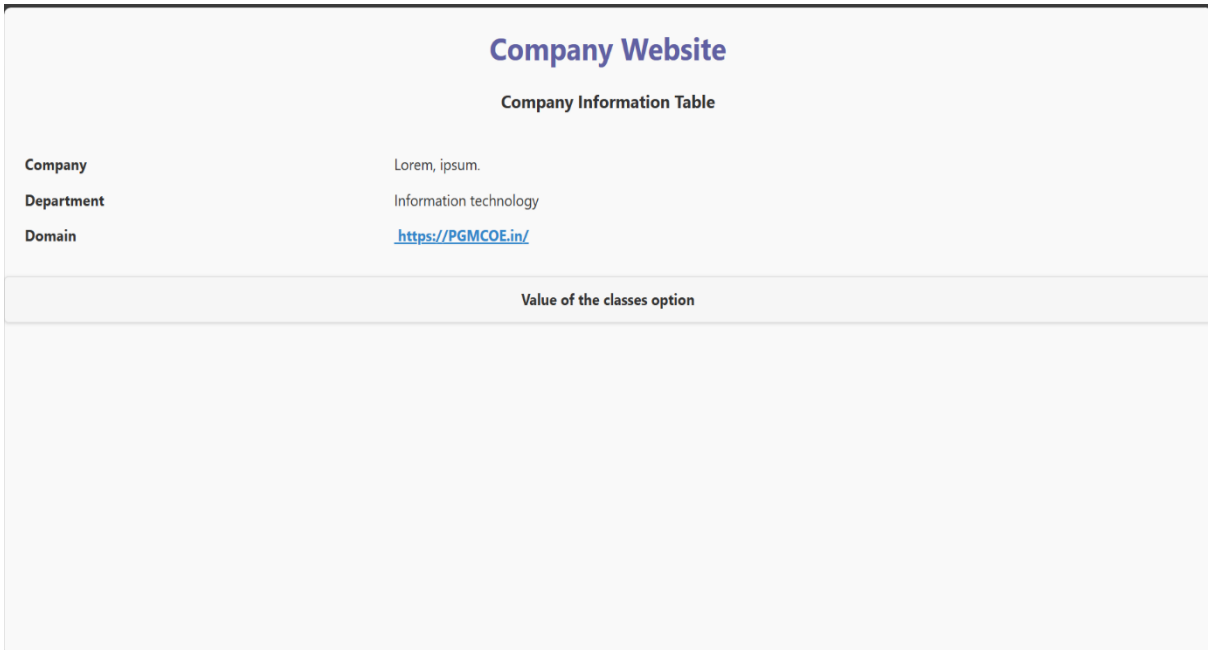
Assignment 4

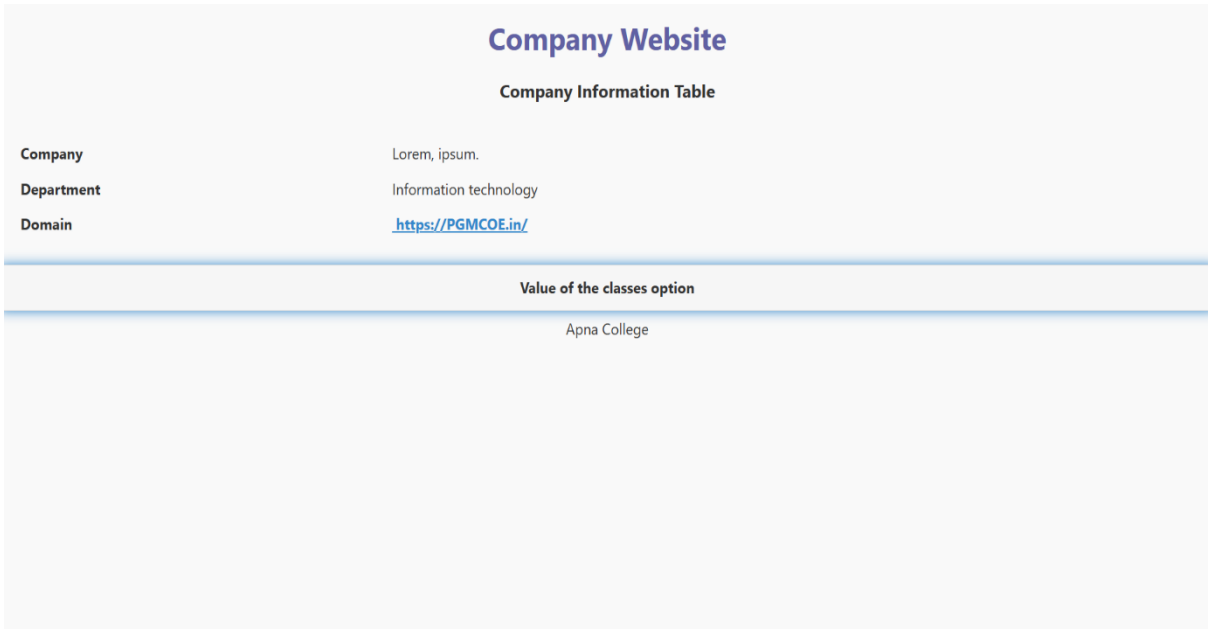
a). Create a simple Mobile Website using jQuery Mobile.

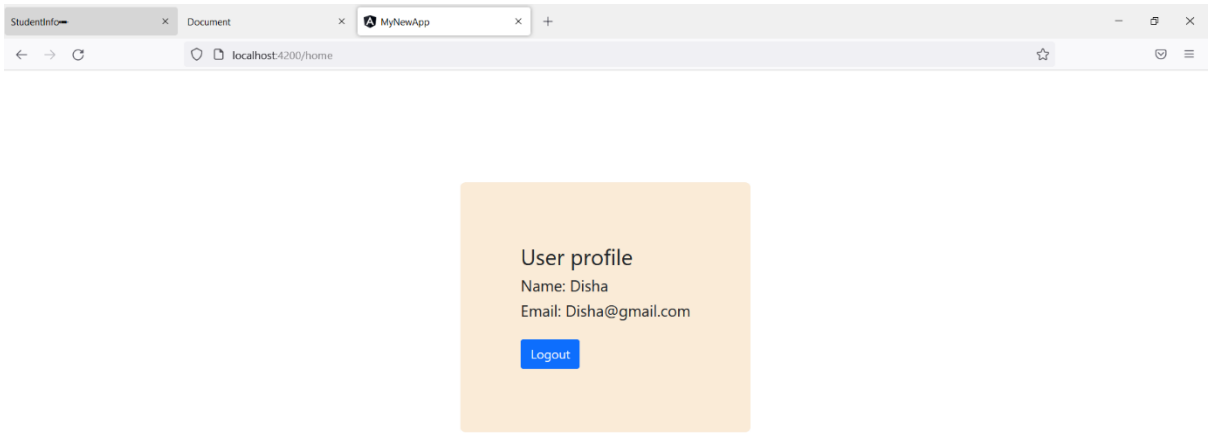
```
<!doctype html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <meta name="viewport" content=
"width=device-width, initial-scale=1">
   <title>table demo</title>
   <link rel="stylesheet" href=
"https://code.jquery.com/mobile/1.5.0-alpha.1/jquery.mobile-1.5.0-alpha.1.min.css">
   <script src=
"https://code.jquery.com/jquery-3.2.1.min.js">
   </script>
   <script src=
"https://code.jquery.com/mobile/1.5.0-alpha.1/jquery.mobile-1.5.0-alpha.1.min.js">
   </script>
</head>
<body>
   <center>
      <div data-role="page">
         <h1 style="color:#6060a3;">
            Company Website
         </h1>
         <h3>Company Information Table</h3>
         <div data-role="header" id="GFG">
            <div role="main" class="ui-content">
               <table data-role="table">
                  <thead>
                     <tr>
                        <th>Company</th>
```

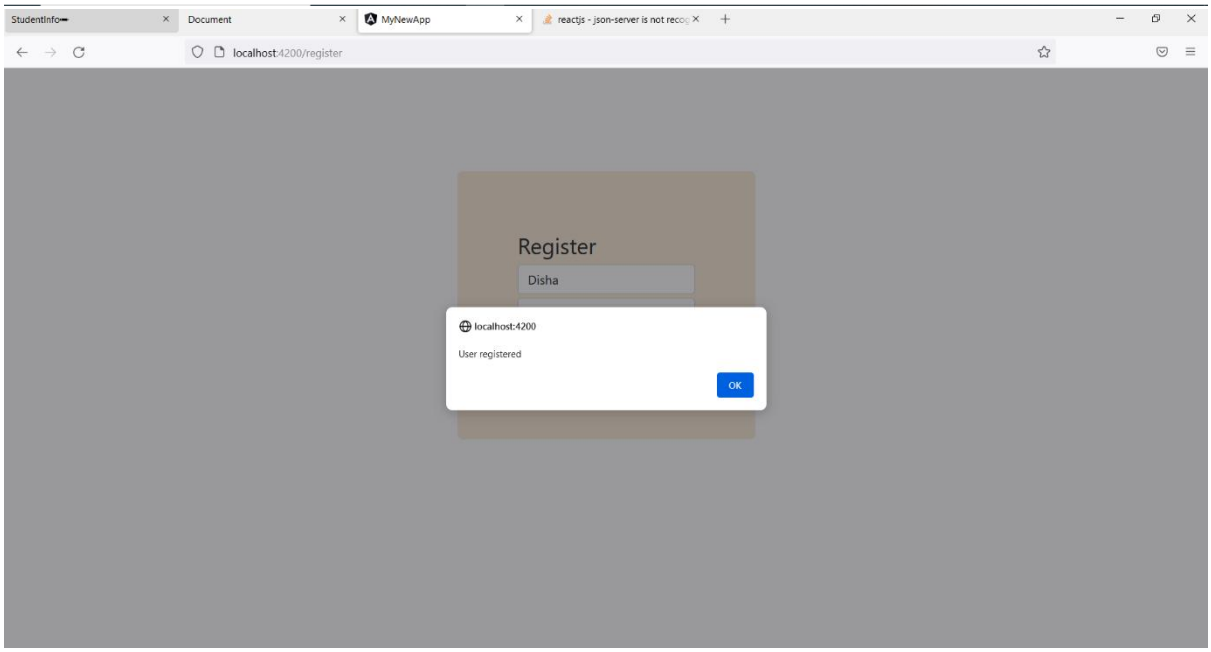
```
                    <th>Department</th>
                    <th>Domain</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>Lorem, ipsum.</td>
                    <td>Computer Science</td>
                    <td><a href="https://pict.edu/"
                            data-rel="external">
                            https://pict.edu/
                        </a>
                    </td>
                </tr>
            </tbody>
        </table>
    </div>
</div>
<input type="button" id="Button"
    value="Value of the classes option">
<div id="log"></div>
</div>
</center>
<script>
$(document).ready(function () {
    $("#GFG").table({
        classes: {
            "classes.Name": "GeeksforGeeks"
        }
});
    $("#GFG").table("option", "classes.Name", "GeeksforGeeks");
    $("#Button").on('click', function () {
```

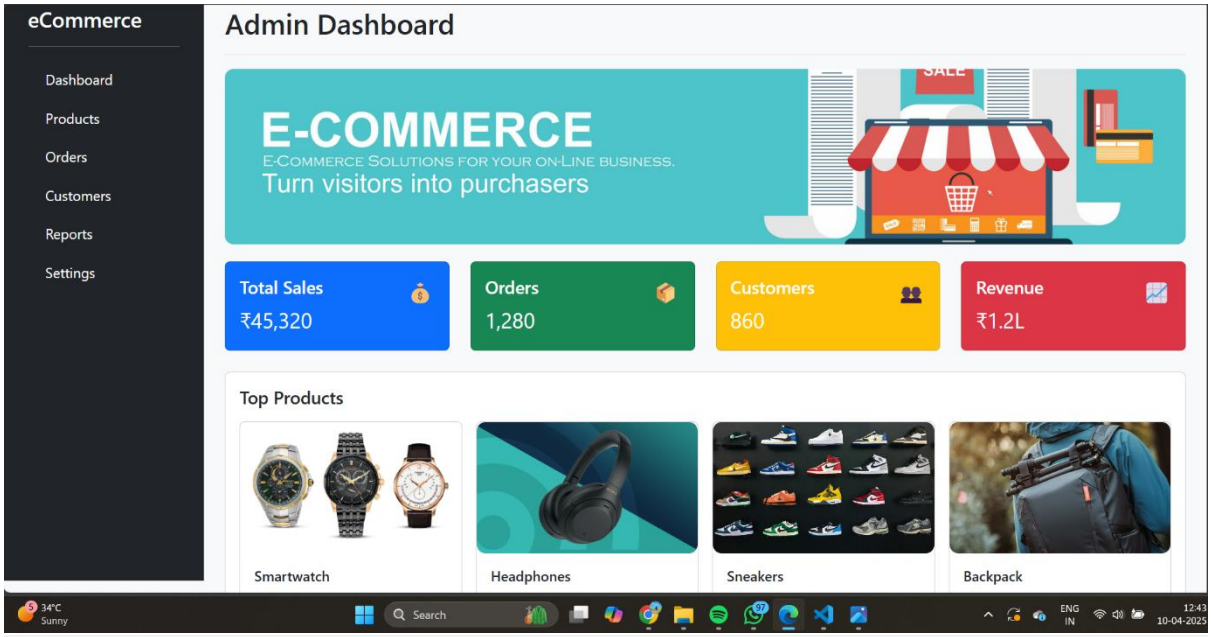
```
            var a = $("#GFG").table("option", "classes.Name");
            $("#log").html(a);
        });
    });
  </script>
</body>
</html>
```






StudentInfo
Document
MyNewApp
localhost:4200/home
User profile
Name: Disha
Email: Disha@gmail.com
Logout


StudentInfo
Document
MyNewApp
reactjs - json-server is not recog
localhost:4200/register
Register
Disha
localhost:4200
User registered
OK



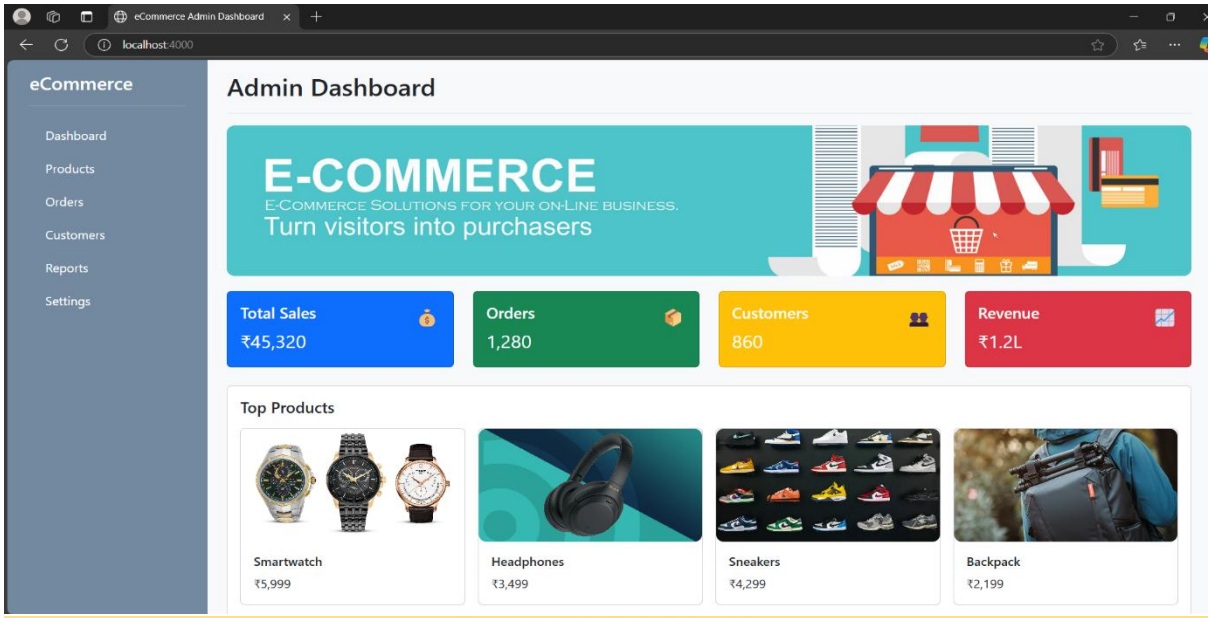
List of Registered Users

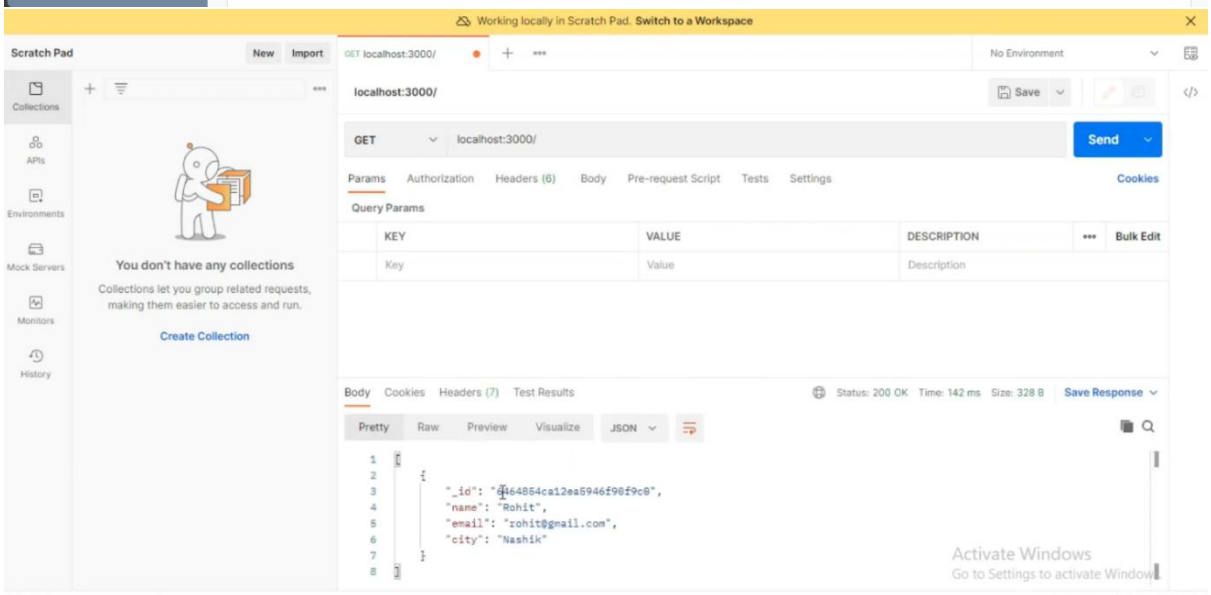
ROHIT TRIMBAKRAO AHER (rohitaher2328@gmail.com)

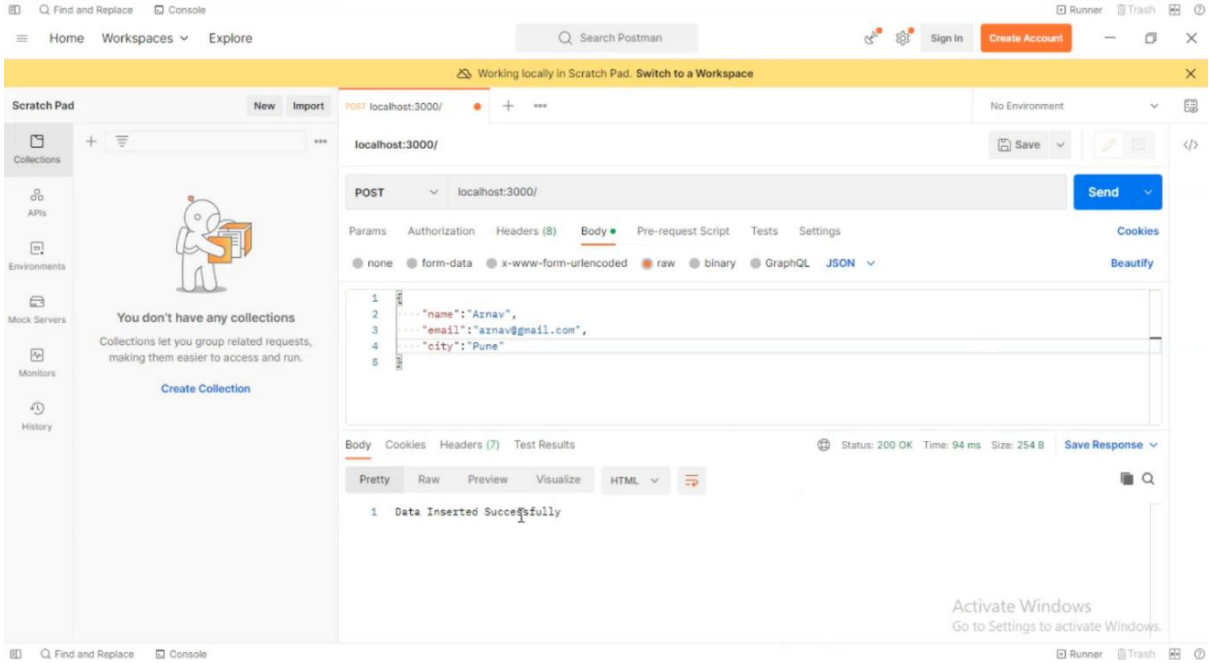
Aniket Shinde (aniketshinde2305@gmail.com)

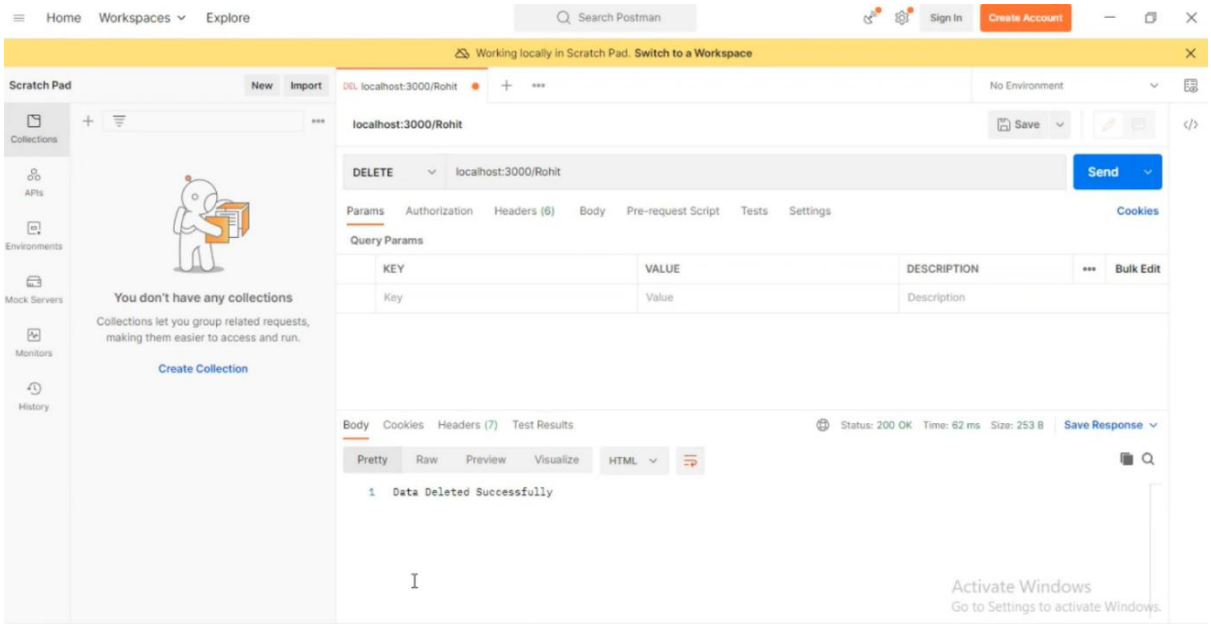
Abhishek (abhi23@gmail.com)

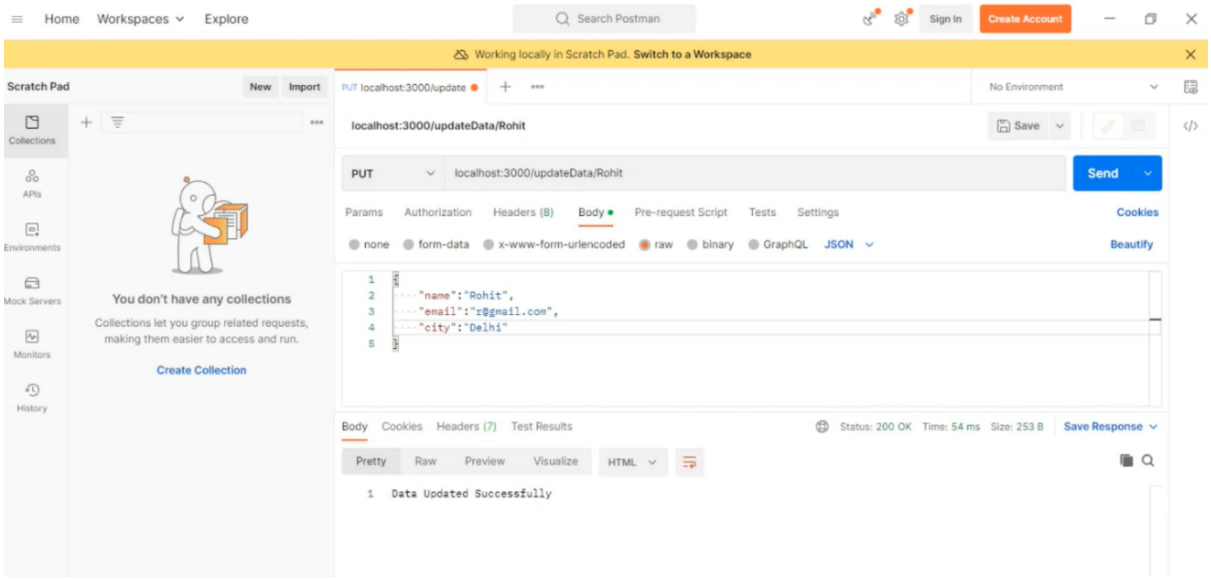
← Back to Registration


eCommerce Admin Dashboard
localhost:4000
eCommerce
Dashboard
Products
Orders
Customers
Reports
Settings
Admin Dashboard
E-COMMERCE
E-COMMERCE SOLUTIONS FOR YOUR ON-LINE BUSINESS.
Turn visitors into purchasers
Total Sales
₹45,320
Orders
1,280
Customers
860
Revenue
₹1.2L
Top Products
Smartwatch
₹5,999
Headphones
₹3,499
Sneakers
₹4,299
Backpack
₹2,199


Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
GET localhost:3000/
No Environment
localhost:3000/
Save
GET
localhost:3000/
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 142 ms
Size: 328 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"_id": "6764864ca12ea5946f98f9c0",
"name": "Rohit",
"email": "rohit@gmail.com",
"city": "Nashik"
}
Activate Windows
Go to Settings to activate Windows.
Find and Replace
Console
Runner
Trash


Home
Workspaces
Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
POST localhost:3000/
No Environment
localhost:3000/
Save
POST
localhost:3000/
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name":"Arnav",
"email":"arnav@gmail.com",
"city":"Pune"
}
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 94 ms
Size: 254 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
Data Inserted Succesfully
Activate Windows
Go to Settings to activate Windows.
Find and Replace
Console
Runner
Trash


Home
Workspaces
Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad
New
Import
DEL localhost:3000/Rohit
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
History
localhost:3000/Rohit
Save
DELETE
localhost:3000/Rohit
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 62 ms
Size: 253 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
1 Data Deleted Successfully
Activate Windows
Go to Settings to activate Windows.


Home
Workspaces
Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad
New
Import
PUT localhost:3000/update
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
History
localhost:3000/updateData/Rohit
Save
PUT
localhost:3000/updateData/Rohit
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
1 {
2 "name":"Rohit",
3 "email":"r@gmail.com",
4 "city":"Delhi"
5 }
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 54 ms
Size: 253 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
1 Data Updated Successfully