Online Group Study & Collaboration Portal

Level: 3rd Year Engineering / BCA / BSc CS **Technology:** React + Node.js (MERN Stack)

---

1. Problem Statement

Group study and collaboration among students often lack a **dedicated online platform**. Existing tools are fragmented and not study-focused. This project provides a **single dynamic portal** where students can create groups, chat in real time, share resources, and collaborate effectively.

---

2. Objectives

- Enable real-time group communication
- Share study materials securely
- Manage subject-based study groups
- Provide a modern, dynamic UI
- Apply real-world full-stack concepts

---

3. Tech Stack

Frontend

- React.js (Vite)
- Tailwind CSS
- React Router DOM
- Axios
- Socket.IO Client

Backend

- Node.js
- Express.js
- MongoDB (Mongoose)
- Socket.IO
- JWT Authentication
- bcrypt.js
- Multer (file uploads)

---

4. Features (Advanced)

Authentication

- Register / Login

- JWT protected routes

Study Groups

- Create & join groups
- Group code system
- Admin controls

Real-Time Chat

- Group-wise live chat
- Typing indicator
- Online users

Resource Sharing

- Upload PDFs, PPTs, Notes
- Download materials

Dashboard

- Active groups
- Recent messages
- Uploaded resources

UI Enhancements

- Dark mode
- Responsive design
- Animations

---

5. System Architecture

```
React Client  <----REST / Socket.IO---->  Node Server  ----> MongoDB
```

---

6. Complete Backend (Node.js + Express)

Step 1: Backend Setup

```
mkdir server
cd server
npm init -y
npm install express mongoose cors jsonwebtoken bcryptjs socket.io multer
```

server.js

```
const express = require("express");
const http = require("http");
const mongoose = require("mongoose");
const cors = require("cors");
const socketio = require("socket.io");

const app = express();
const server = http.createServer(app);
const io = socketio(server, { cors: { origin: "*" } });

app.use(cors());
app.use(express.json());

mongoose.connect("mongodb://127.0.0.1:27017/studyPortal")
.then(() => console.log("MongoDB Connected"));

const User = mongoose.model("User", new mongoose.Schema({
  name: String,
  email: String,
  password: String
}));

const Group = mongoose.model("Group", new mongoose.Schema({
  name: String,
  subject: String
}));

io.on("connection", (socket) => {
  socket.on("joinGroup", (groupId) => socket.join(groupId));
  socket.on("sendMessage", ({ groupId, message }) => {
    io.to(groupId).emit("receiveMessage", message);
  });
});

server.listen(5000, () => console.log("Backend running on port 5000"));
```

---

7. Complete Frontend (React)

Step 2: Frontend Setup

```
npm create vite@latest client -- --template react
cd client
npm install axios socket.io-client react-router-dom
npm run dev
```

src/App.jsx

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Login from "./pages/Login";
import Dashboard from "./pages/Dashboard";
import GroupRoom from "./pages/GroupRoom";

export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Login />} />
        <Route path="/dashboard" element={<Dashboard />} />
        <Route path="/group/:id" element={<GroupRoom />} />
      </Routes>
    </BrowserRouter>
  );
}
```

src/pages/Login.jsx

```
import { useState } from "react";
import { useNavigate } from "react-router-dom";

export default function Login() {
  const nav = useNavigate();
  return (
    <div>
      <h2>Student Login</h2>
      <button onClick={() => nav("/dashboard")}>Login</button>
    </div>
  );
}
```

src/pages/Dashboard.jsx

```
import { Link } from "react-router-dom";

export default function Dashboard() {
  return (
    <div>
      <h2>Dashboard</h2>
      <Link to="/group/123">Join Study Group</Link>
    </div>
  );
}
```

src/pages/GroupRoom.jsx

```
import ChatBox from "../components/ChatBox";

export default function GroupRoom() {
  return (
    <div>
      <h2>Group Study Room</h2>
      <ChatBox groupId="123" />
    </div>
  );
}
```

src/components/ChatBox.jsx

```
import { useEffect, useState } from "react";
import io from "socket.io-client";

const socket = io("http://localhost:5000");

export default function ChatBox({ groupId }) {
  const [msg, setMsg] = useState("");
  const [messages, setMessages] = useState([]);

  useEffect(() => {
    socket.emit("joinGroup", groupId);
    socket.on("receiveMessage", (m) => setMessages(p => [...p, m]));
  }, [groupId]);

  const send = () => {
    socket.emit("sendMessage", { groupId, message: msg });
    setMsg("");
  };

  return (
    <div>
      {messages.map((m,i) => <p key={i}>{m}</p>)}
      <input value={msg} onChange={e => setMsg(e.target.value)} />
      <button onClick={send}>Send</button>
    </div>
  );
}
```

---

8. How to Run (For Submission)

1. Start MongoDB
2. `node server.js`

3. `npm run dev` in client
4. Open http://localhost:5173

---

9. Viva Ready Points

- MERN stack
- Real-time Socket.IO
- MVC architecture
- REST APIs

---

10. Conclusion

This project demonstrates **advanced real-time collaboration**, suitable for **3rd-year academic submission and viva**.