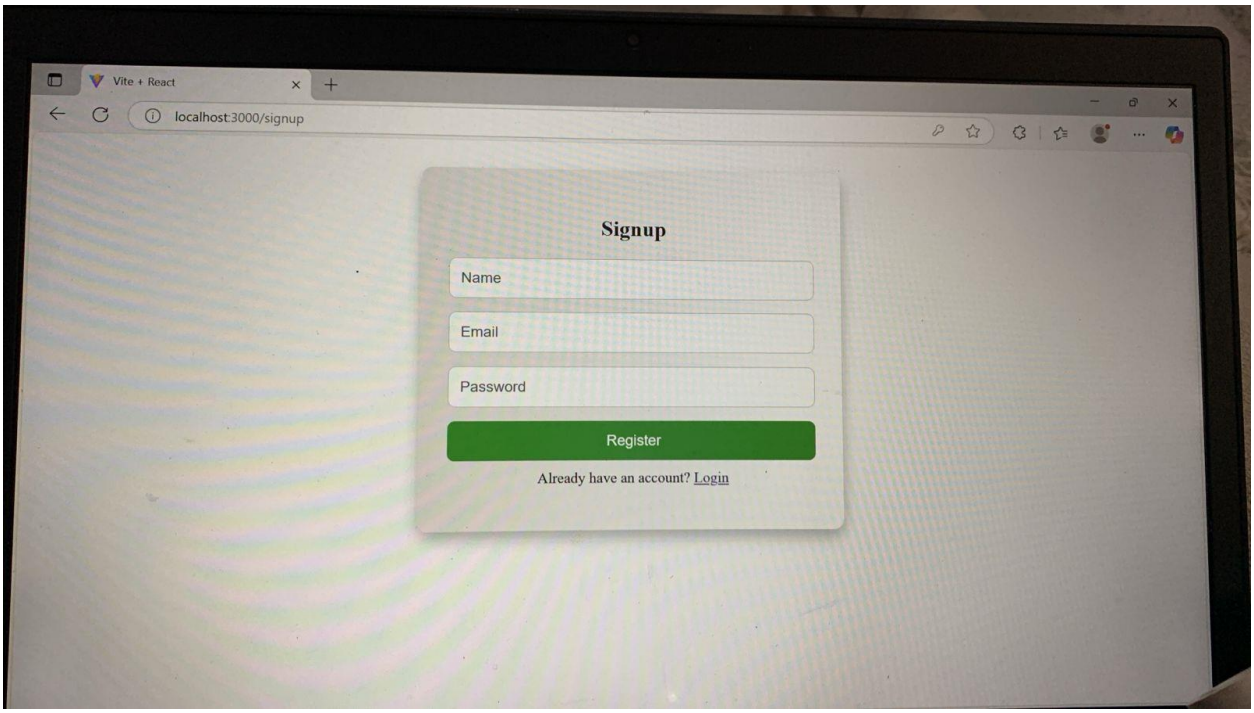
Name Mustafa

Reg sp23-bse-067

Use case signup

Frontened




```
import React, { useState } from "react";
import axios from "axios";
import { toast } from "react-toastify";
import { useNavigate, Link } from "react-router-dom";
import "./Form.css";

function Signup() {
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    password: "",
  });
  const navigate = useNavigate();

  const handleChange = (e) =>
    setFormData({ ...formData, [e.target.name]: e.target.value });
```

```
const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    await axios.post("http://localhost:5000/api/user/register", formData);
    toast.success("Signup successful!");
    setFormData({ name: "", email: "", password: "" });
    navigate("/");
  } catch (error) {
    toast.error(error.response?.data?.message || "Signup failed!");
    setFormData({ name: "", email: "", password: "" });
  }
};

return (
  <form className="form-container" onSubmit={handleSubmit}>
    <h2>Signup</h2>
    <input
      type="text"
      name="name"
      placeholder="Name"
      value={formData.name}
      onChange={handleChange}
      required
    />
    <input
      type="email"
      name="email"
      placeholder="Email"
      value={formData.email}
      onChange={handleChange}
      required
    />
    <input
      type="password"
      name="password"
      placeholder="Password"
      value={formData.password}
      onChange={handleChange}
      required
    />
    <button type="submit">Register</button>
    <p style={{ textAlign: "center", marginTop: "10px" }}>
      Already have an account? <Link to="/">Login</Link>
    </p>
  </form>
```

```
  );
}
export default Signup;
```

LOGIN

```
import React, { useState } from "react";
import axios from "axios";
import { toast } from "react-toastify";
import { Link } from "react-router-dom";
import "./Form.css";

function Login() {
  const [credentials, setCredentials] = useState({ email: "", password: "" });

  const handleChange = (e) =>
    setCredentials({ ...credentials, [e.target.name]: e.target.value });

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      await axios.post("http://localhost:5000/api/user/login", credentials);
      toast.success("Login successful!");
      setCredentials({ email: "", password: "" });
    } catch (error) {
      toast.error(error.response?.data?.message || "Login failed!");
      setCredentials({ email: "", password: "" });
    }
  };

  return (
    <form className="form-container" onSubmit={handleSubmit}>
      <h2>Login</h2>
      <input
        type="email"
        name="email"
        placeholder="Email"
        value={credentials.email}
        onChange={handleChange}
        required
      />
      <input
        type="password"
```

```
        name="password"
        placeholder="Password"
        value={credentials.password}
        onChange={handleChange}
        required
      />
      <button type="submit">Login</button>
      <p style={{ textAlign: "center", marginTop: "10px" }}>
        Don’t have an account? <Link to="/signup">Register</Link>
      </p>
    </form>
  );
}

export default Login;
```

```
.form-container {
    max-width: 400px;
    margin: 30px auto;
    padding: 30px;
    background: linear-gradient(to right, #ece9e6, #ffffff);
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
    border-radius: 12px;
    display: flex;
    flex-direction: column;
}

.form-container h2 {
    text-align: center;
    margin-bottom: 20px;
    color: #333;
}

.form-container input {
    margin-bottom: 15px;
    padding: 12px;
    border: 1px solid #ccc;
    border-radius: 8px;
    font-size: 16px;
}

.form-container button {
    padding: 12px;
```

```
    border: none;
    border-radius: 8px;
    background-color: #4CAF50;
    color: white;
    font-size: 16px;
    cursor: pointer;
    transition: background 0.3s ease;
}

.form-container button:hover {
    background-color: #45a049;
}
```

BACKENED

```
import express from 'express';
import mongoose from 'mongoose';
import dotenv from 'dotenv';
import cors from 'cors';
import UserRoutes from './routes/user.route.js'
dotenv.config();

const app = express();


app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

app.use('/api', UserRoutes);


mongoose.connect(process.env.MONGO_URI)
    .then(() => console.log('Connected to MongoDB'))
    .catch((error) => console.log('Error connecting to MongoDB', error));


app.use((err, req, res, next) => {
    res.status(400).json({ error: err.message });
});


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

ROUTES

```
import express from 'express';
import { User } from '../controller/user.js';

const router = express.Router();

const user = new User();

// Routes
router.post('/user/register', (req, res) => user.register(req, res));
router.post('/user/login', (req, res) => user.login(req, res));

export default router;
```

CONTROLLER

```
import bcrypt from 'bcrypt';
import UserModel from '../model/StudentFeedBack.js';

export class User {
    async register(req, res) {
        const { name, email, password } = req.body;
        if (!name || !email || !password) {
            return res.status(400).json({ message: "All the fields are required"
});
        }
        try {
            const existingUser = await UserModel.findOne({ email });
            if (existingUser) {
                return res.status(409).json({ message: "User is already
registered" });
            }

            const hashedPassword = await bcrypt.hash(password, 10);
            const newUser = new UserModel({ name, email, password: hashedPassword
});
            await newUser.save();

            return res.status(201).json({ message: 'User registered successfully'
});
        } catch (err) {
            return res.status(500).json({ message: "Internal server error" });
        }
    }
```

```
    async login(req, res) {
        const { email, password } = req.body;

        if (!email || !password) {
            return res.status(400).json({ message: 'Email and password are
required' });
        }

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

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

            return res.status(200).json({
                message: "Login successful",
                user, // optional: exclude password before sending
            });
        } catch (err) {
            console.error('Login error:', err);
            return res.status(500).json({ message: 'Internal Server Error' });
        }
    }
}
```

MODEL

```
import mongoose from 'mongoose'
const userSchema = new mongoose.Schema({
    name: {
        type: String,
        required: true,
    },
    email: {
        type: String,
        required: true,
        unique: true
    },
    password: {
        type: String,
        required: true
```

```
    }
}, {
    timestamps: true
});
const User = mongoose.model('User', userSchema);
export default User;
```