Code:

Frontend Folder (frontend/)

1. src/App.js

```
import React, { useState, useEffect } from 'react';
import io from 'socket.io-client';
import Login from './Login';
import Chat from './Chat';

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

function App() {
  const [username, setUsername] = useState('');

  return (
    <div>
      {username ? <Chat username={username} socket={socket} /> : <Login
setUsername={setUsername} />}
    </div>
  );
}

export default App;
```

2. src/Login.js

```
import React, { useState } from 'react';

function Login({ setUsername }) {
  const [name, setName] = useState('');

  const handleLogin = () => {
    if(name.trim() !== '') setUsername(name);
  };

  return (
    <div>
      <h2>Login</h2>
      <input type="text" placeholder="Enter name" value={name}
onChange={(e)=>setName(e.target.value)} />
      <button onClick={handleLogin}>Enter Chat</button>
```

```
    </div>
  );
}

export default Login;
```

3. src/Chat.js

```
import React, { useState, useEffect } from 'react';

function Chat({ username, socket }) {
  const [message, setMessage] = useState('');
  const [messages, setMessages] = useState([]);

  useEffect(() => {
    socket.on('receive_message', (msg) => {
      setMessages(prev => [...prev, msg]);
    });
    return () => socket.off('receive_message');
  }, [socket]);

  const sendMessage = () => {
    if(message.trim() !== '') {
      const msgObj = { text: message, sender: username };
      socket.emit('send_message', msgObj);
      setMessages(prev => [...prev, msgObj]);
      setMessage('');
    }
  };

  return (
    <div>
      <h2>Chat Room</h2>
      <div style={{ border:'1px solid black', height:'300px', overflowY:'scroll' }}>
        {messages.map((msg, idx)=><div key={idx}><b>{msg.sender}:</b>
{msg.text}</div>)}
      </div>
      <input type="text" value={message} onChange={(e)=>setMessage(e.target.value)}
/>
      <button onClick={sendMessage}>Send</button>
    </div>
  );
}

export default Chat;
```

Backend Folder (backend/)

1. server.js

```
const express = require('express');
const http = require('http');
const socketIo = require('socket.io');
const cors = require('cors');
const mongoose = require('mongoose');
const User = require('./models/User');

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

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

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

io.on('connection', (socket) => {
    console.log('User connected:', socket.id);

    socket.on('send_message', (data) => {
        io.emit('receive_message', data); // Broadcast to all users
    });

    socket.on('disconnect', () => {
        console.log('User disconnected:', socket.id);
    });
});

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

2. models/User.js

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

```
const userSchema = new mongoose.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);
```

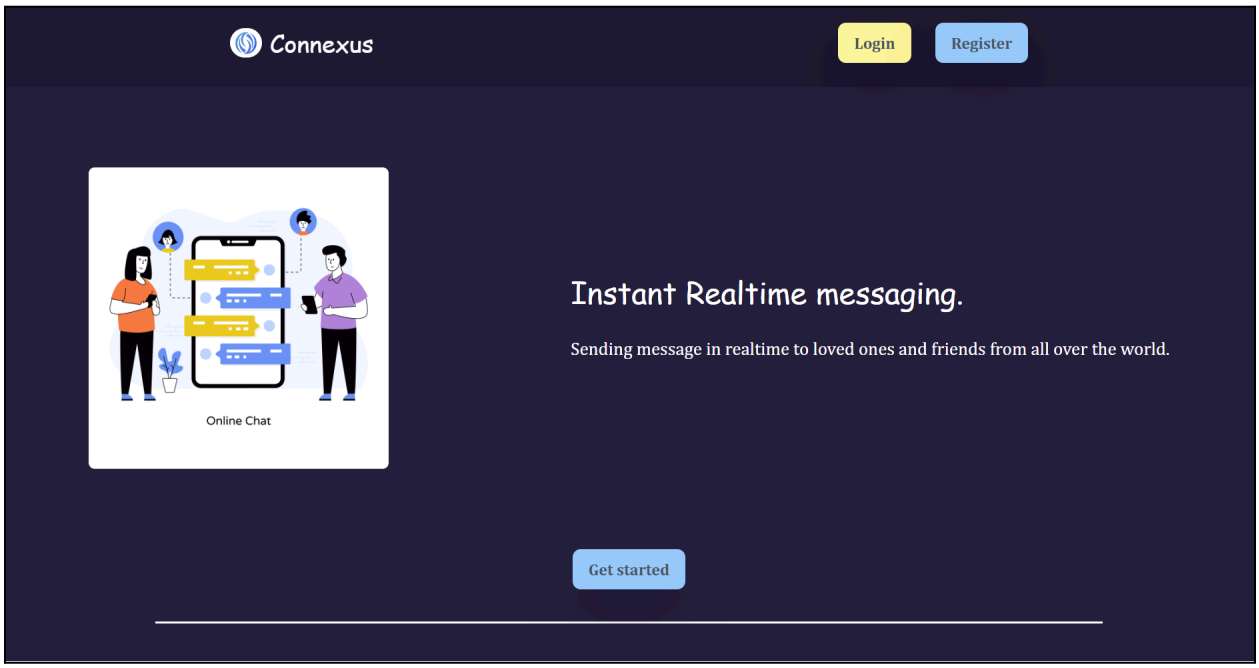
3. routes/auth.js

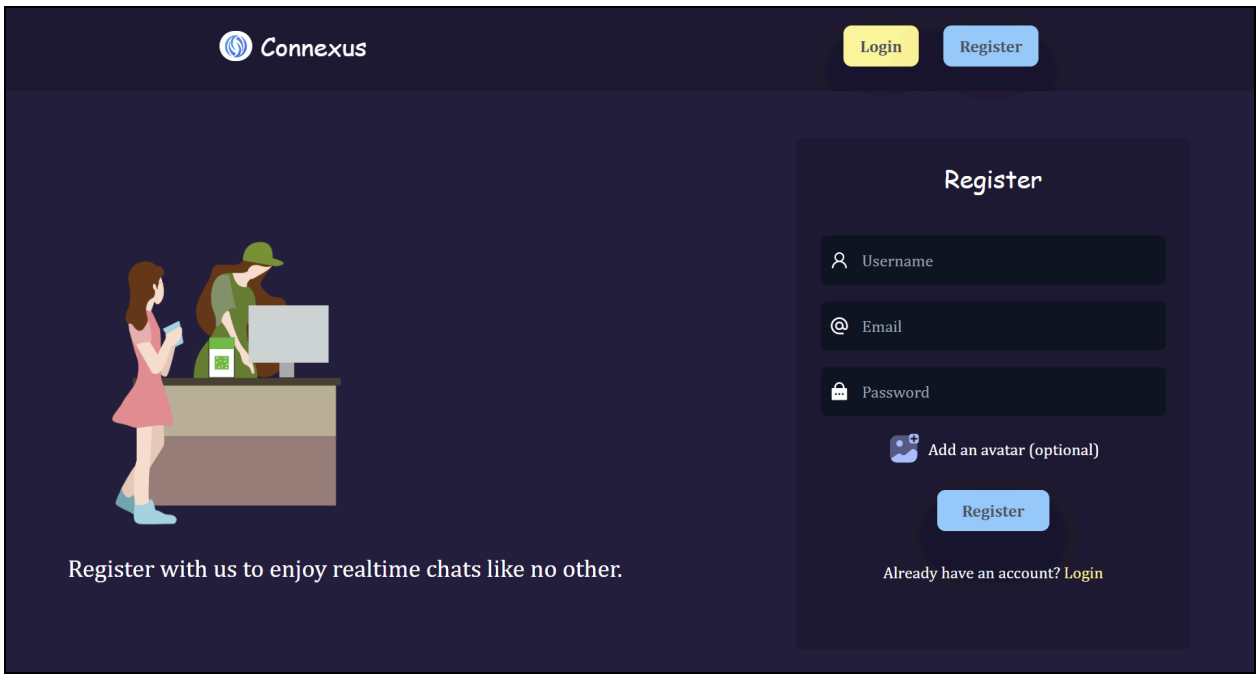
```
const express = require('express');
const router = express.Router();
const User = require('../models/User');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');

// Register
router.post('/register', async (req, res) => {
   const { username, email, password } = req.body;
   const hashedPassword = await bcrypt.hash(password, 10);
   const newUser = new User({ username, email, password: hashedPassword });
   await newUser.save();
   res.status(201).send('User registered');
});

// Login
router.post('/login', async (req, res) => {
   const { email, password } = req.body;
   const user = await User.findOne({ email });
   if(!user) return res.status(400).send('User not found');
   const valid = await bcrypt.compare(password, user.password);
   if(!valid) return res.status(400).send('Invalid credentials');
   const token = jwt.sign({ id: user._id }, 'SECRET_KEY');
   res.json({ token });
});

module.exports = router;
```

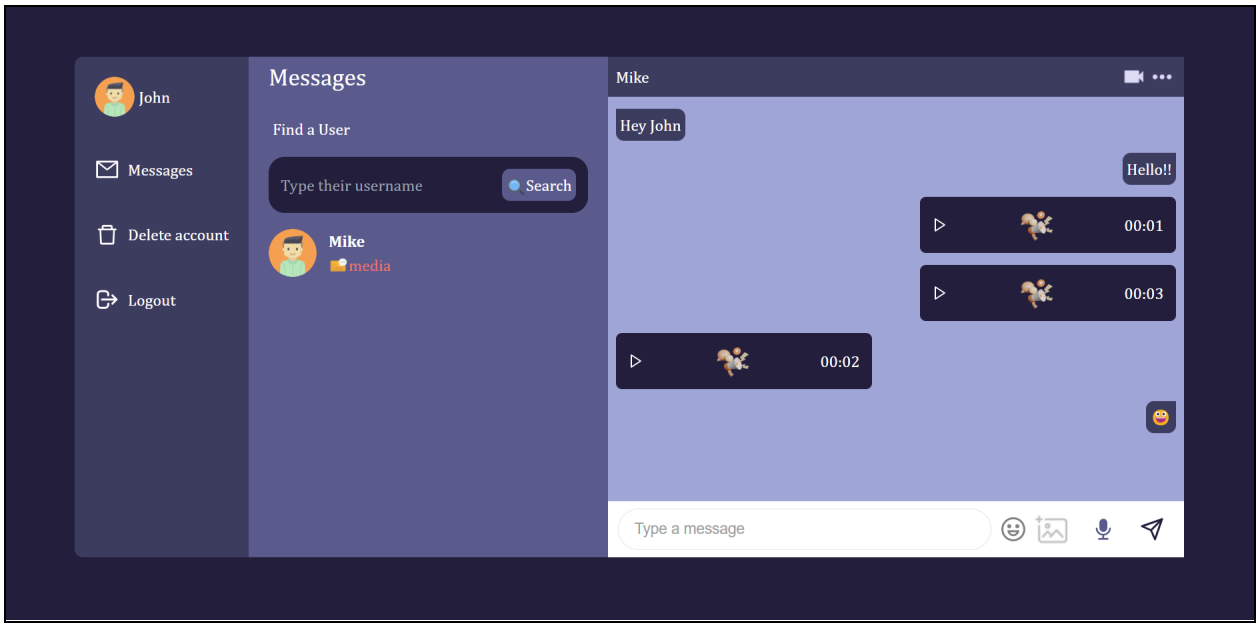
4. backend/.env

```
PORT=5000
DB_URI=your_mongodb_connection_string
SECRET=your_jwt_secret
```

Output:


Connexus
Login
Register
Online Chat
Instant Realtime messaging.
Sending message in realtime to loved ones and friends from all over the world.
Get started


Connexus
Login
Register
Register
Username
Email
Password
Add an avatar (optional)
Register
Already have an account? Login
Register with us to enjoy realtime chats like no other.


John
Messages
Delete account
Logout
Messages
Find a User
Type their username
Search
Mike
media
Mike
Hey John
Hello!!
00:01
00:03
00:02
Type a message