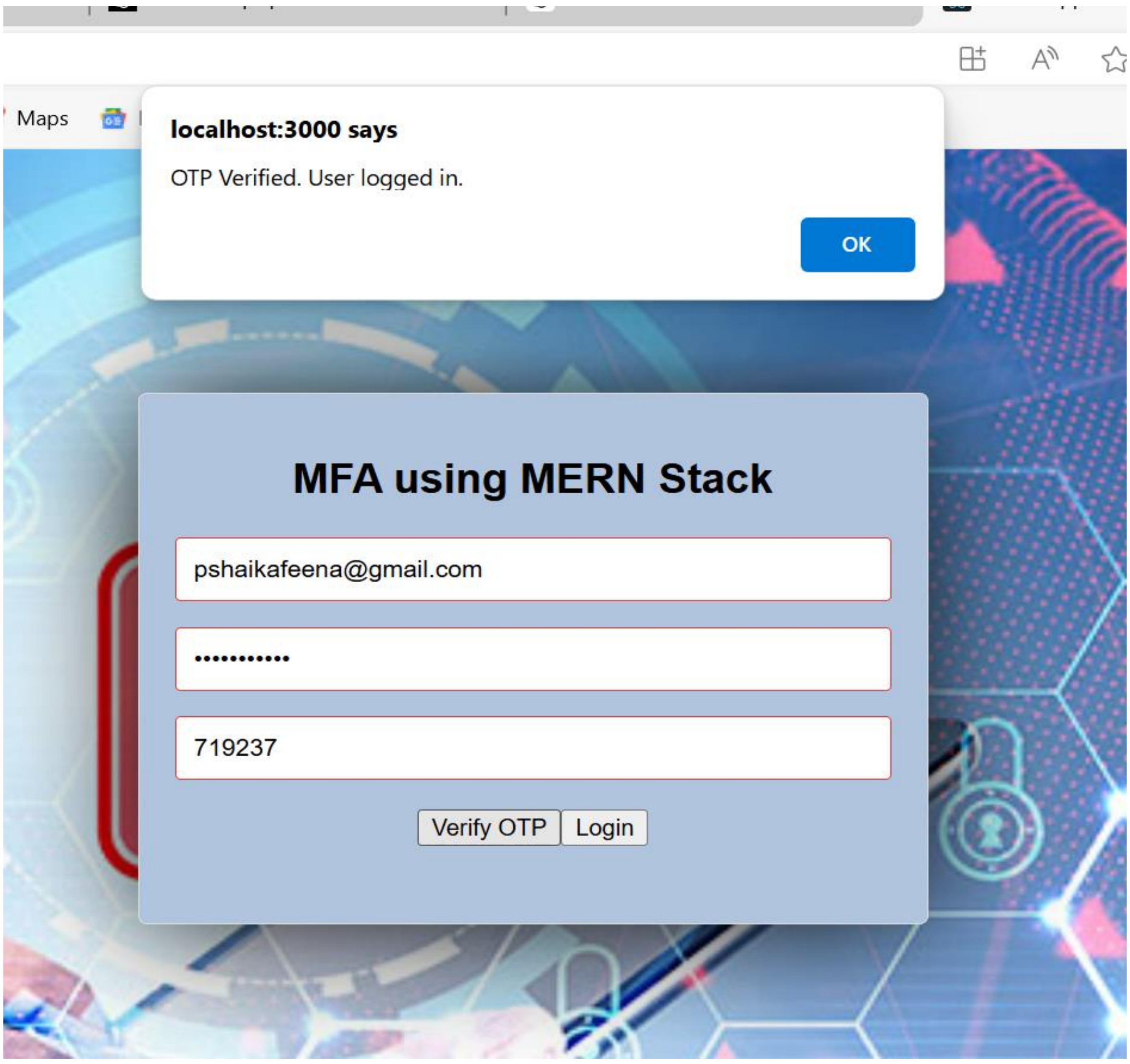
Report and Documentation

of

Multifactor Authentication using MERN

The MFA system integrates with the existing MERN stack application to enhance security by requiring a second layer of authentication through OTP. The process involves the user logging in with credentials, after which an OTP is generated and sent via email using Nodemailer. MongoDB stores user details and OTP for validation, and Express handles routing and backend logic. The front end in React manages the UI/UX for OTP input and login feedback.

Preview of final output: Lets have a look at how the final output will look like.



localhost:3000 says

OTP Verified. User logged in.

MFA using MERN Stack

pshaikafeena@gmail.com

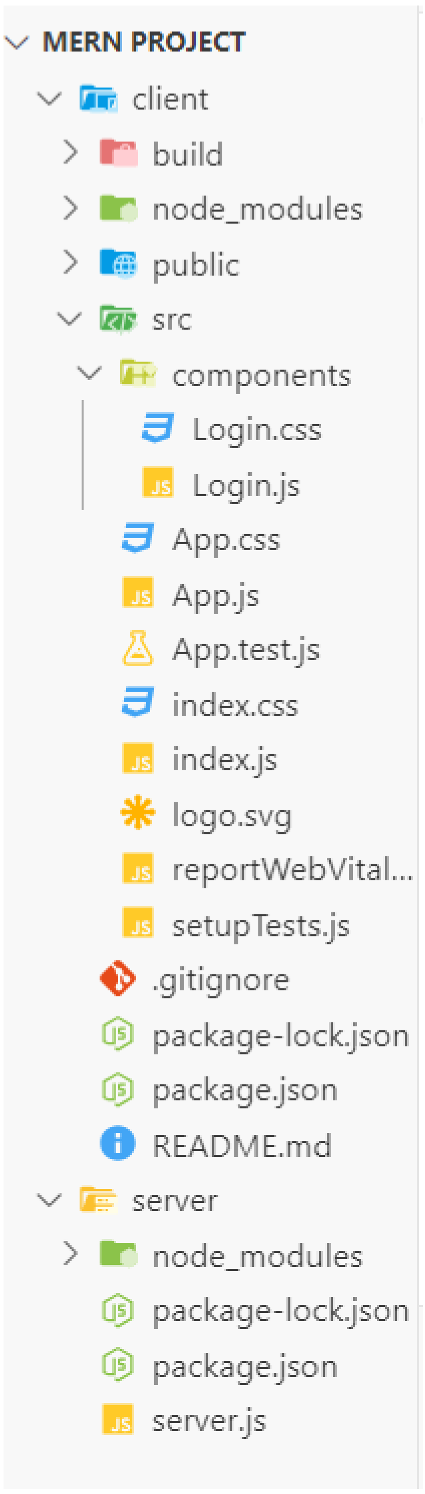
•••••••••••

719237

Approach to create Multifactor Authentication:

The main approach of this project is to Implement user registration and login functionality, ensuring passwords are securely hashed with bcryptjs. Upon a successful login attempt, generate a One-Time Password (OTP) and send it to the user via email using nodemailer. Store the OTP securely in MongoDB for validation. Create an API route to verify the OTP, allowing access only if the OTP is correct. On the frontend, design React components for the login process and OTP entry. Finally, thoroughly test the system and deploy the application.

Project Structure



```
MERN PROJECT
  client
    build
    node_modules
    public
    src
      components
        Login.css
        Login.js
      App.css
      App.js
      App.test.js
      index.css
      index.js
      logo.svg
      reportWebVital...
      setupTests.js
    .gitignore
    package-lock.json
    package.json
    README.md
  server
    node_modules
    package-lock.json
    package.json
    server.js
```

Steps to create the application

Step 1: Open the root directory in vs code and create a folder `server` and initialize the express application.

```
Cd server
npm init -y
```

Step 2: Install the required dependencies

```
npm install express mongoose body-parser cors nodemon
```

Dependencies(Backend):

```
“dependencies": {
   "body-parser": "^1.20.2",
   "cors": "^2.8.5",
   "express": "^4.19.2",
   "mongodb": "^6.8.0",
   "mongoose": "^7.8.0",
   "nodemailer": "^6.9.14",
   "randomatic": "^3.1.1"
 },
```

Example: Create a server.js file, additional files and add required code into it.

//Server.js

```
const express = require('express');
const cors = require('cors');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const nodemailer = require('nodemailer');
const randomize = require('randomatic');
// const bcrypt = require('bcrypt'); // Uncomment if using hashed passwords


const app = express();
const PORT = 3003;


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

// Connect to MongoDB with connection pooling
mongoose.connect('mongodb://localhost:27017/mernproject', {
   useNewUrlParser: true,
   useUnifiedTopology: true,
   // poolSize: 10,
})
.then(() => console.log('MongoDB connected'))
.catch(err => console.log('MongoDB connection error:', err));

// Define User Schema and Model
const userSchema = new mongoose.Schema({
   email: { type: String, required: true, unique: true },
   password: { type: String, required: true }, // Store hashed passwords ideally
   otp: String,
});

const User = mongoose.model('User', userSchema);

// Function to send OTP to the user's email
async function sendOtpEmail(email, otp) {
   try {
      const transporter = nodemailer.createTransport({
         service: 'gmail',
         auth: {
            // Replace with your email and password
            user: 'your-email@gmail.com',
            pass: 'your-password',
            },
        });

        const mailOptions = {
            from: 'your-email@gmail.com',
            to: email,
            subject: 'OTP Verification',
            text: `Your OTP is: ${otp}`,
        };

        const info = await transporter.sendMail(mailOptions);
        console.log('Email sent: ' + info.response);
    } catch (error) {
        console.error('Error sending email:', error);
    }
}

// Login Route
app.post('/auth/login', async (req, res) => {
    const { email, password } = req.body;

    try {
        const user = await User.findOne({ email });
        if (!user) {
            return res.json({
                success: false,
                message: 'Invalid credentials',
            });
        }
```

```
// If using plain text passwords (not recommended)
if (user.password !== password) {
   return res.json({
      success: false,
      message: 'Invalid credentials',
   });
}

// If using hashed passwords, use bcrypt to compare:
// const isMatch = await bcrypt.compare(password, user.password);
// if (!isMatch) {
//    return res.json({
//       success: false,
//       message: 'Invalid credentials',
//    });
// }

// Generate OTP
const generatedOtp = randomize('0', 6);
user.otp = generatedOtp;
await user.save();

// Send OTP Email
await sendOtpEmail(email, generatedOtp);

// Respond with success
return res.json({
   success: true,
            message: 'OTP sent to your email',
        });
    } catch (error) {
        console.error('Error during login:', error.message);
        return res.status(500).json({
            success: false,
            message: 'An error occurred during login',
        });
    }
});

// OTP Verification Route
app.post('/auth/verify-otp', async (req, res) => {
    const { otp } = req.body;

    try {
        const user = await User.findOne({ otp });

        if (!user) {
            return res.json({
                success: false,
                message: 'Invalid OTP',
            });
        }

        // Clear the OTP after successful verification
        user.otp = '';
        await user.save();
        return res.json({
            success: true,
            message: 'OTP verified, login successful',
        });
    } catch (error) {
        console.error('Error during OTP verification:', error.message);
        return res.status(500).json({
            success: false,
            message: 'An error occurred during OTP verification',
        });
    }
});


// Start the Server
app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```

Step 3: To start the server run the following command**.**

```
node server.js
```

Step 4: Open a new terminal in project root directory and run the following command to create react app.

```
 npx create-react-app client
cd client
```

Step 5: Install Axios

```
npm install axios
```

Project dependencies:

```
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.3.4",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.26.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
```

Example: Create the required files and add the below code.

/* src/App.css */

```
/* Container for the form */
.App {
  width: 400px;
  margin: 0 auto;
  padding: 20px;
  border: 1px solid #ddd;
  border-radius: 5px;
  box-shadow: 0px 0px 80px black;
  background-color: lightsteelblue;
  text-align: center;
  justify-content: center;
  height: 270px;
}
/* Headline */
.App h1 {
  font-family: 'Arial', sans-serif;
  font-size: 24px;
  margin-bottom: 20px;
}
```

```css
.App input[type="text"]{
  width:100%;
  padding: 10px;
  margin-bottom: 15px;
  border: 1px solid #d83a3a;
  border-radius: 3px;
  box-sizing: border-box;
  font-size: 14px;
}

/* Input fields */
.App input[type="email"],
.App input[type="password"] {
  width: 100%;
  padding: 10px;
  margin-bottom: 15px;
  border: 1px solid #d83a3a;
  border-radius: 3px;
  box-sizing: border-box;
  font-size: 14px;
}

/* Login button */
#login-button[type="submit"] {
  width: 100%;
  padding: 10px;
  background-color: #4CAF50;
  color: white;
  border: none;
  border-radius: 3px;

  font-size: 16px;

  cursor: pointer;


}


/* Button hover effect */

.button[type="submit"]:hover {

  background-color: #45a049;

}

body {

  margin: 0;

  padding: 200px;

  height: 100vh; /* Full height */

  background-image: url('https://www.foiassist.com.au/wp-
content/uploads/2021/02/service_img8.jpg'); /* Correct URL */

  background-size: cover; /* Cover the entire area */

  background-position: center; /* Center the image */

  background-repeat: no-repeat; /* Prevent repetition */


}
```

// src/App.js

```
// client/src/App.js

import React from 'react';

import Login from './components/Login';

import './App.css'


function App() {

    return (
      <div className="App">
        <div style={centerStyle}>
          <h1>
            MFA using
            MERN Stack
          </h1>
        </div>
        <Login />
      </div>
    );
  }

const centerStyle = {
  textAlign: 'center',
};
export default App;
```

// src/components/Login.css

```
/* Container for the form */
.App {
  width: 400px;
  margin: 0 auto;
  padding: 20px;
  border: 1px solid #ddd;
  border-radius: 5px;
  box-shadow: 0px 0px 80px black;
  background-color: lightsteelblue;
  text-align: center;
  justify-content: center;
  height: 270px;
```

```css
}
/* Headline */
.App h1 {
  font-family: 'Arial', sans-serif;
  font-size: 24px;
  margin-bottom: 20px;
}
.App input[type="text"]{
  width:100%;
  padding: 10px;
  margin-bottom: 15px;
  border: 1px solid #d83a3a;
  border-radius: 3px;
  box-sizing: border-box;
  font-size: 14px;
}

/* Input fields */
.App input[type="email"],
.App input[type="password"] {
  width: 100%;
  padding: 10px;
  margin-bottom: 15px;
  border: 1px solid #d83a3a;
  border-radius: 3px;
  box-sizing: border-box;
  font-size: 14px;
}
```

```
/* Login button */
#login-button[type="submit"] {
  width: 100%;
  padding: 10px;
  background-color: #4CAF50;
  color: white;
  border: none;
  border-radius: 3px;
  font-size: 16px;
  cursor: pointer;

}



/* Button hover effect */
.button[type="submit"]:hover {
  background-color: #45a049;
}
body {
  margin: 0;
  padding: 200px;
  height: 100vh; /* Full height */
  background-image: url('https://www.foiassist.com.au/wp-
content/uploads/2021/02/service_img8.jpg'); /* Correct URL */
  background-size: cover; /* Cover the entire area */
  background-position: center; /* Center the image */
  background-repeat: no-repeat; /* Prevent repetition */


}
```

// src/components/Login.js

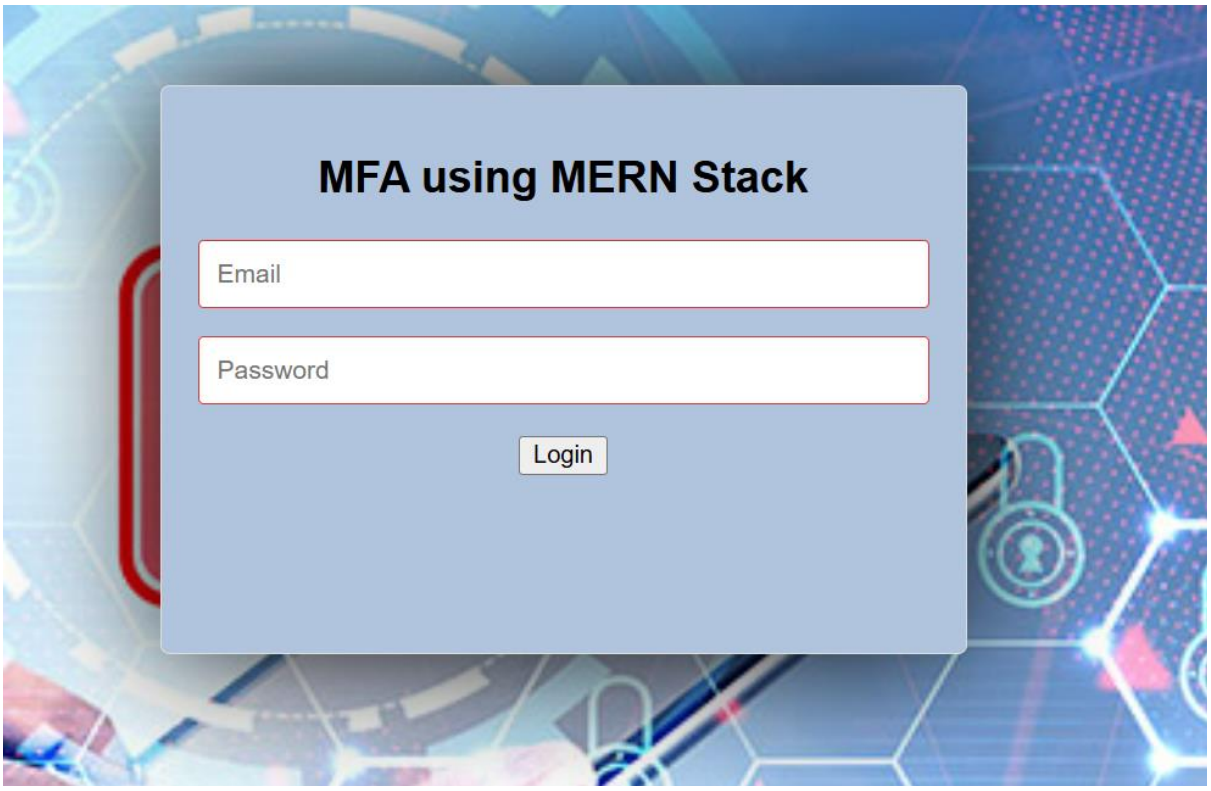
```
// client/src/components/Login.js
import React, { useState } from 'react';
import axios from 'axios';
import './Login.css';

const Login = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [otp, setOtp] = useState('');
  const [showOtpField, setShowOtpField] = useState(false);

  const handleLogin = async () => {
    try {
      const response = await axios.post('http://localhost:3003/auth/login', {
        email,
        password,
      });

      if (response.data.success) {
        setShowOtpField(true);
        alert('OTP sent to your email. Check your inbox.');
      } else {
        alert(response.data.message);
      }
    } catch (error) {
      console.error('Error during login:', error.message);
      alert('An error occurred during login');
    }
};

const handleOtpVerification = async () => {
    try {
        const otpResponse = await axios.post('http://localhost:3003/auth/verify-otp', {
            otp,
        });

        if (otpResponse.data.success) {
            alert('OTP Verified. User logged in.');
            // Redirect to your dashboard or perform any additional actions for successful login
        } else {
            alert('Invalid OTP. Please try again.');
        }
    } catch (error) {
        console.error('Error during OTP verification:', error.message);
        alert('An error occurred during OTP verification');
    }
};

return (
    <div className="login-container">
        <input
            type="email"
            placeholder="Email"
            onChange={(e) => setEmail(e.target.value)}
        />
        <input
```

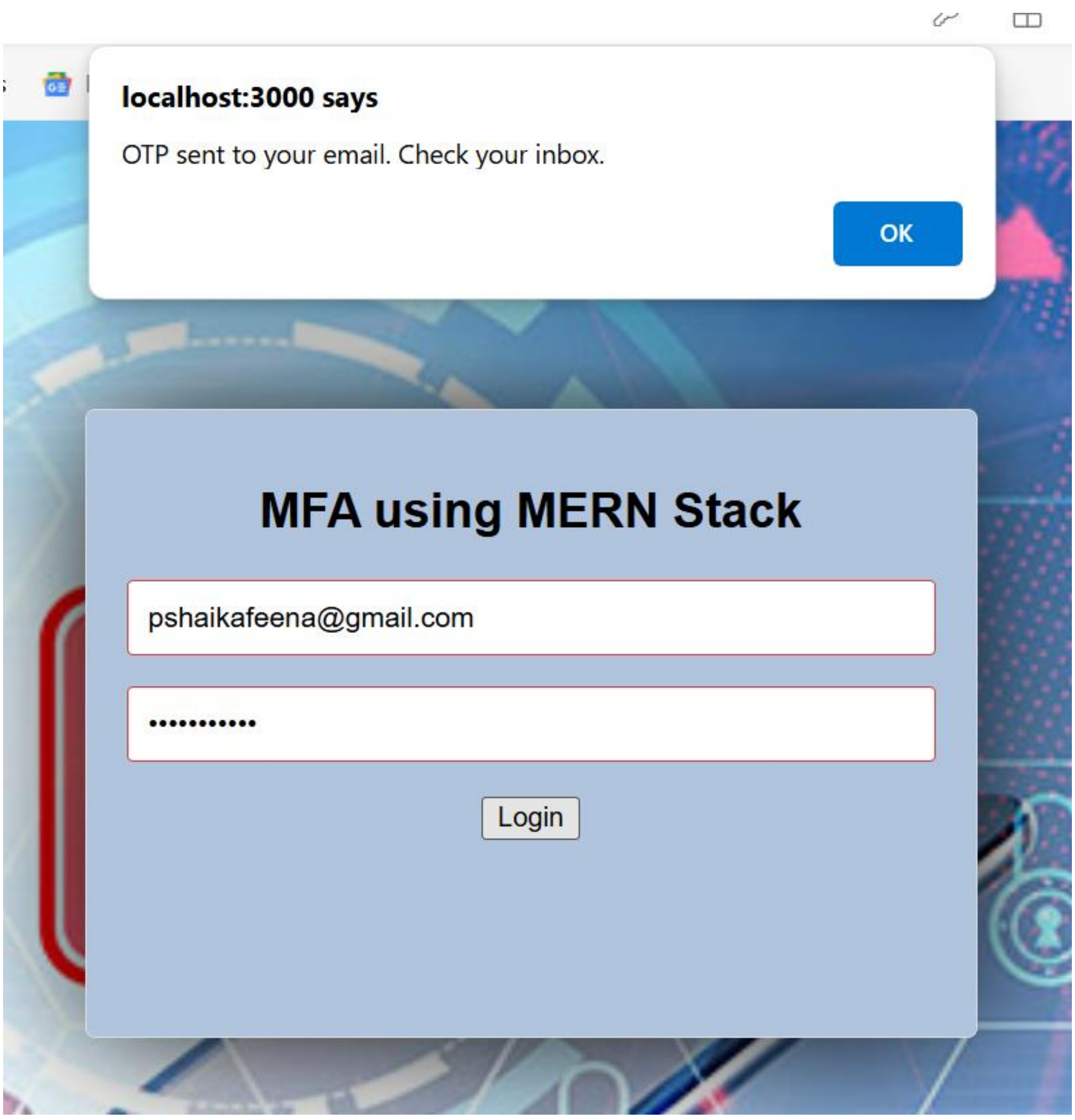
```
            type="password"
            placeholder="Password"
            onChange={(e) => setPassword(e.target.value)}
          />

          {showOtpField && (
            <>
              <input
                type="text"
                placeholder="OTP"
                onChange={(e) => setOtp(e.target.value)}
              />
              <button className="login-button" onClick={handleOtpVerification}>
                Verify OTP
              </button>
            </>
          )}

          <button className="login-button" onClick={handleLogin}>
            Login
          </button>
        </div>
    );
};
export default Login;
```

Step 6: To start the frontend run the following command.

```
npm start
```

Output: It include screenshots of the user login page, the OTP entry interface, and the OTP sent to email. Additionally, provide visuals of the backend console logs for OTP generation and verification, and a confirmation message after successful authentication.


MFA using MERN Stack
Email
Password
Login



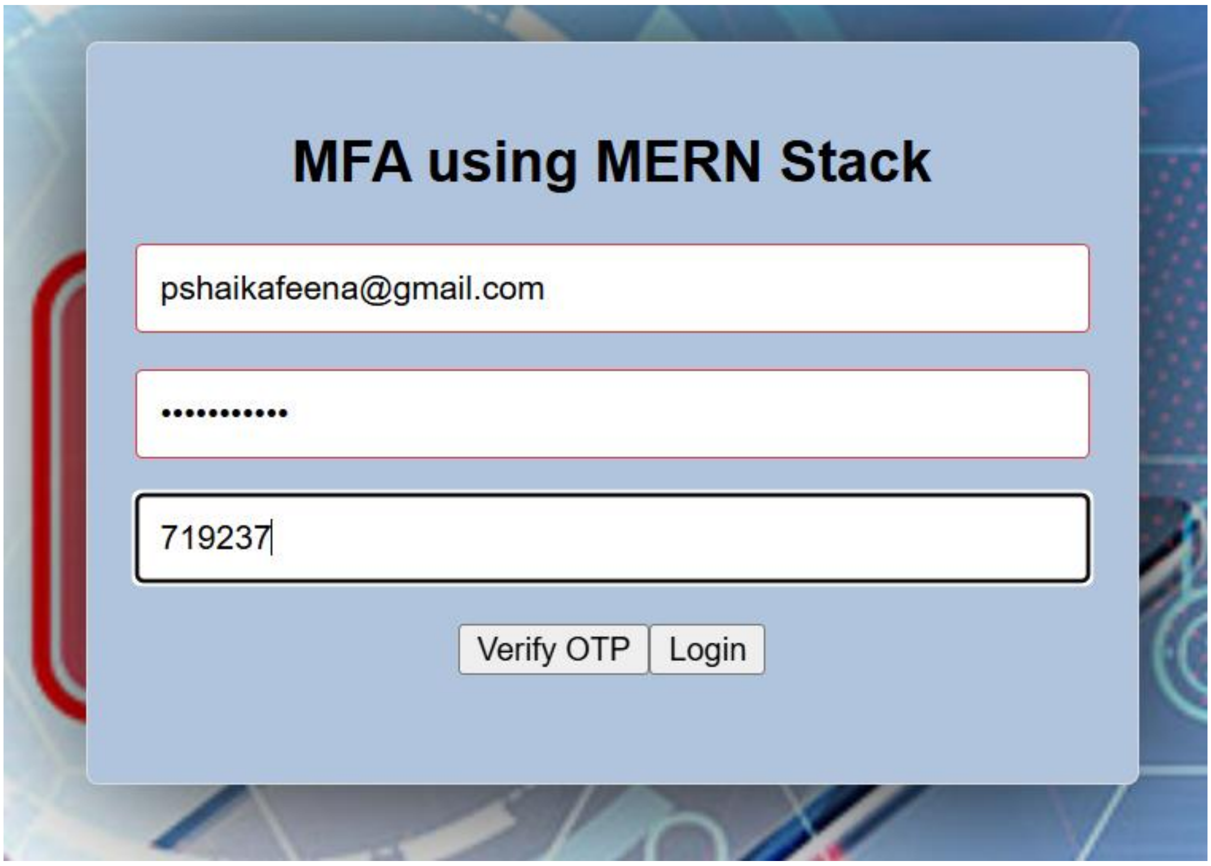
localhost:3000 says

OTP sent to your email. Check your inbox.

MFA using MERN Stack

pshaikafeena@gmail.com

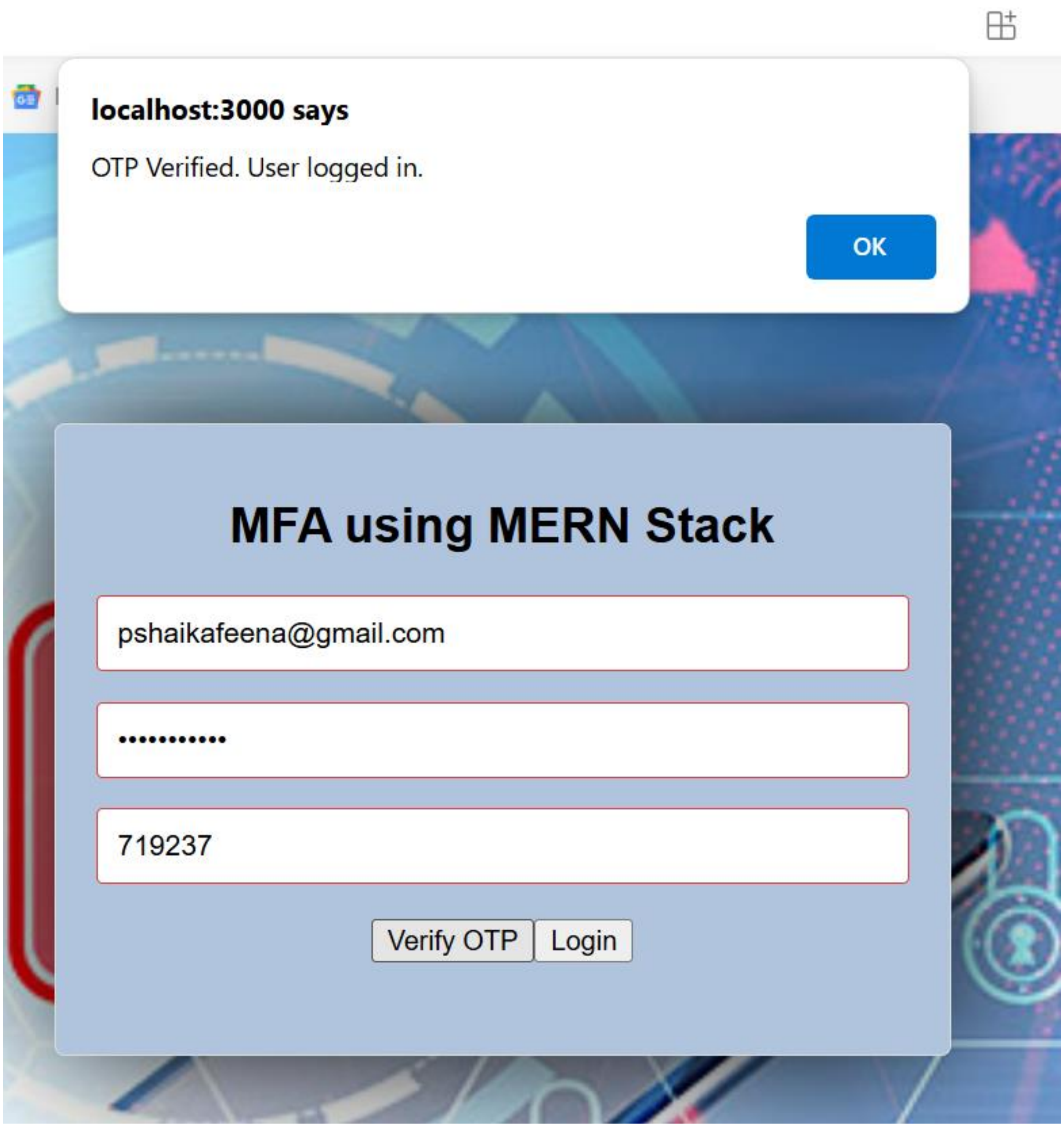
•••••••••••


MFA using MERN Stack

pshaikafeena@gmail.com

•••••••••••

719237



localhost:3000 says

OTP Verified. User logged in.

MFA using MERN Stack

pshaikafeena@gmail.com

•••••••••••

719237