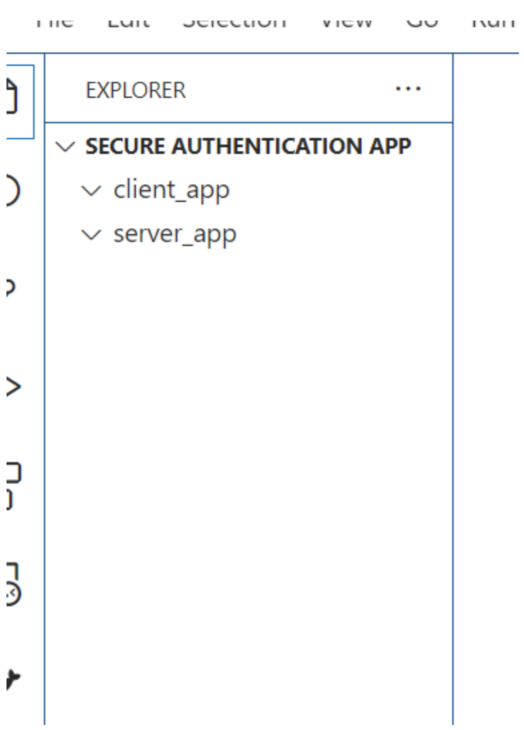
Answer Key

Step 1: Create the project structure.

1.1 Create the Folder as Secure_Authentication_App which contains two sub folder as client_app and server_app

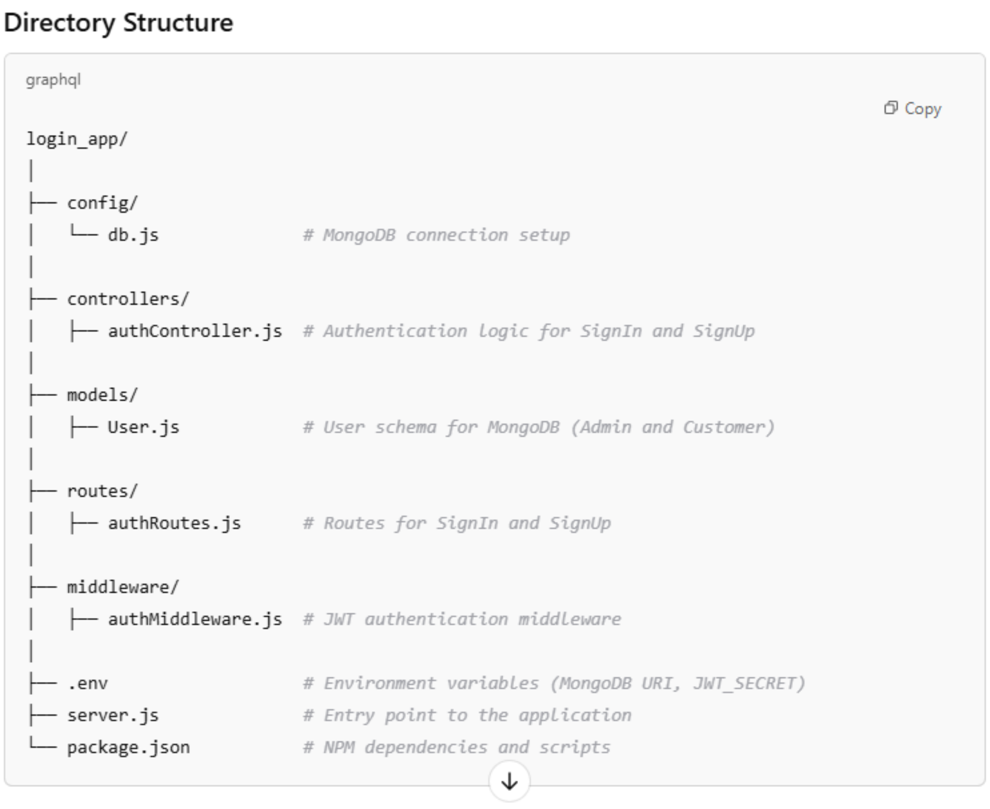
1.2 Then open the folder in VS Code editor



Step 2: Create the end point using backend technology with help of chat gpt.

2.1 Open the chat gpt and ask the prompt message
"Provide the directory structure as well as code for login_app using Express.js with MongoDB database for admin as well as customer for the SignIn and SignUp endpoints"

2.2The directory structure


Directory Structure

graphql

```
login_app/
│
├── config/
│   └── db.js              # MongoDB connection setup
│
├── controllers/
│   ├── authController.js  # Authentication logic for SignIn and SignUp
│
├── models/
│   ├── User.js            # User schema for MongoDB (Admin and Customer)
│
├── routes/
│   ├── authRoutes.js      # Routes for SignIn and SignUp
│
├── middleware/
│   ├── authMiddleware.js  # JWT authentication middleware
│
├── .env                   # Environment variables (MongoDB URI, JWT_SECRET)
├── server.js              # Entry point to the application
└── package.json           # NPM dependencies and scripts
```

2.3follow all these steps to create the project.

2.4Install Dependencies inside a server_app folder

```
npm init -y
npm install express mongoose bcryptjs jsonwebtoken dotenv body-parser
```

2.5.env file

```
PORT=5000
MONGO_URI=mongodb://localhost:27017/login_app
JWT_SECRET=your_secret_key
```

2.6config/db.js - MongoDB Connection

```
const mongoose = require('mongoose');
require('dotenv').config();
```

```
const connectDB = async () => {
   try {
      await mongoose.connect(process.env.MONGO_URI);
      console.log('MongoDB connected...');
   } catch (error) {
      console.error(error);
      process.exit(1);
   }
};

module.exports = connectDB;
```

2.7models/User.js - User Model (Admin & Customer)

```
const mongoose = require('mongoose');
const bcrypt = require('bcryptjs');

const userSchema = new mongoose.Schema({
   username: {
      type: String,
      required: true,
      unique: true
   },
   password: {
      type: String,
      required: true
   },
   role: {
      type: String,
      enum: ['admin', 'customer'],
      default: 'customer'
   }
});

// Hash password before saving
userSchema.pre('save', async function(next) {
    if (!this.isModified('password')) {
        return next();
    }
    this.password = await bcrypt.hash(this.password, 10);
});

// Method to check password
userSchema.methods.matchPassword = async function(enteredPassword) {
    return await bcrypt.compare(enteredPassword, this.password);
};

const User = mongoose.model('User', userSchema);

module.exports = User;
```

2.8controllers/authController.js - Authentication Logic

```
const jwt = require('jsonwebtoken');
const User = require('../models/User');

// Sign Up
const signup = async (req, res) => {
    const { username, password, role } = req.body;

    try {
        const userExists = await User.findOne({ username });
        if (userExists) {
            return res.status(400).json({ message: 'User already exists' });
        }

        const user = new User({ username, password, role });
        await user.save();

        res.status(201).json({ message: 'User created successfully', user });
    } catch (error) {
        res.status(500).json({ message: 'Server error' });
    }
};

// Sign In
const signin = async (req, res) => {
  const { username, password } = req.body;

  try {
    const user = await User.findOne({ username });
    if (!user) {
      return res.status(401).json({ message: 'Invalid credentials' });
    }

    const isMatch = await user.matchPassword(password);
    if (!isMatch) {
      return res.status(401).json({ message: 'Invalid credentials' });
    }

    const token = jwt.sign(
      { id: user._id, username: user.username, role: user.role },
      process.env.JWT_SECRET,
      { expiresIn: '1h' }
    );

    res.status(200).json({ message: 'Login successful', token });
  } catch (error) {
    res.status(500).json({ message: 'Server error' });
  }
};

module.exports = { signup, signin };
```

2.9routes/authRoutes.js - Routes for SignUp and SignIn

```
const express = require('express');
const { signup, signin } = require('../controllers/authController');

const router = express.Router();
```

```
// Sign Up Route
router.post('/signup', signup);

// Sign In Route
router.post('/signin', signin);

module.exports = router;
```

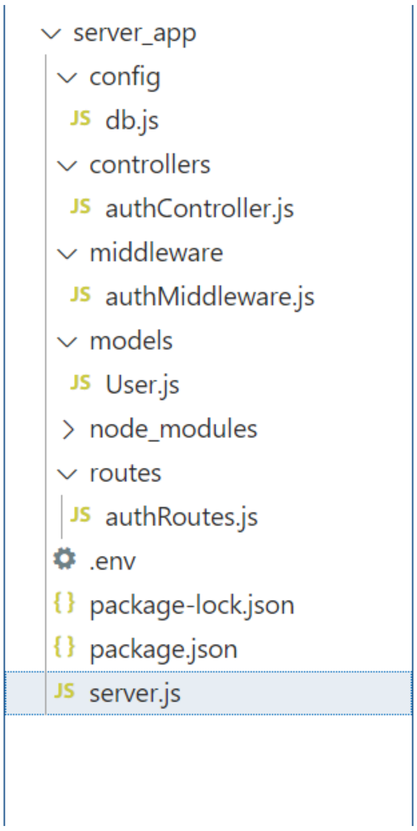
2.10 middleware/authMiddleware.js - JWT Authentication Middleware

```
const jwt = require('jsonwebtoken');

const protect = (req, res, next) => {
  const token = req.header('Authorization')?.split(' ')[1];

  if (!token) {
    return res.status(401).json({ message: 'No token, authorization denied'
});
  }

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.user = decoded;
    next();
  } catch (error) {
    res.status(401).json({ message: 'Token is not valid' });
  }
};

module.exports = { protect };
```

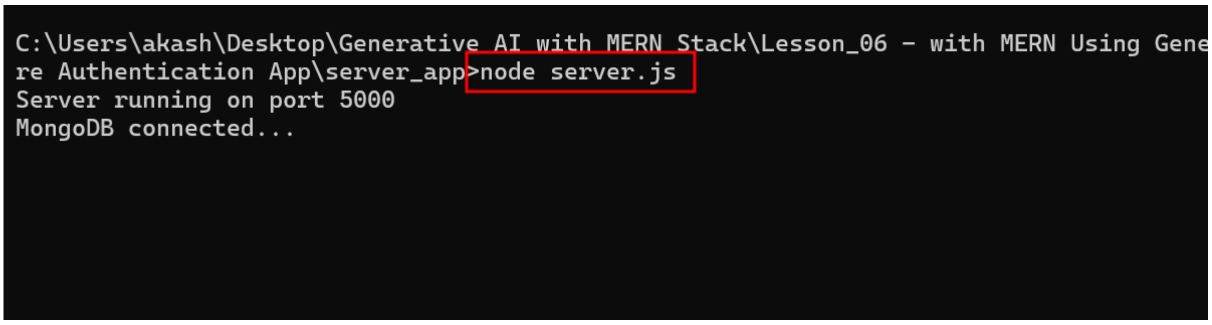
2.11 server.js - Entry Point to the Application

```
const express = require('express');
const bodyParser = require('body-parser');
const connectDB = require('./config/db');
const authRoutes = require('./routes/authRoutes');
require('dotenv').config();

const app = express();

// Middleware
app.use(bodyParser.json());

// Connect to database
connectDB();

// Routes
app.use('/api/auth', authRoutes);

// Start the server
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
   console.log(`Server running on port ${PORT}`);
});
```

2.12 after created folder and file in VS code. The folder structure as



2.13 Run the application using node server.js file



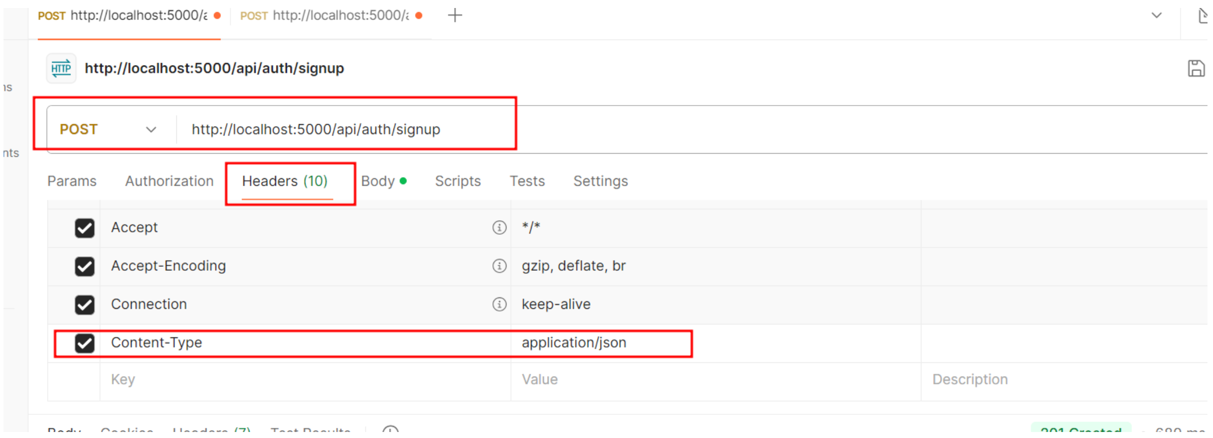
```
C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_06 - with MERN Using Gene
re Authentication App\server_app>node server.js
Server running on port 5000
MongoDB connected...
```

Step 3: Testing backend technology using post man client.

3.1 Open the post man client

3.2 Admin account creation
Method : post
URL : http://localhost:5000/api/auth/signup
Header property : content-type – application/json
Data in body part as : for admin user

```
{
   "username":"admin@gmail.com",
   "password":"admin@123",
   "role":"admin"
}
```



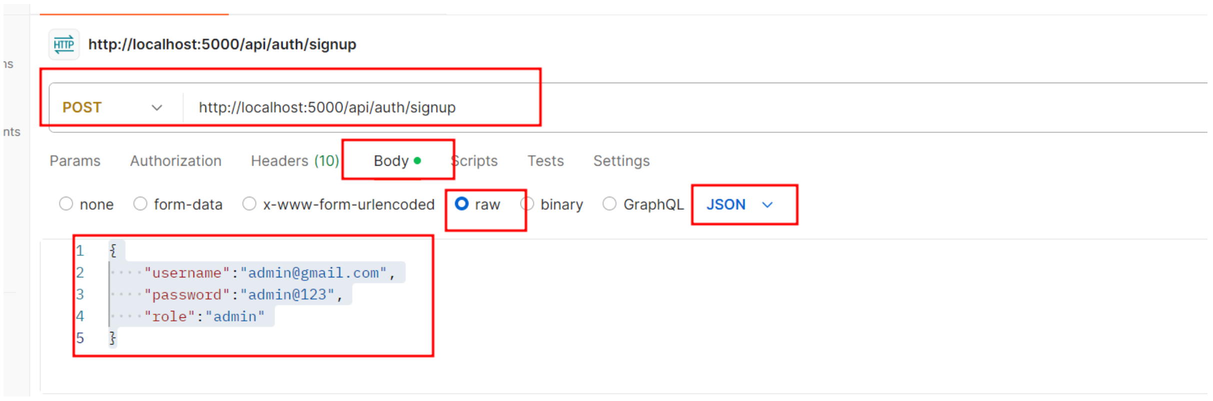
POST http://localhost:5000/a POST http://localhost:5000/a

http://localhost:5000/api/auth/signup

POST http://localhost:5000/api/auth/signup

Params Authorization Headers (10) Body Scripts Tests Settings

	Key	Value	Description
✓	Accept	*/*	
✓	Accept-Encoding	gzip, deflate, br	
✓	Connection	keep-alive	
✓	Content-Type	application/json	



http://localhost:5000/api/auth/signup

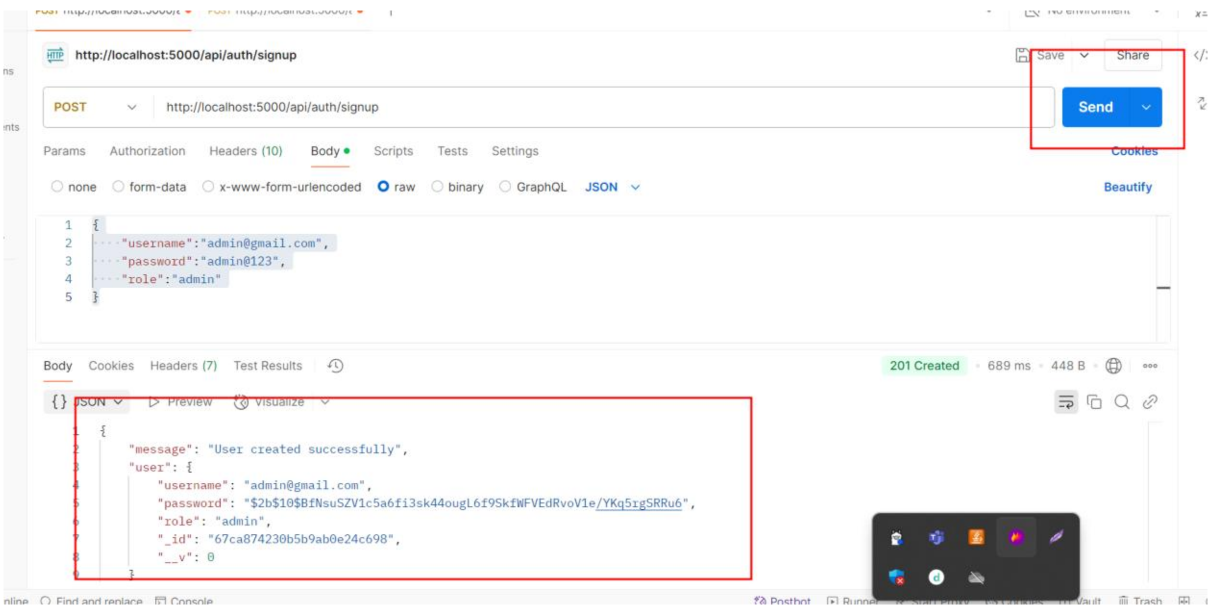
POST http://localhost:5000/api/auth/signup

Params Authorization Headers (10) Body Scripts Tests Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
····"username":"admin@gmail.com",
····"password":"admin@123",
····"role":"admin"
}
```

Now click on submit button



http://localhost:5000/api/auth/signup

POST http://localhost:5000/api/auth/signup

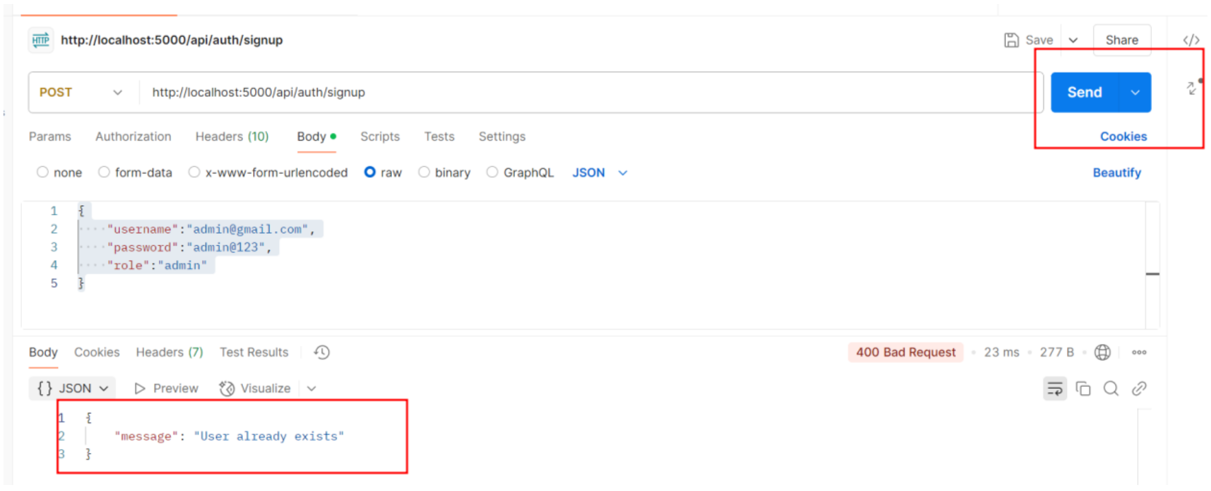
Body: raw, JSON

```
{
    "username":"admin@gmail.com",
    "password":"admin@123",
    "role":"admin"
}
```

201 Created · 689 ms · 448 B

```
{
    "message": "User created successfully",
    "user": {
        "username": "admin@gmail.com",
        "password": "$2b$10$BfNsuSZV1c5a6fi3sk44ougL6f9SkfWFVEdRvoV1e/YKq5rgSRRu6",
        "role": "admin",
        "_id": "67ca874230b5b9ab0e24c698",
        "__v": 0
    }
```

If you try to create once again same account



http://localhost:5000/api/auth/signup

POST http://localhost:5000/api/auth/signup

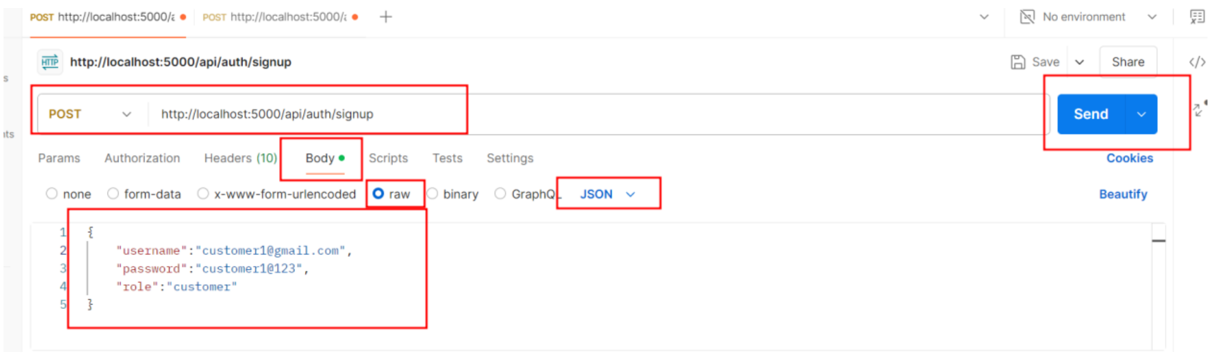
Body: raw, JSON

```
{
    "username":"admin@gmail.com",
    "password":"admin@123",
    "role":"admin"
}
```

400 Bad Request · 23 ms · 277 B

```
{
    "message": "User already exists"
}
```

3.3 Customer account creation



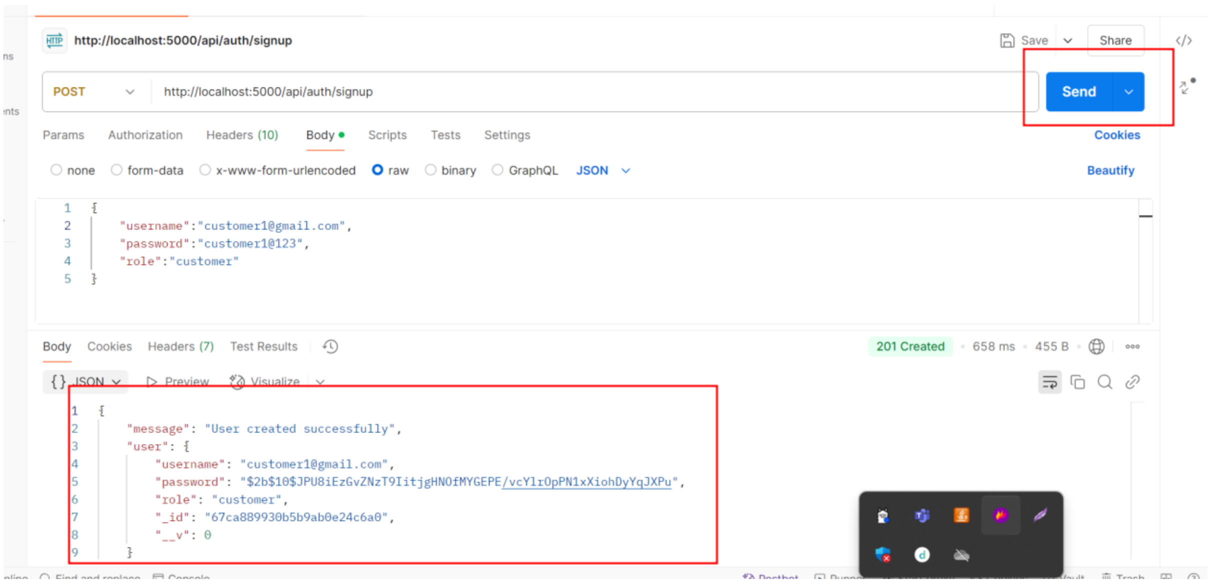
http://localhost:5000/api/auth/signup

POST http://localhost:5000/api/auth/signup

Body: raw, JSON

```
{
    "username":"customer1@gmail.com",
    "password":"customer1@123",
    "role":"customer"
}
```

After click on send button



POST http://localhost:5000/api/auth/signup

Body: raw JSON

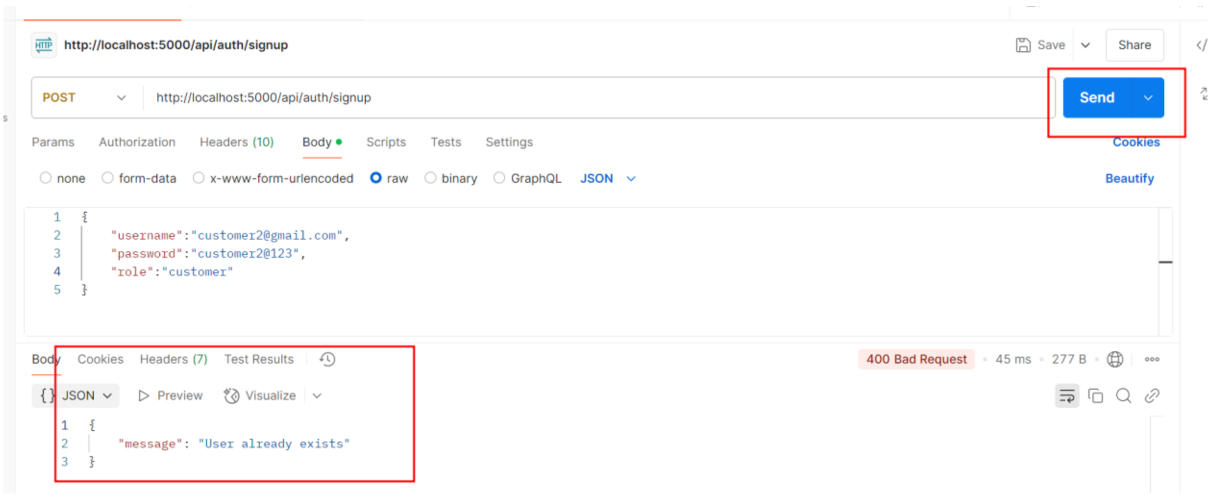
```
{
    "username":"customer1@gmail.com",
    "password":"customer1@123",
    "role":"customer"
}
```

201 Created · 658 ms · 455 B

```
{
    "message": "User created successfully",
    "user": {
        "username": "customer1@gmail.com",
        "password": "$2b$10$JPU8iEzGvZNzT9IitjgHNOfMYGEPE/vcYlrOpPN1xXiohDyYqJXPu",
        "role": "customer",
        "_id": "67ca889930b5b9ab0e24c6a0",
        "__v": 0
    }
```

Create more customer account

If you create same customer account, we will get the error as



POST http://localhost:5000/api/auth/signup

Body: raw JSON

```
{
    "username":"customer2@gmail.com",
    "password":"customer2@123",
    "role":"customer"
}
```

400 Bad Request · 45 ms · 277 B

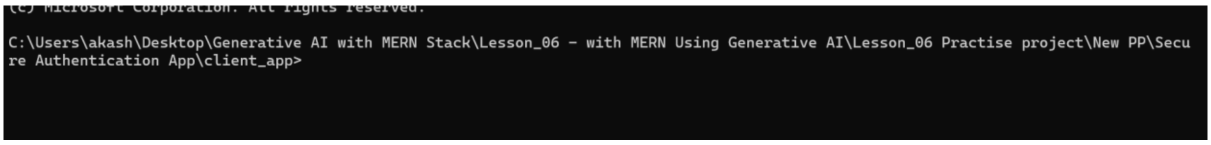
```
{
    "message": "User already exists"
}
```

Step 4: Create the frontend using react js technology with help of chat gpt.

"react js frontend application for admin and customer signing and signup with jwt token with vite framework"

4.1 create the react js project using vite framework

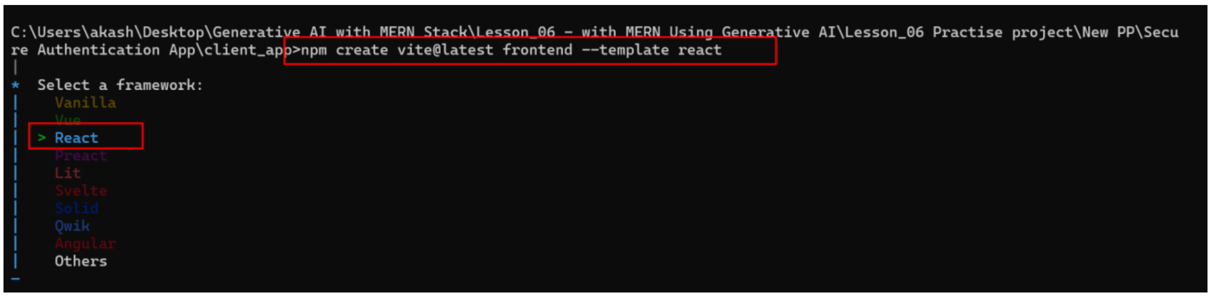
Open the command prompt or terminal inside client_app folder



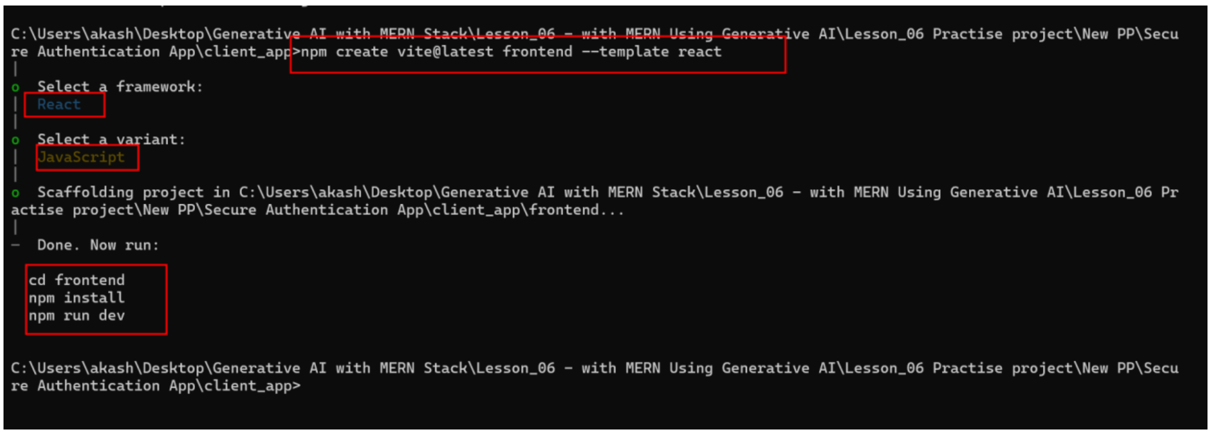
```
(c) Microsoft Corporation. All rights reserved.

C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_06 - with MERN Using Generative AI\Lesson_06 Practise project\New PP\Secu
re Authentication App\client_app>
```

4.2 create the project

npm create vite@latest frontend --template react

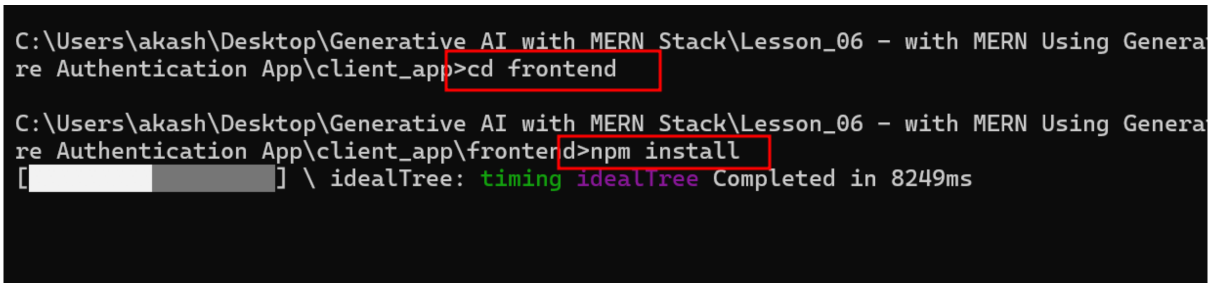


```
C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_06 - with MERN Using Generative AI\Lesson_06 Practise project\New PP\Secu
re Authentication App\client_app>npm create vite@latest frontend --template react
|
*  Select a framework:
|    Vanilla
|    Vue
|  > React
|    Preact
|    Lit
|    Svelte
|    Solid
|    Qwik
|    Angular
|    Others
—
```

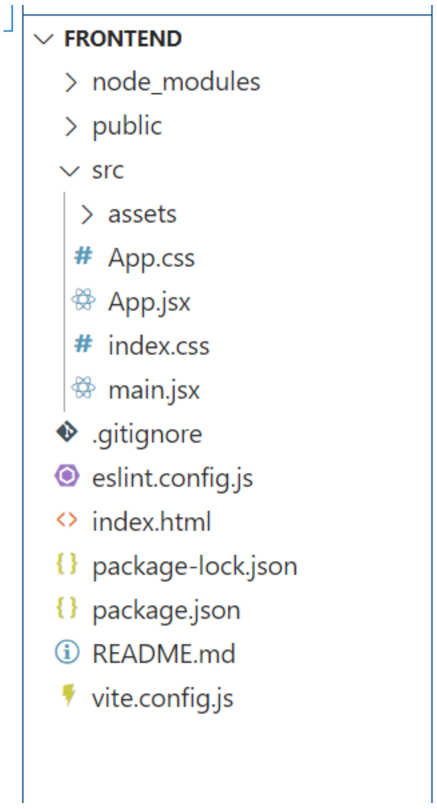


```
C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_06 - with MERN Using Generative AI\Lesson_06 Practise project\New PP\Secu
re Authentication App\client_app>npm create vite@latest frontend --template react
|
o  Select a framework:
|  React
|
o  Select a variant:
|  JavaScript
|
o  Scaffolding project in C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_06 - with MERN Using Generative AI\Lesson_06 Pr
actise project\New PP\Secure Authentication App\client_app\frontend...
|
—  Done. Now run:

  cd frontend
  npm install
  npm run dev

C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_06 - with MERN Using Generative AI\Lesson_06 Practise project\New PP\Secu
re Authentication App\client_app>
```

4.3 move inside a project and run the command as npm install to installed required dependencies to run the react js project


```
C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_06 - with MERN Using Genera
re Authentication App\client_app>cd frontend

C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_06 - with MERN Using Genera
re Authentication App\client_app\frontend>npm install
[                    ] \ idealTree: timing idealTree Completed in 8249ms
```

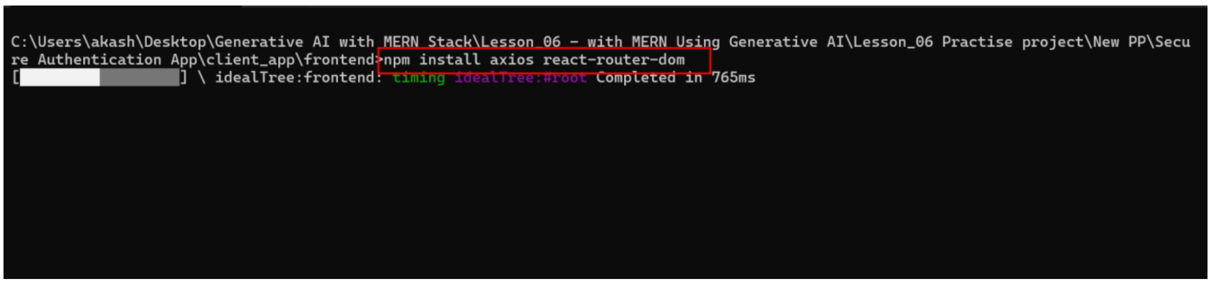
4.4 open the project in VS code


```
FRONTEND
  > node_modules
  > public
  v src
    > assets
    App.css
    App.jsx
    index.css
    main.jsx
  .gitignore
  eslint.config.js
  index.html
  package-lock.json
  package.json
  README.md
  vite.config.js
```

4.5 **Install Dependencies**

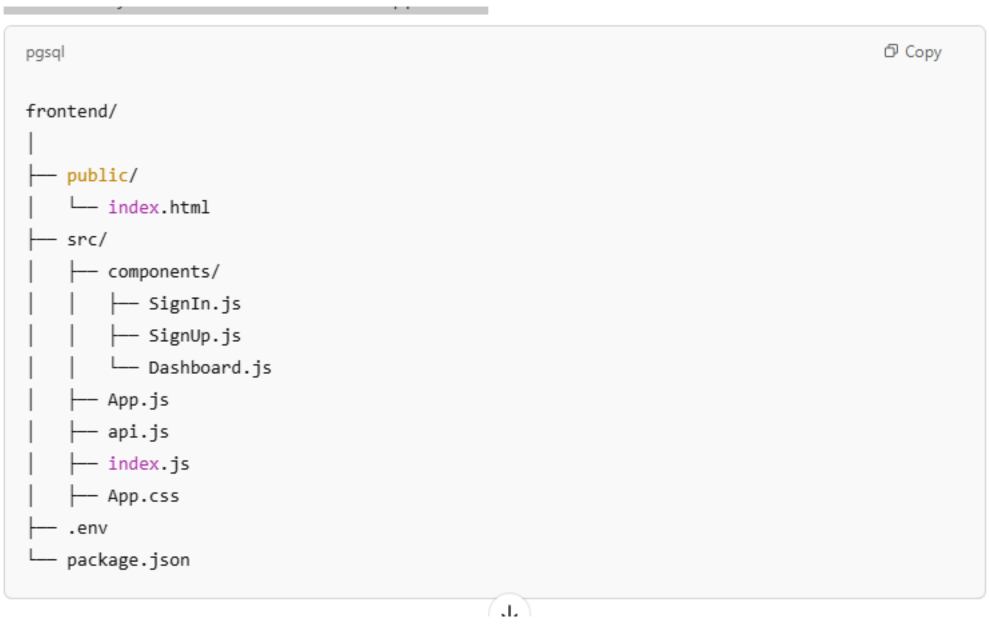
You will need axios for API requests and react-router-dom for routing.

npm install axios react-router-dom


```
C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_06 - with MERN Using Generative AI\Lesson_06 Practise project\New PP\Secu
re Authentication App\client_app\frontend>npm install axios react-router-dom
[              ] \ idealTree:frontend: timing idealTree:#root Completed in 765ms
```

4.6 **Set up the Directory Structure**

The directory structure for this frontend app will be:



pgsql

Copy

```
frontend/
│
├── public/
│   └── index.html
├── src/
│   ├── components/
│   │   ├── SignIn.js
│   │   ├── SignUp.js
│   │   └── Dashboard.js
│   ├── App.js
│   ├── api.js
│   ├── index.js
│   ├── App.css
├── .env
└── package.json
```

4.7 **Set up .env for Environment Variables**

In the root of your frontend directory, create a .env file for storing the API base URL.

.env file

VITE_API_URL=http://localhost:5000/api/auth

4.8 Create the Axios Configuration (src/api.js)
Set up Axios to handle API requests to the backend.

// src/api.js

```
// src/api.js
import axios from "axios";

// Create an Axios instance for API calls
const api = axios.create({
  baseURL: import.meta.env.VITE_API_URL,  // Get the base URL from the .env
file
  headers: {
    "Content-Type": "application/json",
  },
});

export default api;
```

4.9 **Create the SignIn Component (src/components/SignIn.jsx)**

This component will allow users to log in by providing their username and password.

```
// src/components/SignIn.jsx

import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import api from "../api";
import { Link } from "react-router-dom";

const SignIn = () => {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");
  const navigate = useNavigate();

  const handleSignIn = async (e) => {
    e.preventDefault();

    try {
      const response = await api.post("/signin", { username, password });
      localStorage.setItem("token", response.data.token); // Save the JWT token
      localStorage.setItem("username", username); // Save the username
      navigate("/dashboard");
    } catch (err) {
      setError("Invalid username or password");
    }
  };

  return (
    <div className="auth-container">
      <h2>Sign In</h2>
      {error && <p className="error-message">{error}</p>}
      <form onSubmit={handleSignIn} className="auth-form">
        <div>
          <label>Username:</label>
          <input
            type="text"
            value={username}
            onChange={(e) => setUsername(e.target.value)}
            required
          />
        </div>
        <div>
          <label>Password:</label>
          <input
            type="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            required
          />
        </div>
        <button type="submit">Sign In</button>
      </form>

      <p>
        Don't have an account? <Link to="/signup">Sign Up</Link>
      </p>
    </div>
  );
};

export default SignIn;
```

4.10 **Create the SignUp Component (src/components/SignUp.jsx)**

This component will allow new users to create an account.

```
// src/components/SignUp.jsx

import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import api from "../api";
import { Link } from "react-router-dom";

const SignUp = () => {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [role, setRole] = useState("customer");
  const [error, setError] = useState("");
  const navigate = useNavigate();

  const handleSignUp = async (e) => {
    e.preventDefault();

    try {
      await api.post("/signup", { username, password, role });
      navigate("/signin"); // Redirect to SignIn after successful signup
    } catch (err) {
      setError("Error creating user, please try again.");
    }
  };

  return (
    <div className="auth-container">
      <h2>Sign Up</h2>
      {error && <p className="error-message">{error}</p>}
      <form onSubmit={handleSignUp} className="auth-form">
        <div>
          <label>Username:</label>
          <input
            type="text"
            value={username}
            onChange={(e) => setUsername(e.target.value)}
            required
          />
        </div>
        <div>
          <label>Password:</label>
          <input
            type="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            required
          />
        </div>
        <div>
          <label>Role:</label>
          <select value={role} onChange={(e) => setRole(e.target.value)}>
            <option value="customer">Customer</option>
            <option value="admin">Admin</option>
          </select>
        </div>
        <button type="submit">Sign Up</button>
      </form>

      <p>
        Already have an account? <Link to="/signin">Sign In</Link>
      </p>
    </div>
  );
};

export default SignUp;
```

4.11 **Create the Dashboard Component (src/components/Dashboard.jsx)**

This component will be the user dashboard and can display data based on the user role.

```
// src/components/Dashboard.js
import React from "react";
import { useNavigate } from "react-router-dom";

const Dashboard = () => {
  const navigate = useNavigate();

  // Get the username and token from localStorage
  const username = localStorage.getItem("username");

  const handleLogout = () => {
    // Clear JWT token and username from localStorage
    localStorage.removeItem("token");
    localStorage.removeItem("username");

    // Redirect to SignIn page
    navigate("/signin");
  };

  return (
    <div className="dashboard-container">
      <h2>Welcome to the Dashboard</h2>
      <p>Welcome, {username}!</p>
      <button onClick={handleLogout} className="logout-button">
        Logout
      </button>
    </div>
  );
};

export default Dashboard;
```

4.12 **Set up Routing in App.jsx**

```
// src/App.jsx
import React from "react";
import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
import SignIn from "./components/SignIn";
import SignUp from "./components/SignUp";
import Dashboard from "./components/Dashboard";
import "./App.css";
function App() {
  return (
    <Router>
      <div className="App">
        <Routes>
          <Route path="/signin" element={<SignIn />} />
          <Route path="/signup" element={<SignUp />} />
          <Route path="/dashboard" element={<Dashboard />} />
          <Route path="/" element={<SignIn />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

4.13 **Styling (Optional)**

You can add some basic styling for the forms and other elements in App.css.

```
/* src/App.css */
/* src/App.css */
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}
```

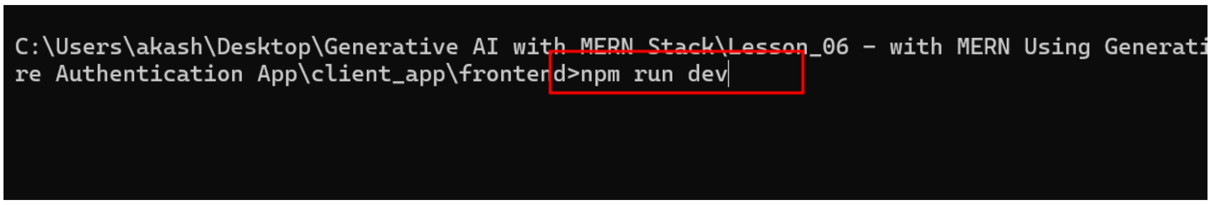
```
body {
  font-family: Arial, sans-serif;
  background-color: #e9ecef;
  padding: 20px;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

.App {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100%;
  width: 100%;
}

.auth-container {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  max-width: 400px;
  width: 100%;
  padding: 30px;
  background-color: #f4f4f9;
  border-radius: 8px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.auth-form {
  width: 100%;
  display: flex;
  flex-direction: column;
}

.auth-form input,
.auth-form select {
  padding: 12px;
  margin: 10px 0;
  border-radius: 4px;
  border: 1px solid #ccc;
  font-size: 16px;
}

.auth-form button {
  padding: 12px;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  font-size: 16px;
}

.auth-form button:hover {
  background-color: #0056b3;
}

.error-message {
  color: red;
  margin-bottom: 10px;
}

p {
  margin-top: 10px;
}

a {
  color: #007bff;
  text-decoration: none;
}

a:hover {
  text-decoration: underline;
}
```

```
h2 {
  margin-bottom: 20px;
  font-size: 24px;
  text-align: center;
  color: #333;
}

p {
  text-align: center;
  font-size: 16px;
}
```

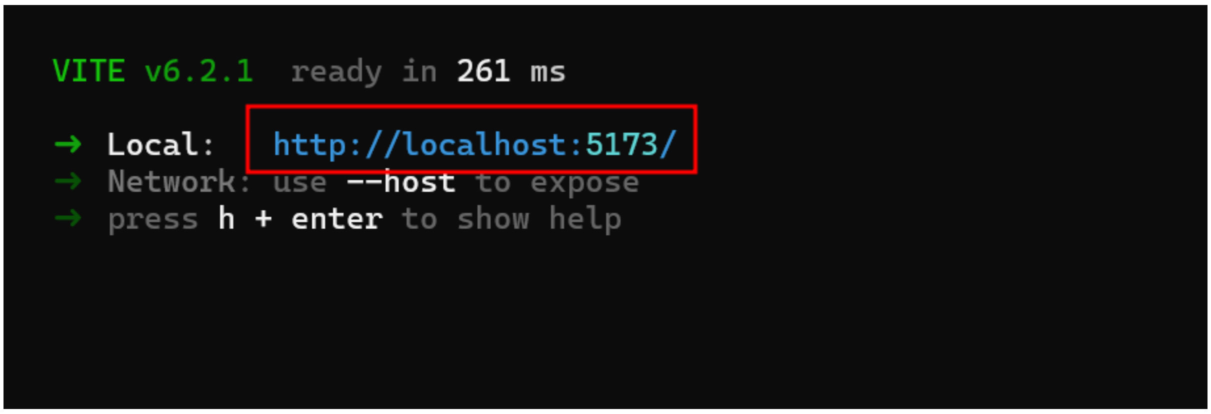
4.15 **Run the Application**

Now, start the Vite development server:

npm run dev



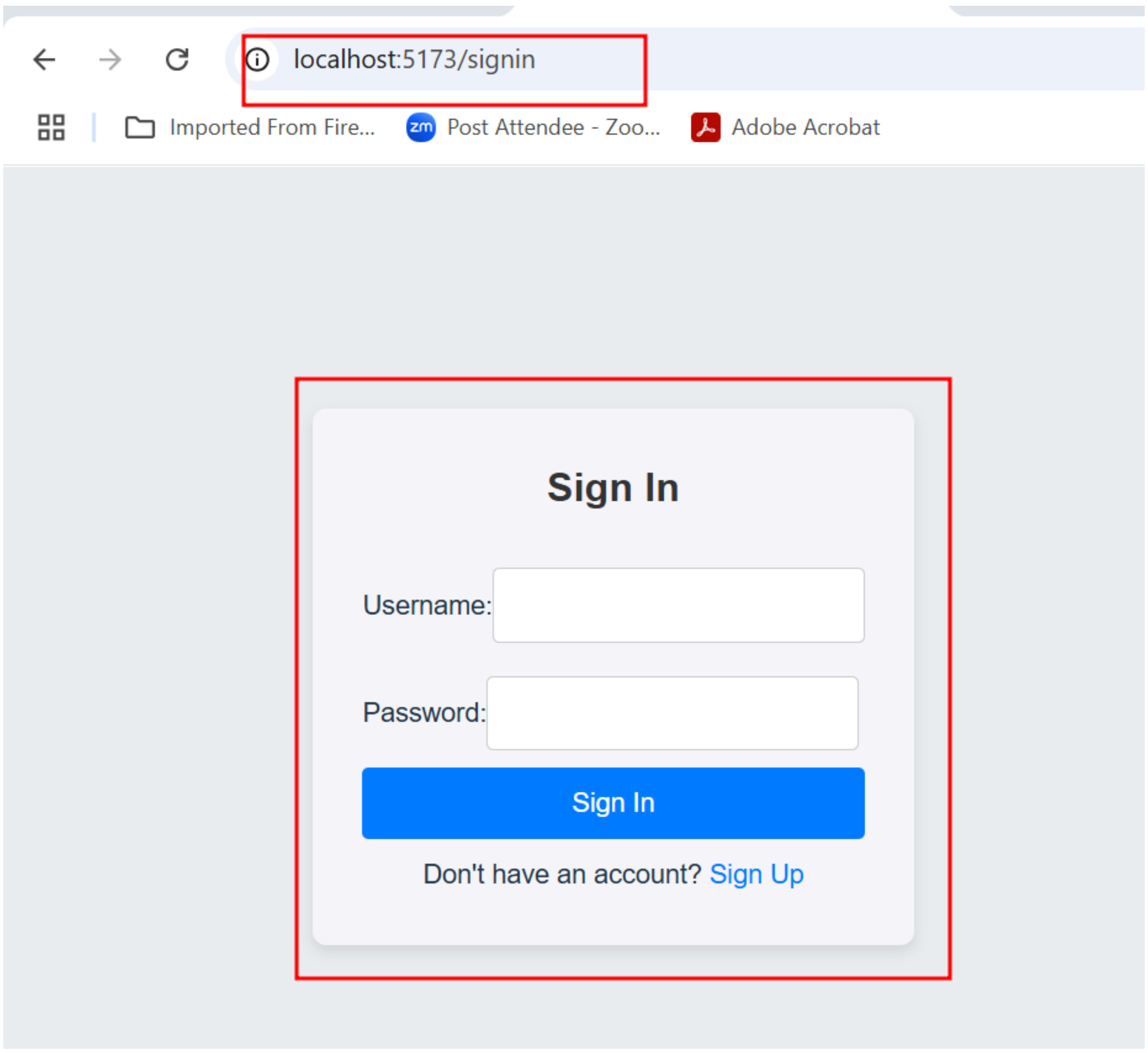
```
C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_06 - with MERN Using Generati
re Authentication App\client_app\frontend>npm run dev
```



```
  VITE v6.2.1  ready in 261 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

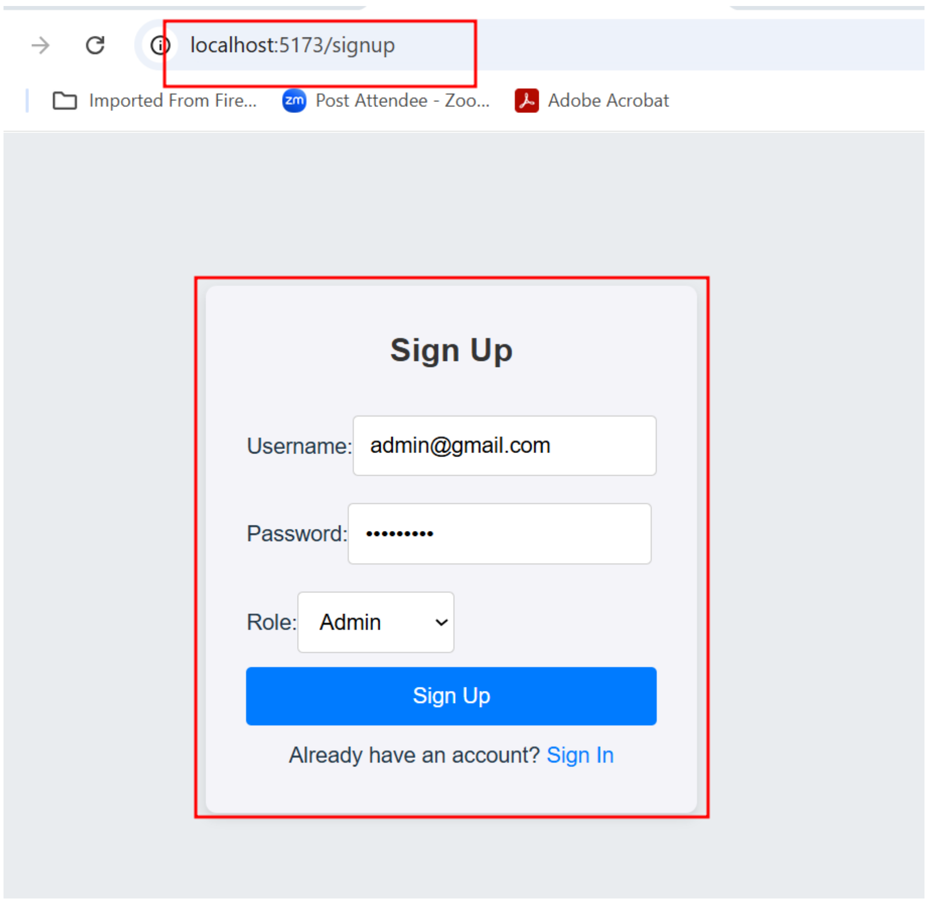
4.16 open this url on browser

http://localhost:5173

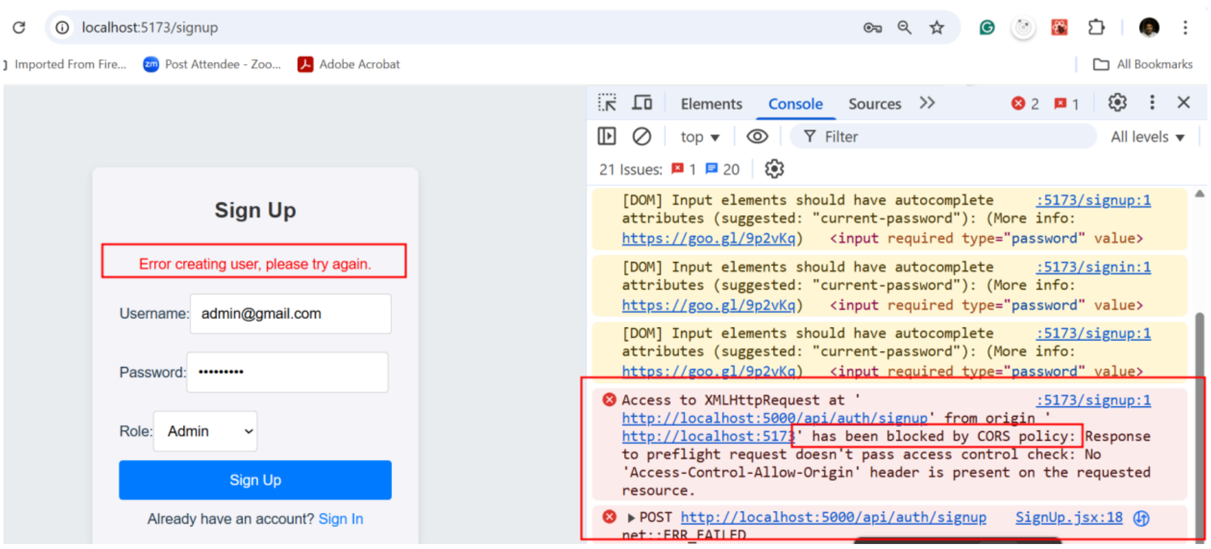

localhost:5173/signin

Imported From Fire... Post Attendee - Zoo... Adobe Acrobat

4.17 try to create the admin account.



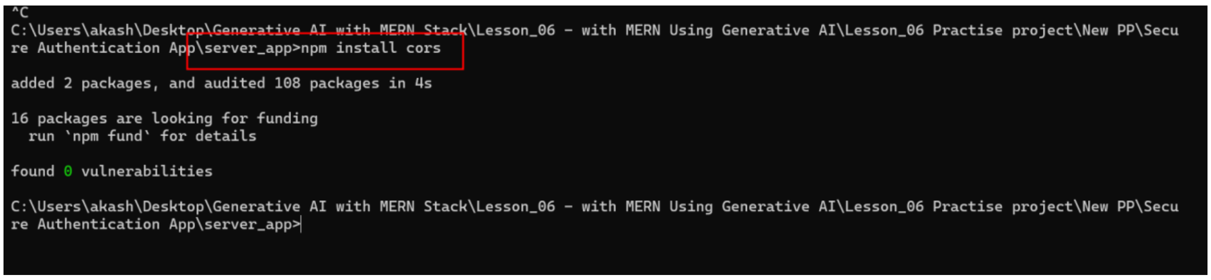
4.18 try to create the admin account



```
[DOM] Input elements should have autocomplete     :5173/signup:1
attributes (suggested: "current-password"): (More info:
https://goo.gl/9p2vKq)  <input required type="password" value>
[DOM] Input elements should have autocomplete     :5173/signin:1
attributes (suggested: "current-password"): (More info:
https://goo.gl/9p2vKq)  <input required type="password" value>
[DOM] Input elements should have autocomplete     :5173/signup:1
attributes (suggested: "current-password"): (More info:
https://goo.gl/9p2vKq)  <input required type="password" value>
Access to XMLHttpRequest at '                     :5173/signup:1
http://localhost:5000/api/auth/signup' from origin '
http://localhost:5173' has been blocked by CORS policy: Response
to preflight request doesn't pass access control check: No
'Access-Control-Allow-Origin' header is present on the requested
resource.
▶ POST http://localhost:5000/api/auth/signup     SignUp.jsx:18
net::ERR_FAILED
```

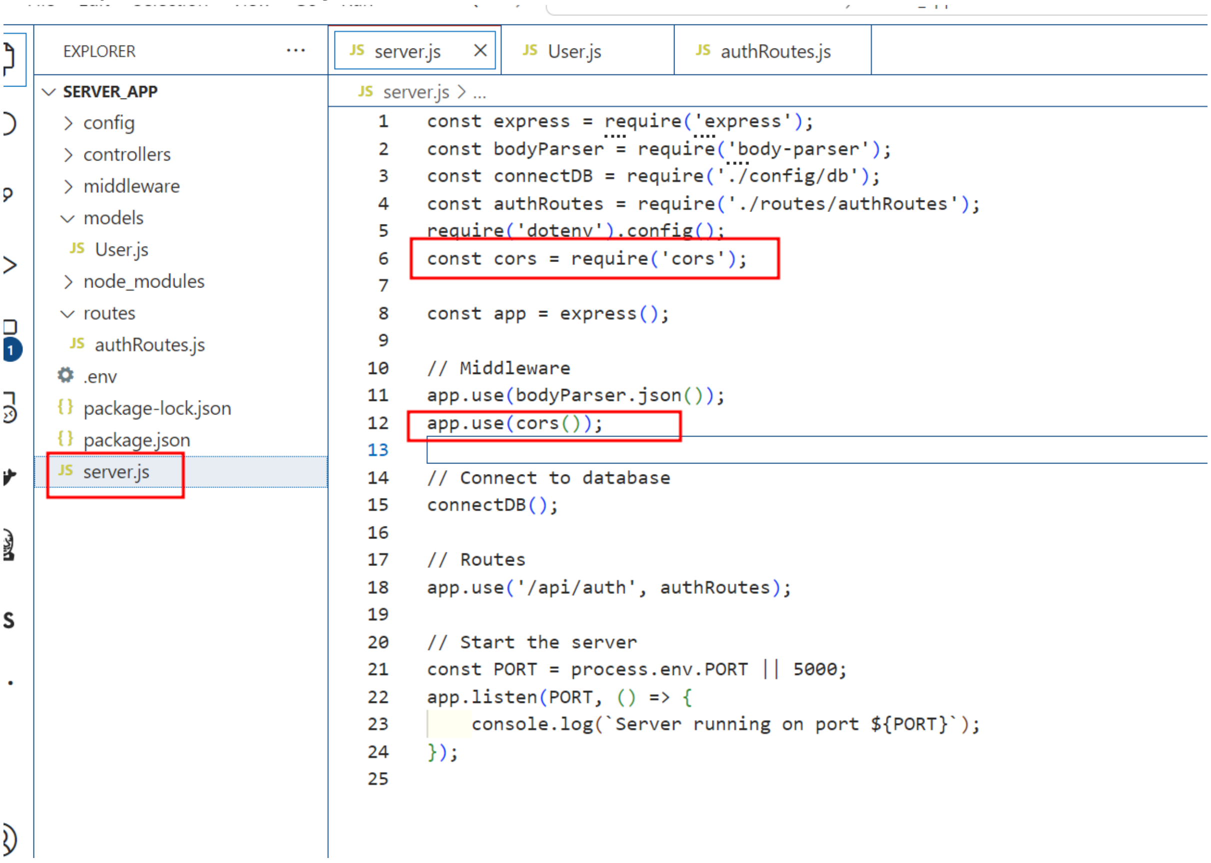
If you get error as cors (cross origin resource sharing) issue. Then in backend technology you need to install cors node js module and enable middleware

4.19 In backend technology terminology



```
^C
C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_06 - with MERN Using Generative AI\Lesson_06 Practise project\New PP\Secu
re Authentication App\server_app>npm install cors

added 2 packages, and audited 108 packages in 4s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\akash\Desktop\Generative AI with MERN Stack\Lesson_06 - with MERN Using Generative AI\Lesson_06 Practise project\New PP\Secu
re Authentication App\server_app>
```

4.20 open the server.js file and add cors middle module



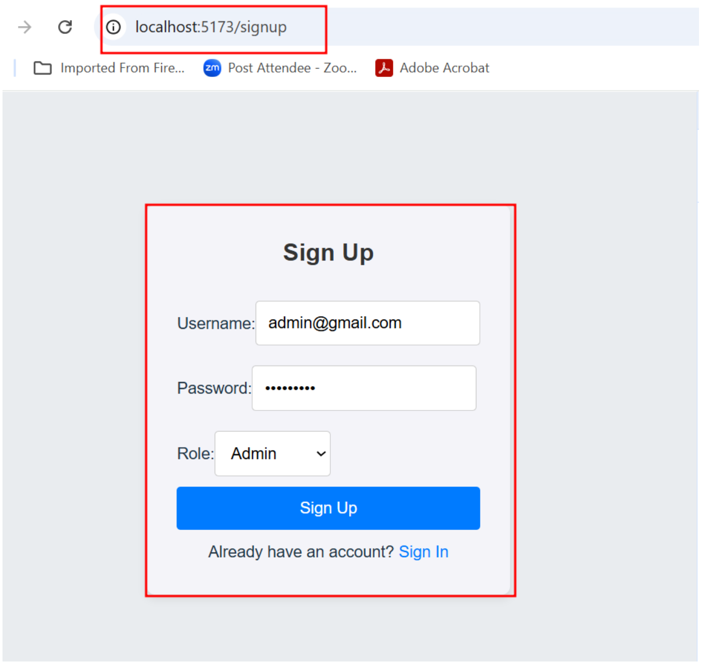
```
const express = require('express');
const bodyParser = require('body-parser');
const connectDB = require('./config/db');
const authRoutes = require('./routes/authRoutes');
require('dotenv').config();
const cors = require('cors');

const app = express();

// Middleware
app.use(bodyParser.json());
app.use(cors());

// Connect to database
connectDB();

// Routes
app.use('/api/auth', authRoutes);

// Start the server
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
    console.log(`Server running on port ${PORT}`);
});
```

Updated server.js file

```
const express = require('express');
const bodyParser = require('body-parser');
const connectDB = require('./config/db');
const authRoutes = require('./routes/authRoutes');
require('dotenv').config();
const cors = require('cors');

const app = express();
```

```
// Middleware
app.use(bodyParser.json());
app.use(cors());

// Connect to database
connectDB();

// Routes
app.use('/api/auth', authRoutes);

// Start the server
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

4.21 re run the application



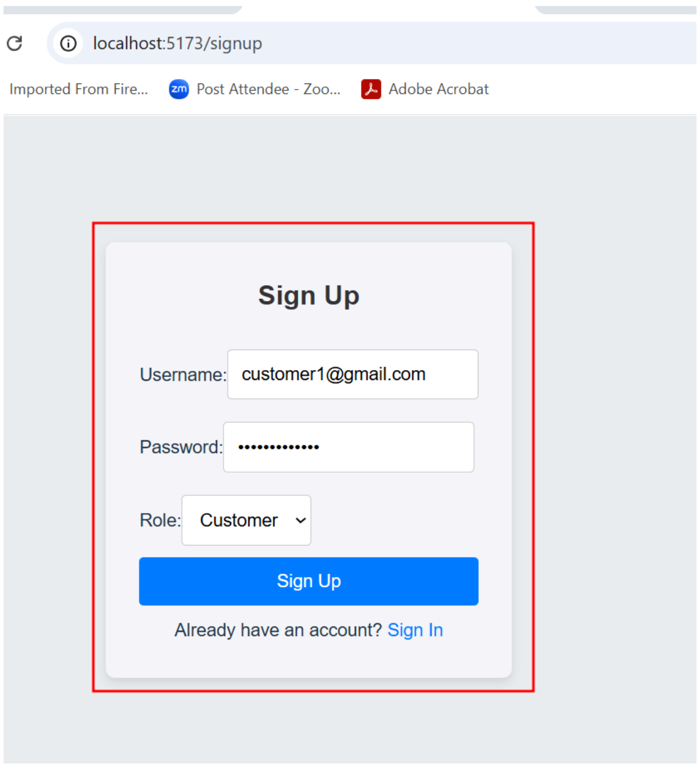
Sign Up

Username: admin@gmail.com

Password: •••••••••

Role: Admin

4.22 Now you can create more than one account for user as well as admin



localhost:5173/signup

Imported From Fire... Post Attendee - Zoo... Adobe Acrobat

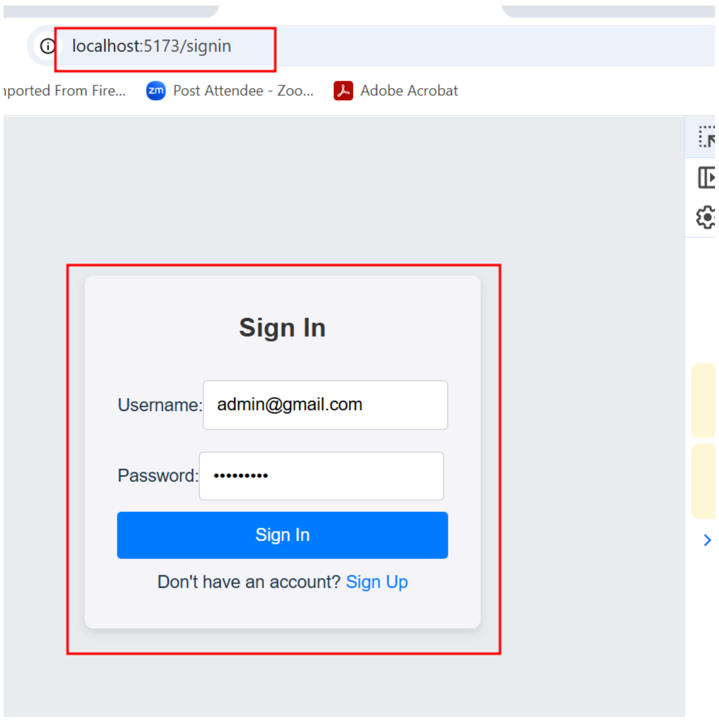
Sign Up

Username: customer1@gmail.com

Password: •••••••••••••

Role: Customer

4.23 Now you can do the SignIn with admin as well as customer account



localhost:5173/signin

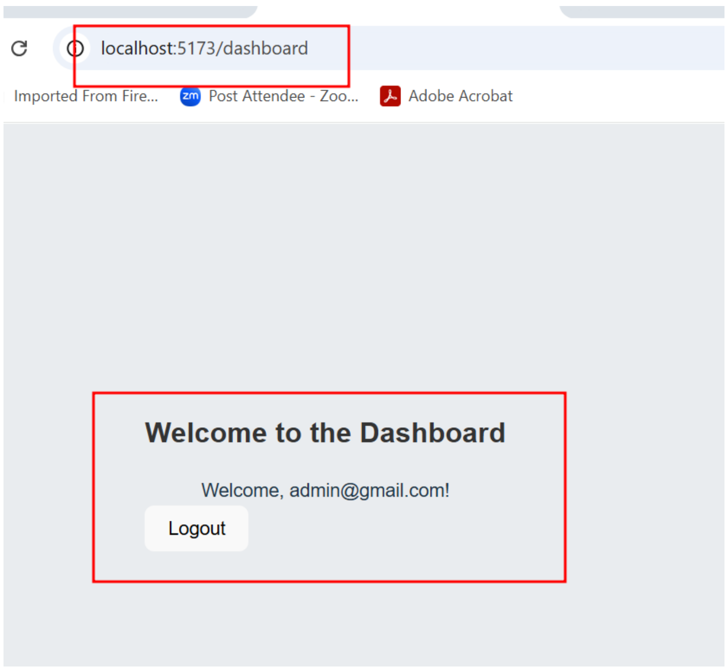
ported From Fire... Post Attendee - Zoo... Adobe Acrobat

Sign In

Username: admin@gmail.com

Password: •••••••••

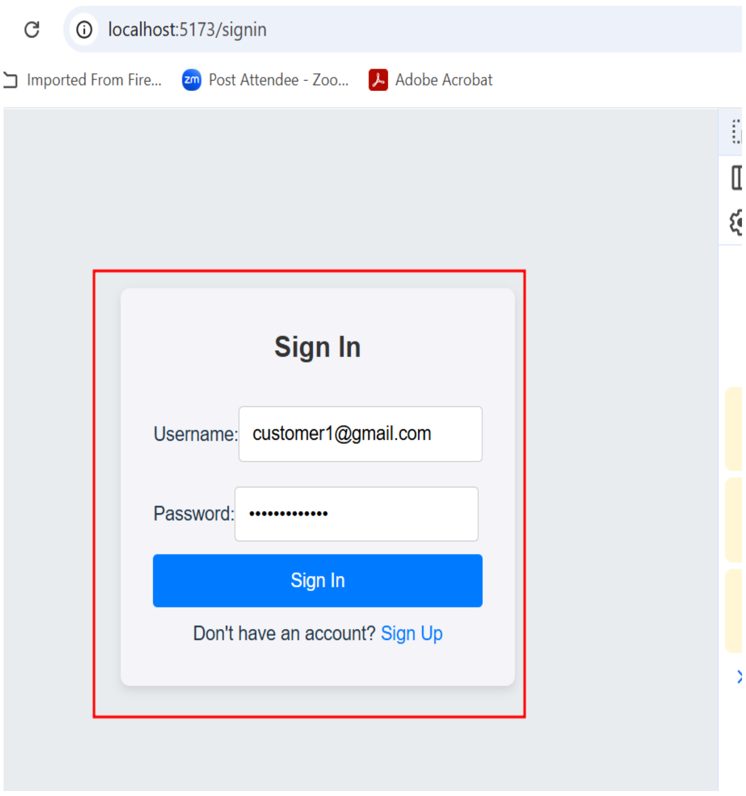
4.24 after account created successfully


localhost:5173/dashboard

Imported From Fire... Post Attendee - Zoo... Adobe Acrobat

Welcome to the Dashboard

Welcome, admin@gmail.com!


localhost:5173/signin

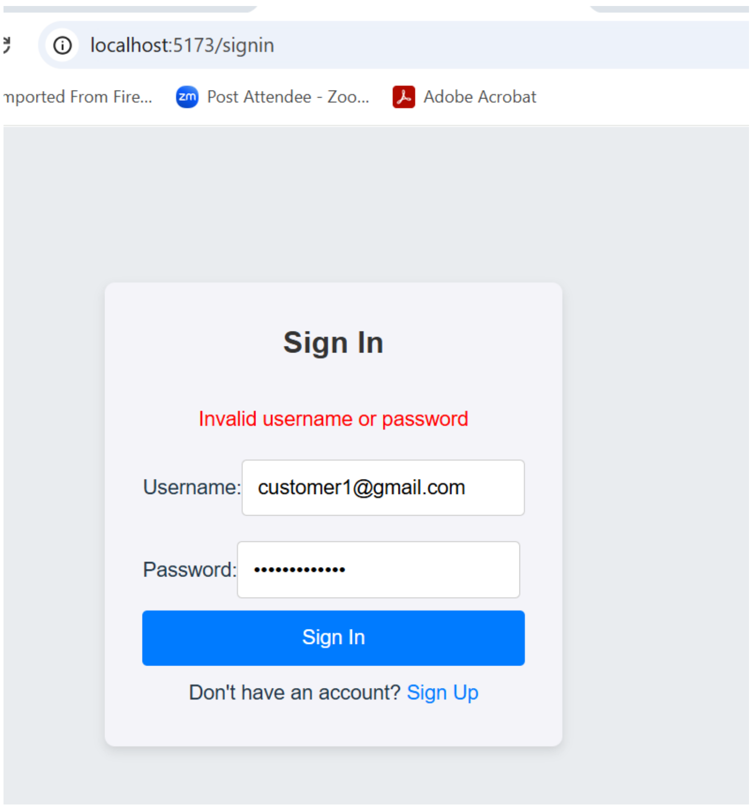
Imported From Fire... Post Attendee - Zoo... Adobe Acrobat

Sign In

Username: customer1@gmail.com

Password: •••••••••••••

Don't have an account? Sign Up


localhost:5173/signin

nported From Fire... Post Attendee - Zoo... Adobe Acrobat

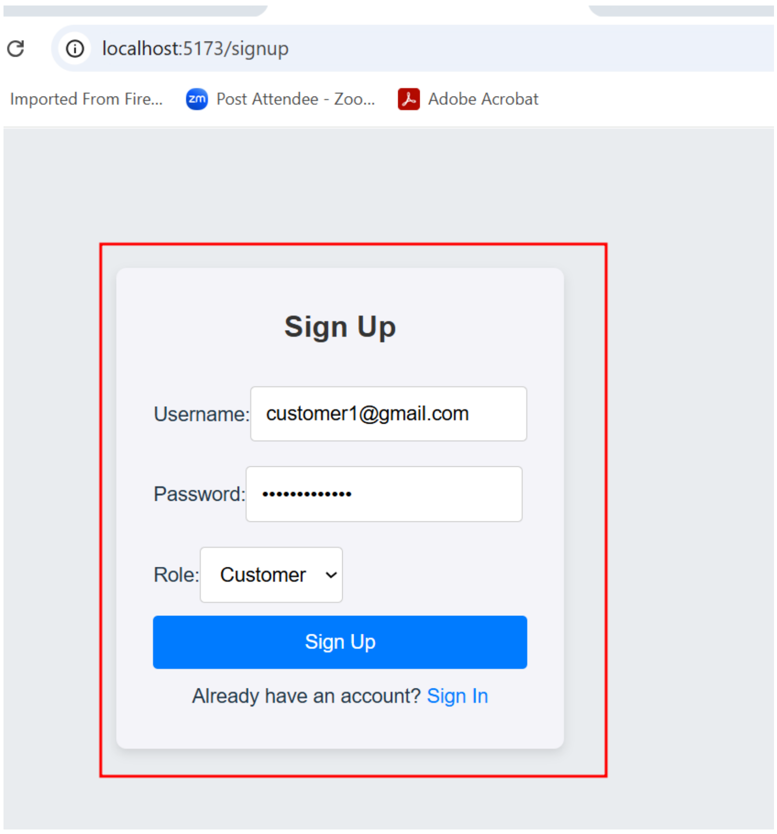
Sign In

Invalid username or password

Username: customer1@gmail.com

Password: •••••••••••••

4.25 create the customer account


localhost:5173/signup

Imported From Fire... Post Attendee - Zoo... Adobe Acrobat

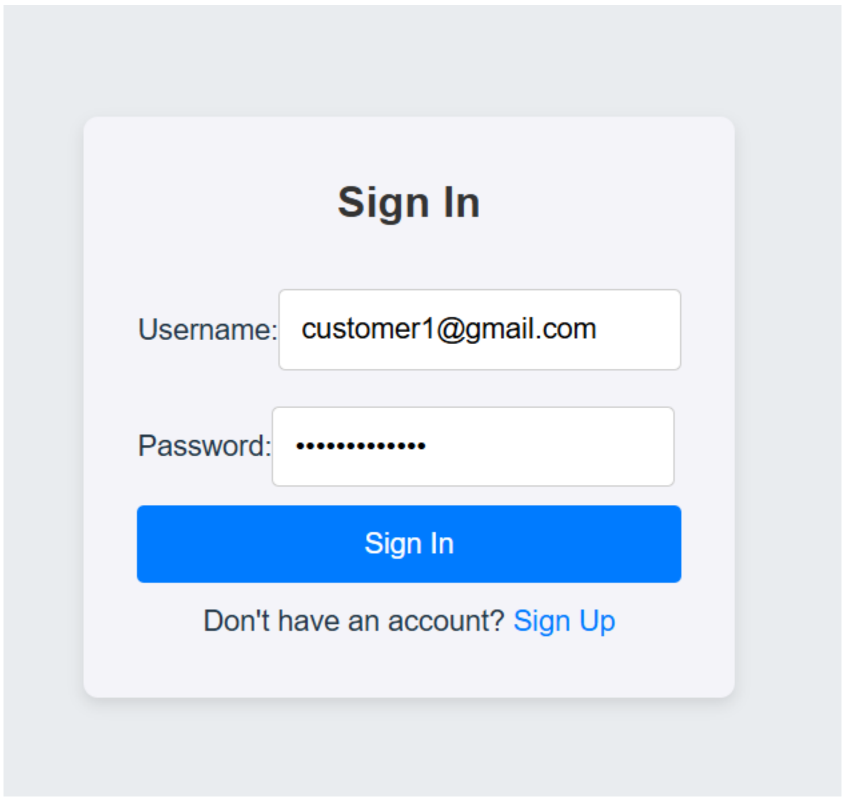
Sign Up

Username: customer1@gmail.com

Password: •••••••••••••

Role: Customer

4.26 Do SignIn for customer login

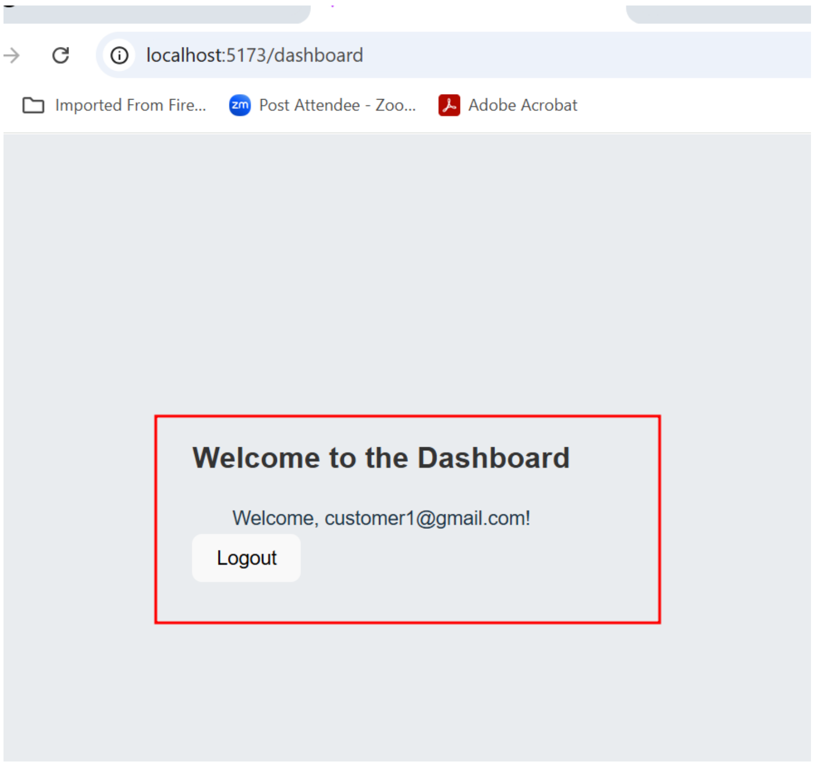

Sign In

Username: customer1@gmail.com

Password: ••••••••••••••

Sign In

Don't have an account? Sign Up


localhost:5173/dashboard

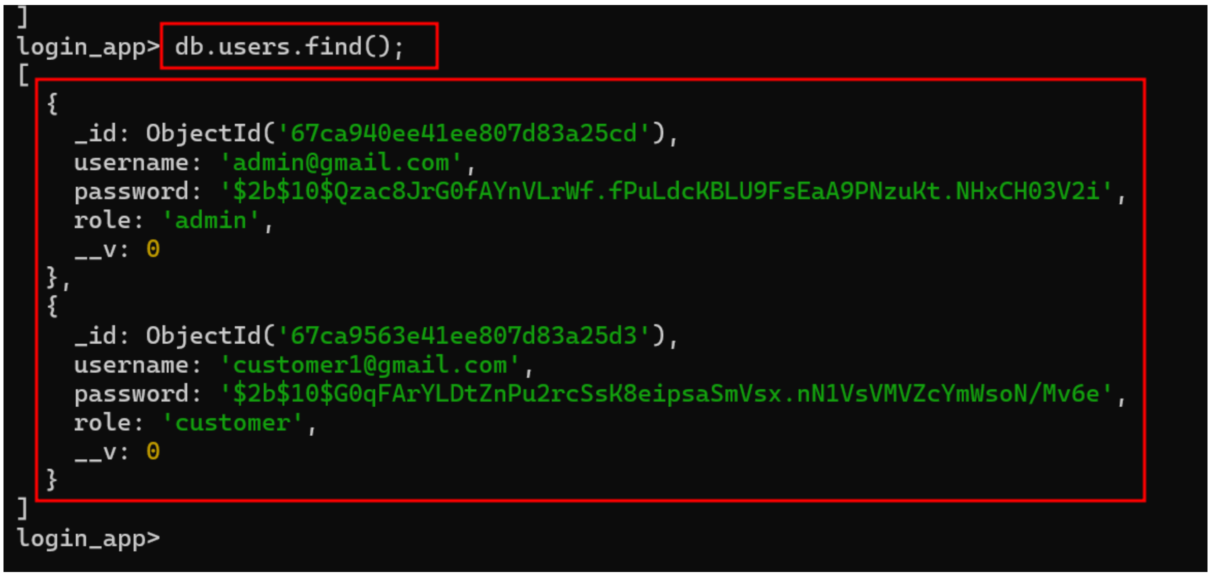
Imported From Fire... Post Attendee - Zoo... Adobe Acrobat

Welcome to the Dashboard

Welcome, customer1@gmail.com!

Logout

4.27 verify tables created in database



```
]
login_app> db.users.find();
[
  {
    _id: ObjectId('67ca940ee41ee807d83a25cd'),
    username: 'admin@gmail.com',
    password: '$2b$10$Qzac8JrG0fAYnVLrWf.fPuLdcKBLU9FsEaA9PNzuKt.NHxCH03V2i',
    role: 'admin',
    __v: 0
  },
  {
    _id: ObjectId('67ca9563e41ee807d83a25d3'),
    username: 'customer1@gmail.com',
    password: '$2b$10$G0qFArYLDtZnPu2rcSsK8eipsaSmVsx.nN1VsVMVZcYmWsoN/Mv6e',
    role: 'customer',
    __v: 0
  }
]
login_app>
```