Experiment No - 08

Roll No.	52
Name	Kshitij Nangare
Class	D15B
Subject	Full Stack Development
Lab Outcome	L5
Date of Performance / Submission	06/10/2025 13/10/2025
Signature & Grades	

Experiment 8

Aim : Enable real-time communication via WebSockets

Code :

```
import { Server } from "socket.io";  311.2k (gzipped: 66.7k)
import http from "http";
import express from "express";

const app = express();
const server = http.createServer(app);

const io = new Server(server, {
  cors: {
    origin: ["http://localhost:5173"],
  },
});

export function getReceiverSocketId(userId) {
  return userSocketMap[userId];
}

// used to store online users
const userSocketMap = {}; // {userId: socketId}

io.on("connection", (socket) => {

  const userId = socket.handshake.query.userId;
  if (userId) userSocketMap[userId] = socket.id;

  // io.emit() is used to send events to all the connected clients
  io.emit("getOnlineUsers", Object.keys(userSocketMap));

  socket.on("disconnect", () => {
    delete userSocketMap[userId];
    io.emit("getOnlineUsers", Object.keys(userSocketMap));
  });
});

export { io, app, server };
```

Figure 1

C:\Users\Roshan Yadav\jupyter_python\ChatApp\backend\src\lib\util.js (*preview*)

```
import jwt from "jsonwebtoken"  54.1k (gzipped: 16.2k)

export const generateToken = (userId,res) =>{

    const token = jwt.sign({userId},process.env.JWT_SECRET,{
        expiresIn:"7d"
    })

    res.cookie("jwt",token,{
        maxAge:7*24*60*60*1000, // in milisecond 7 days
        httpOnly:true,
        sameSite:"strict",
        secure:process.env.NODE_ENV !== "development",
    })

    return token
}
```

Figure 2

backend > src > lib > JS cloudinary.js > ...

```
import {v2 as cloudinary} from "cloudinary";  193.1k (gzipped: 61.7k)

import {config} from "dotenv";  6.8k (gzipped: 2.9k)

config();

cloudinary.config({
    cloud_name: process.env.CLOUDINARY_CLOUD_NAME,
    api_key: process.env.CLOUDINARY_API_KEY,
    api_secret: process.env.CLOUDINARY_API_SECRET,
})

export default cloudinary;
```

Figure 3


backend > src > controllers > auth.controler.js > ...
export const signup = async (req,res) => {
const {fullName,email,password} = req.body;
try{
// hash password
if (!fullName || !email || !password){
return res.status(400).json({message:"All fields are required"});
}
if (password.length < 6){
return res.status(400).json({message:"Password must be at leat 6 characters"});
}
const user = await User.findOne({email});
if (user) return res.status(400).json({message:"Email already exists"});
const salt = await bcrypt.genSalt(10);
const hashedPassword = await bcrypt.hash(password,salt);
const newUser = new User({
fullName:fullName,
email:email,
password:hashedPassword,
})
if(newUser){
//generate jwt token
generateToken(newUser._id,res)
await newUser.save();
res.status(201).json({
_id: newUser._id,
email: newUser.email,
profilePic: newUser.profilePic,
})
}
else{
res.status(400).json({message:"Invalid user data"});
}
}catch(err){
console.log("Error in signup controller",err.message);
res.status(500).json({message:"Internal Server Error"})
}
};

Figure 4

backend > src > controllers > auth.controler.js > ...

```
49
50  export const login = async (req,res) => {
51      const {email,password} = req.body
52      try{
53          const user = await User.findOne({email})
54
55          if (!user){
56              return res.status(400).json({message:"Invalid email or password"});
57          }
58
59          const isPasswordCorrect = await bcrypt.compare(password, user.password)
60
61          if (!isPasswordCorrect){
62              return res.status(400).json({message:"Invalid email or password"});
63          }
64
65          generateToken(user._id,res)
66
67          res.status(200).json({
68              _id: user._id,
69              fullName: user.fullName,
70              email: user.email,
71              profilePic: user.profilePic,
72          })
73      } catch(err){
74          console.log("Error in signup controller",err.message);
75          res.status(500).json({message:"Internal Server Error"});
76      }
77  };
78
79  export const logout = (req,res) => {
80      try{
81          res.cookie("jwt","",{maxAge:0});
82          res.status(200).json({message: "Logged out successfully"});
83      } catch(err){
84          console.log("Error in signout controller",err.message);
85          res.status(500).json({message:"Internal Server Error"});
86      }
87  };
```

Figure 5

backend > src > controllers > message.controler.js > ...

```
import cloudinary from "../lib/cloudinary.js";
import { getReceiverSocketId, io } from "../lib/socket.js";
import Message from "../models/message.model.js";
import User from "../models/user.model.js";

export const getUsersForSidebar = async (req,res) => {
    try{
        const loggedInUserId = req.user._id;
        const filteredUsers = await User.find({_id: {$ne: loggedInUserId}}).select("-password");

        res.status(200).json(filteredUsers)
    } catch(err){
        console.log("Error in getUserForSidebar: ",err.message);
        res.status(500).json({err: "Internal server error"});
    }
}

export const getMessages = async (req,res) => {
    try{
        const {id:userToChatId} = req.params;
        const myId = req.user._id;

        const messages = await Message.find({
            $or: [
                {senderId: myId, receiverId:userToChatId},
                {senderId: userToChatId, receiverId: myId},
            ],
        })

        res.status(200).json(messages)
    }catch(err){
        console.log("Error in getMessages: ",err.message);
        res.status(500).json({err: "Internal server error"});
    }
}

```

Figure 6

```
export const sendMessage = async (req,res) => {
    try{
        const {text, image} = req.body;
        const {id: receiverId} = req.params;
        const senderId = req.user._id;

        let imageUrl;

        if(image) {
            // upload image to cloudinary
            const uploadResponse = await cloudinary.uploader.upload(image);
            imageUrl = uploadResponse.secure_url
        }

        const newMessage = new Message({
            senderId,
            receiverId,
            text,
            image: imageUrl,
          });

        await newMessage.save();

        const recieverSocketId = getReceiverSocketId(receiverId)
        if(recieverSocketId){
            io.to(recieverSocketId).emit("newMessage", newMessage)
        }

        res.status(201).json(newMessage);

    }catch(err){
        console.log("Error in sendMessages: ",err.message);
        res.status(500).json({err: "Internal server error"});
    }
}
```

Figure 7

frontend > src > store > useChatStore.js > ...

```
1  import { create } from "zustand";  822 (gzipped: 466)
2  import toast from "react-hot-toast";  8.6k (gzipped: 3.4k)
3  import { axiosInstance } from "../lib/axios";
4  import { useAuthStore } from "./useAuthStore";
5
6  export const useChatStore = create((set, get) => ({
7    messages: [],
8    users: [],
9    selectedUser: null,
10   isUsersLoading: false,
11   isMessagesLoading: false,
12
13   getUsers: async () => {
14     set({ isUsersLoading: true });
15     try {
16       const res = await axiosInstance.get("/messages/users");
17       set({ users: res.data });
18     } catch (error) {
19       toast.error(error.response.data.message);
20     } finally {
21       set({ isUsersLoading: false });
22     }
23   },
24
25   getMessages: async (userId) => {
26     set({ isMessagesLoading: true });
27     try {
28       const res = await axiosInstance.get(`/messages/${userId}`);
29       set({ messages: res.data });
30     } catch (error) {
31       toast.error(error.response.data.message);
32     } finally {
33       set({ isMessagesLoading: false });
34     }
35   },
36   sendMessage: async (messageData) => {
37     const { selectedUser, messages } = get();
38     try {
39       const res = await axiosInstance.post(`/messages/send/${selectedUser._id}`, messageData);
40       set({ messages: [...messages, res.data] });
41     } catch (error) {
42       toast.error(error.response.data.message);
43     }
44   },
45
```

Figure 8

```
  subscribeToMessages: () => {
    const { selectedUser } = get();
    if (!selectedUser) return;

    const socket = useAuthStore.getState().socket;

    socket.on("newMessage", (newMessage) => {
      const isMessageSentFromSelectedUser = newMessage.senderId === selectedUser._id;
      if (!isMessageSentFromSelectedUser) return;

      set({
        messages: [...get().messages, newMessage],
      });
    });
  },

  unsubscribeFromMessages: () => {
    const socket = useAuthStore.getState().socket;
    socket.off("newMessage");
  },

  setSelectedUser: (selectedUser) => set({ selectedUser }),
}));
```

Figure 9

frontend > src > App.jsx > ...

```jsx
import { Routes, Route} from "react-router-dom"  224.9k (gzipped: 71k)
import Navbar from "./components/Navbar"
import HomePage from "./pages/HomePage"
import SignUpPage from "./pages/SignUpPage"
import LoginPage from "./pages/LoginPage"
import SettingsPage from "./pages/SettingsPage"
import ProfilePage from "./pages/ProfilePage"
import { useAuthStore } from "./store/useAuthStore"
import { useEffect } from "react";  4.2k (gzipped: 1.9k)
import { Loader } from "lucide-react"  1.5k (gzipped: 856)
import { Navigate } from "react-router-dom"  224.9k (gzipped: 71k)
import { Toaster } from "react-hot-toast"  11.6k (gzipped: 4.6k)

function App() {

  const {authUser,checkAuth,isCheckingAuth,onlineUsers} = useAuthStore();

  useEffect(() => {
    checkAuth()
  },[checkAuth]);

  if(isCheckingAuth && !authUser) return (
    <div className="flex items-center justify-center h-screen">
      <Loader className="size-10 animate-spin"/>
    </div>
  )

  return (
    <>
      <div>
        <Navbar/>

        <Routes>
          <Route path="/" element={authUser ? <HomePage/> : <Navigate to="/login"/>}/>
          <Route path="/signup" element={!authUser ? <SignUpPage/> : <Navigate to="/"/>}/>
          <Route path="/login" element={!authUser ? <LoginPage/> : <Navigate to="/"/>}/>
          <Route path="/settings" element={<SettingsPage/>}/>
          <Route path="/profile" element={authUser ? <ProfilePage/> : <Navigate to="/login"/>}/>
        </Routes>

        <Toaster/>
      </div>
    </>
  )
}
```

Figure 10

frontend > src > store > useAuthStore.js > ...

```
1  import { create } from "zustand";  822 (gzipped: 466)
2  import { axiosInstance } from "../lib/axios.js";
3  import toast from "react-hot-toast";  8.6k (gzipped: 3.4k)
4  import { io } from "socket.io-client";  105.3k (gzipped: 31.4k)
5
6  const BASE_URL = import.meta.env.MODE === "development" ? "http://localhost:5001" : "/";
7
8  export const useAuthStore = create((set, get) => ({
9    authUser: null,
10   isSigningUp: false,
11   isLoggingIn: false,
12   isUpdatingProfile: false,
13   isCheckingAuth: true,
14   onlineUsers: [],
15   socket: null,
16
17   checkAuth: async () => {
18     try {
19       const res = await axiosInstance.get("/auth/check");
20
21       set({ authUser: res.data });
22       get().connectSocket();
23     } catch (error) {
24       console.log("Error in checkAuth:", error);
25       set({ authUser: null });
26     } finally {
27       set({ isCheckingAuth: false });
28     }
29   },
30
31   signup: async (data) => {
32     set({ isSigningUp: true });
33     try {
34       const res = await axiosInstance.post("/auth/signup", data);
35       set({ authUser: res.data });
36       toast.success("Account created successfully");
37       get().connectSocket();
38     } catch (error) {
39       toast.error(error.response.data.message);
40     } finally {
41       set({ isSigningUp: false });
42     }
43   },
```

Figure 11

```
login: async (data) => {
  set({ isLoggingIn: true });
  try {
    const res = await axiosInstance.post("/auth/login", data);
    set({ authUser: res.data });
    toast.success("Logged in successfully");

    get().connectSocket();
  } catch (error) {
    toast.error(error.response.data.message);
  } finally {
    set({ isLoggingIn: false });
  }
},

logout: async () => {
  try {
    await axiosInstance.post("/auth/logout");
    set({ authUser: null });
    toast.success("Logged out successfully");
    get().disconnectSocket();
  } catch (error) {
    toast.error(error.response.data.message);
  }
},

updateProfile: async (data) => {
  set({ isUpdatingProfile: true });
  try {
    const res = await axiosInstance.put("/auth/update-profile", data);
    set({ authUser: res.data });
    toast.success("Profile updated successfully");
  } catch (error) {
    console.log("error in update profile:", error);
    toast.error(error.response.data.message);
  } finally {
    set({ isUpdatingProfile: false });
  }
},
```

Figure 12


frontend > src > store > useAuthStore.js > ...
8 export const useAuthStore = create((set, get) => ({
71 updateProfile: async (data) => {
77 } catch (error) {
78 console.log("error in update profile:", error);
79 toast.error(error.response.data.message);
80 } finally {
81 set({ isUpdatingProfile: false });
82 }
83 },
84
85 connectSocket: () => {
86 const { authUser } = get();
87 if (!authUser || get().socket?.connected) return;
88
89 const socket = io(BASE_URL, {
90 query: {
91 userId: authUser._id,
92 },
93 });
94 socket.connect();
95
96 set({ socket: socket });
97
98 socket.on("getOnlineUsers", (userIds) => {
99 set({ onlineUsers: userIds });
100 });
101 },
102 disconnectSocket: () => {
103 if (get().socket?.connected) get().socket.disconnect();
104 },
105 }));

Figure 13

frontend > src > components > ChatContainer.jsx > ...

```jsx
import { useChatStore } from "../store/useChatStore";
import { useEffect, useRef } from "react";  4.3k (gzipped: 1.9k)

import ChatHeader from "./ChatHeader";
import MessageInput from "./MessageInput";
import MessageSkeleton from "./skeletons/MessageSkeleton";
import { useAuthStore } from "../store/useAuthStore";
import { formatMessageTime } from "../lib/utils";

const ChatContainer = () => {
  const {
    messages,
    getMessages,
    isMessagesLoading,
    selectedUser,
    subscribeToMessages,
    unsubscribeFromMessages,
  } = useChatStore();
  const { authUser } = useAuthStore();
  const messageEndRef = useRef(null);

  useEffect(() => {
    getMessages(selectedUser._id);

    subscribeToMessages();

    return () => unsubscribeFromMessages();
  }, [selectedUser._id, getMessages, subscribeToMessages, unsubscribeFromMessages]);

  useEffect(() => {
    if (messageEndRef.current && messages) {
      messageEndRef.current.scrollIntoView({ behavior: "smooth" });
    }
  }, [messages]);

  if (isMessagesLoading) {
    return (
      <div className="flex-1 flex flex-col overflow-auto">
        <ChatHeader />
        <MessageSkeleton />
        <MessageInput />
      </div>
    );
  }

```

```jsx
const ChatContainer = () => {
  return (
    <div className="flex-1 flex flex-col overflow-auto">
      <ChatHeader />

      <div className="flex-1 overflow-y-auto p-4 space-y-4">
        {messages.map((message,index) => (
          <div
            key={message._id || index}
            className={`chat ${message.senderId === authUser._id ? "chat-end" : "chat-start"}`}
            ref={messageEndRef}
          >
            <div className=" chat-image avatar">
              <div className="size-10 rounded-full border">
                <img
                  src={
                    message.senderId === authUser._id
                      ? authUser.profilePic || "/avatar.png"
                      : selectedUser.profilePic || "/avatar.png"
                  }
                  alt="profile pic"
                />
              </div>
            </div>
            <div className="chat-header mb-1">
              <time className="text-xs opacity-50 ml-1">
                {formatMessageTime(message.createdAt)}
              </time>
            </div>
            <div className="chat-bubble flex flex-col">
              {message.image && (
                <img
                  src={message.image}
                  alt="Attachment"
                  className="sm:max-w-[200px] rounded-md mb-2"
                />
              )}
              {message.text && <p>{message.text}</p>}
            </div>
          </div>
        ))}
      </div>

      <MessageInput />
    </div>
```

Figure 14

Output :

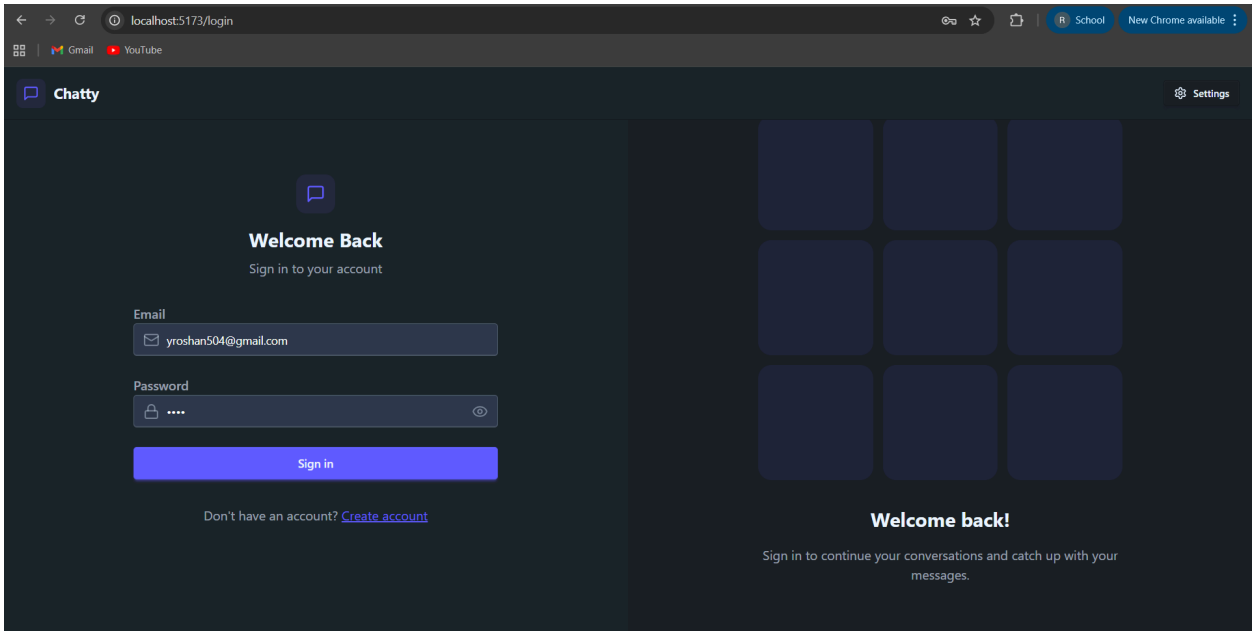

Figure 15

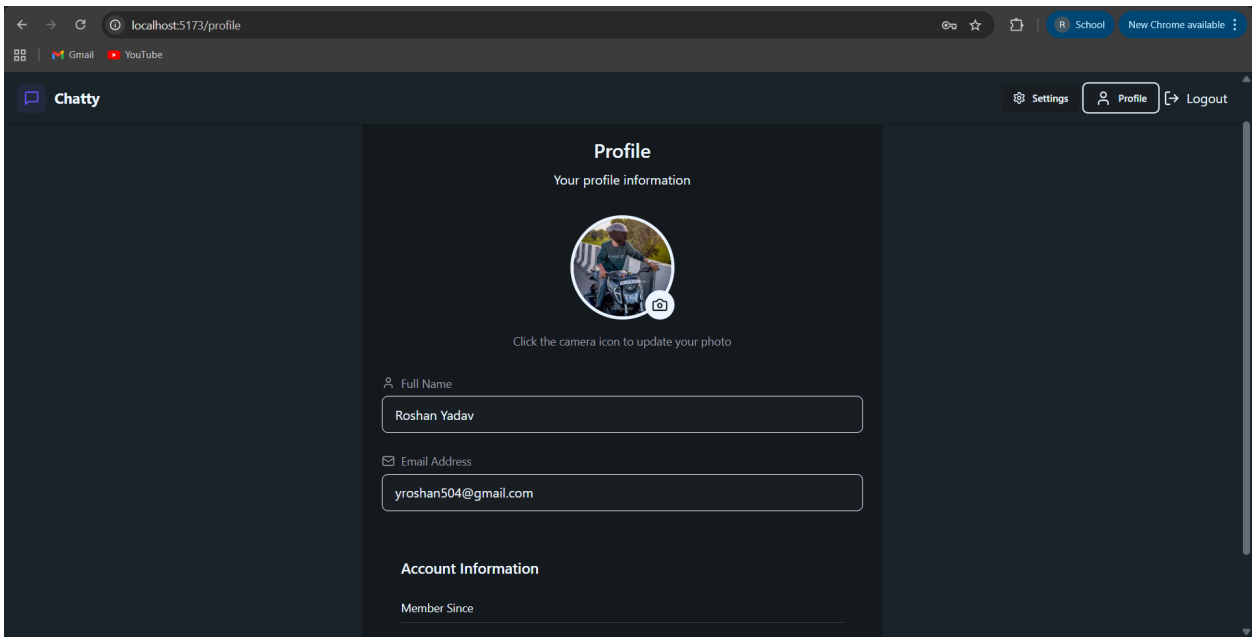

Figure 16

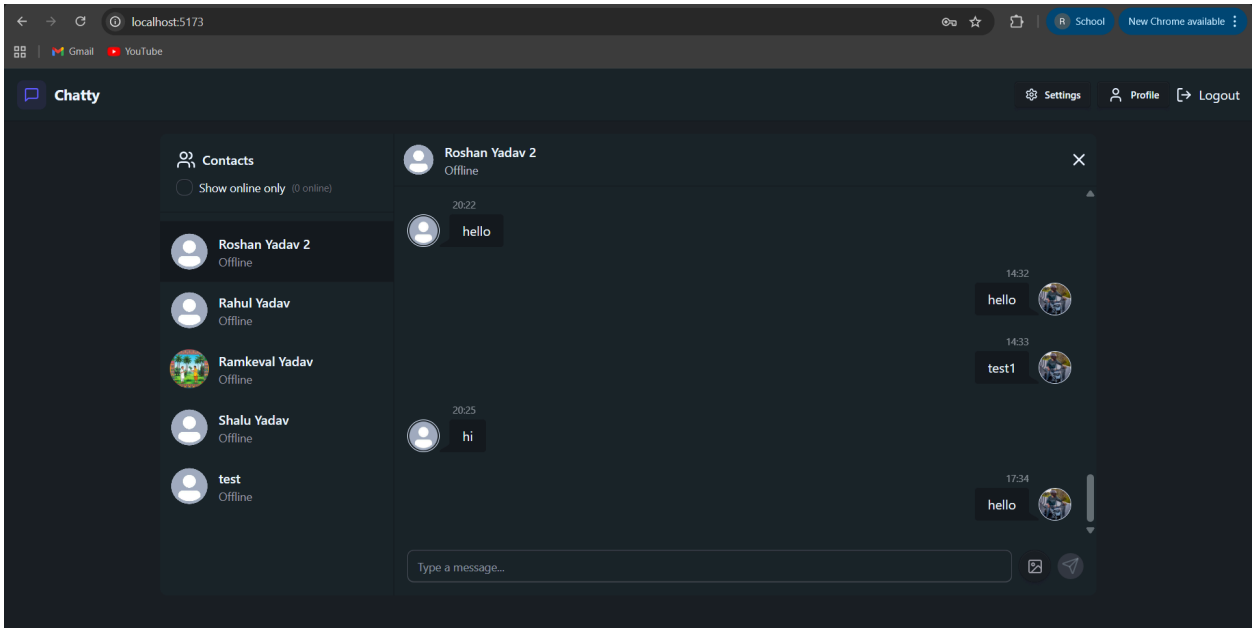
localhost:5173
School
New Chrome available
Gmail
YouTube
Chatty
Settings
Profile
Logout
Contacts
Show online only (0 online)
Roshan Yadav 2
Offline
Rahul Yadav
Offline
Ramkeval Yadav
Offline
Shalu Yadav
Offline
test
Offline
Roshan Yadav 2
Offline
20:22
hello
14:32
hello
14:33
test1
20:25
hi
17:34
hello
Type a message...

Figure 17