Backend Development

1.1. Setup Express Server

```
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');
const mongoose = require('mongoose');
const cors = require('cors');
require('dotenv').config();

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

const PORT = process.env.PORT || 5000;
const MONGO_URI = process.env.MONGO_URI;

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

// MongoDB Connection
mongoose.connect(MONGO_URI, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => console.log('MongoDB connected'))
  .catch(err => console.error(err));

// Socket.io Setup
io.on('connection', (socket) => {
  console.log('New client connected');

  socket.on('joinRoom', ({ username, room }) => {
    socket.join(room);
```

```
    socket.to(room).emit('userJoined', username);
    socket.on('chatMessage', (msg) => {
      io.to(room).emit('chatMessage', msg);
    });
    socket.on('disconnect', () => {
      console.log('Client disconnected');
    });
  });
});

// Routes
app.use('/api/auth', require('./routes/auth'));
app.use('/api/rooms', require('./routes/rooms'));

// Start Server
server.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

.env

```
PORT=5000
MONGO_URI=your_mongodb_connection_string
```

1.2. Configure MongoDB

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const UserSchema = new Schema({
  username: { type: String, required: true, unique: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true }
});

module.exports = mongoose.model('User', UserSchema);
```

backend/models/Room.js

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const RoomSchema = new Schema({
  roomId: { type: String, required: true, unique: true },
  participants: [{ type: String }]
});

module.exports = mongoose.model('Room', RoomSchema);
```

1.3. Add Authentication

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

exports.register = async (req, res) => {
  const { username, email, password } = req.body;
  try {
    const hashedPassword = await bcrypt.hash(password, 10);
    const newUser = new User({ username, email, password: hashedPassword });
    await newUser.save();
    res.status(201).send('User created');
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};

exports.login = async (req, res) => {
  const { username, password } = req.body;
  try {
    const user = await User.findOne({ username });
    if (!user || !(await bcrypt.compare(password, user.password))) {
```

```
      return res.status(401).send('Invalid credentials');
    }
    const token = jwt.sign({ id: user._id }, 'your_jwt_secret', { expiresIn: '1h' });
    res.json({ token });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
const express = require('express');
const router = express.Router();
const { register, login } = require('../controllers/authController');

router.post('/register', register);
router.post('/login', login);

module.exports = router;
```

1.4. Create Rooms

```
const Room = require('../models/Room');

exports.createRoom = async (req, res) => {
  const { roomId } = req.body;
  try {
    const newRoom = new Room({ roomId, participants: [] });
    await newRoom.save();
    res.status(201).send('Room created');
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};

exports.joinRoom = async (req, res) => {
```

```
  const { roomId, username } = req.body;
  try {
    const room = await Room.findOne({ roomId });
    if (!room) return res.status(404).send('Room not found');
    room.participants.push(username);
    await room.save();
    res.status(200).send('Joined room');
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```

backend/routes/rooms.js

```
const express = require('express');
const router = express.Router();
const { createRoom, joinRoom } = require('../controllers/roomController');

router.post('/create', createRoom);
router.post('/join', joinRoom);

module.exports = router;
```

2. Frontend Setup (React.js)

2.1. Install Dependencies

```
npm install axios react-router-dom @material-ui/core
```

2.2. Configure Authentication and Rooms

```
import axios from 'axios';

const API_URL = 'http://localhost:5000/api';

export const register = (user) => axios.post(`${API_URL}/auth/register`, user);
export const login = (user) => axios.post(`${API_URL}/auth/login`, user);
export const createRoom = (room) => axios.post(`${API_URL}/rooms/create`, room);
```

```
export const joinRoom = (room) => axios.post(`${API_URL}/rooms/join`, room);
```

src/components/Login.js

```
import React, { useState } from 'react';
import { TextField, Button, Typography } from '@material-ui/core';
import { login } from '../api';
import { useHistory } from 'react-router-dom';

const Login = () => {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const history = useHistory();

  const handleSubmit = async () => {
    try {
      const { data } = await login({ username, password });
      localStorage.setItem('token', data.token);
      history.push('/video-call');
    } catch (error) {
      console.error(error);
    }
  };

  return (
    <div>
      <Typography variant="h4">Login</Typography>
      <TextField label="Username" onChange={(e) => setUsername(e.target.value)} />
      <TextField label="Password" type="password" onChange={(e) => setPassword(e.target.value)} />
      <Button onClick={handleSubmit}>Login</Button>
    </div>
  );
};
```

```
export default Login;
```

src/components/RoomManagement.js

```
import React, { useState } from 'react';
import { createRoom, joinRoom } from '../api';
import { TextField, Button, Typography } from '@material-ui/core';

const RoomManagement = () => {
  const [roomId, setRoomId] = useState('');

  const handleCreateRoom = async () => {
    try {
      await createRoom({ roomId });
      alert('Room created');
    } catch (error) {
      console.error(error);
    }
  };

  const handleJoinRoom = async () => {
    try {
      await joinRoom({ roomId, username: 'user' });
      alert('Joined room');
    } catch (error) {
      console.error(error);
    }
  };

  return (
    <div>
      <Typography variant="h4">Room Management</Typography>
```

```
      <TextField label="Room ID" onChange={(e) => setRoomId(e.target.value)} />
      <Button onClick={handleCreateRoom}>Create Room</Button>
      <Button onClick={handleJoinRoom}>Join Room</Button>
    </div>
  );
};

export default RoomManagement;
```

3. Connecting Frontend and Backend

Ensure that your React frontend and Node.js backend are configured to interact with each other. Make sure you handle CORS on the backend and use Axios for API calls.