Implementing Google Login in a Node.js Application

Tasadduq Ali · Follow

3 min read · Jul 12, 2023

@codeculturepro
1
How to implement Login with google in Node js
IF YOU WANT EDITABLE CODE GO CHECK US ON MEDIUM
LINK IN BIO.
Tassu
@codeculturepro

To implement Google Login, you need to set up a Google API project. Follow these steps:

1. Go to the Google Developers Console (**https://console.developers.google.com/**).
2. Create a new project or select an existing one.
3. Enable the "Google+ API" for your project.

4. Navigate to the "Credentials" section and click on "Create Credentials."
5. Choose "OAuth client ID" as the credential type.
6. Select "Web application" as the application type.
7. Enter a name for your OAuth client ID.
8. Add authorized JavaScript origins (e.g., **http://localhost:3000**) and redirect URIs (e.g., **http://localhost:3000/auth/google/callback**).
9. Click "Create" to generate the client ID and client secret.

Step 2: Set Up the Node.js Project

Start by setting up a new Node.js project using a package manager like npm or yarn. Run the following command in your terminal:

```
csharpCopy code
npm init
```

This will create a `package.json` file to manage your project's dependencies.

Step 3: Install Required Packages

To implement Google Login in Node.js, we need to install the necessary packages. Run the following command in your terminal:

```
Copy code
```

```
npm install express axios
```

This will install Express.js, a popular Node.js web framework, and the **`axios`** package, which we will use to make HTTP requests.

Step 4: Implement the Authentication Routes

Create a new file, **`authRoutes.js`**, to define the authentication routes for your application. In this file, implement the routes for initiating the Google Login flow, handling the callback URL, and logging out the user.

```
javascriptCopy code
const express = require('express');
const axios = require('axios');
const router = express.Router();

const CLIENT_ID = 'YOUR_CLIENT_ID';
const CLIENT_SECRET = 'YOUR_CLIENT_SECRET';
const REDIRECT_URI = '<http://localhost:3000/auth/google/callback>';

// Initiates the Google Login flow
router.get('/auth/google', (req, res) => {
  const url = `https://accounts.google.com/o/oauth2/v2/auth?client_id=${CLIENT_ID}&r
  res.redirect(url);
});

// Callback URL for handling the Google Login response
router.get('/auth/google/callback', async (req, res) => {
  const { code } = req.query;

  try {
    // Exchange authorization code for access token
    const { data } = await axios.post('<https://oauth2.googleapis.com/token>', {
      client_id: CLIENT_ID,
      client_secret: CLIENT_SECRET,
      code,
      redirect_uri: REDIRECT_URI,
      grant_type: 'authorization_code',
    });
    const { access_token, id_token } = data;

    // Use access_token or id_token to fetch user profile
    const { data: profile } = await axios.get('<https://www.googleapis.com/oauth2/v1
      headers: { Authorization: `Bearer ${access_token}` },
    });

    // Code to handle user authentication and retrieval using the profile data

    res.redirect('/');
  } catch (error) {
    console.error('Error:', error.response.data.error);
    res.redirect('/login');
  }
});

// Logout route
router.get('/logout', (req, res) => {
  // Code to handle user logout
  res.redirect('/login');
});

module.exports = router;
```

Make sure to replace **`'YOUR_CLIENT_ID'`** and **`'YOUR_CLIENT_SECRET'`** with your own credentials obtained from the Google API project.

Step 5: Set Up the Express.js Server

In your main server file (e.g., **`app.js`**), import the required packages and configure the Express.js server. Include the authentication routes defined in **`authRoutes.js`**.

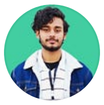
```
javascriptCopy code
const express = require('express');
const authRoutes = require('./authRoutes');
const app = express();

app.use('/', authRoutes);

// Start the server
app.listen(3000, () => {
  console.log('Server started on port 3000');
});
```

if you have any questions or suggestions just do let me know on my Instagram or at codeculturepro@gmail.com

Oauth Google Authenticator Node React Express



Written by Tasadduq Ali

Follow

402 Followers · 1 Following

I am MERN Stack developer working in UAE Govt to digitize their massive services. I will help you to become highly skilled Coder 😉

Responses (3)

What are your thoughts?

Respond

Richard Shaju
5 months ago

Nice and to the point.

Reply

HOLLIX
over 1 year ago

Thanks for sharing this great tutorial!

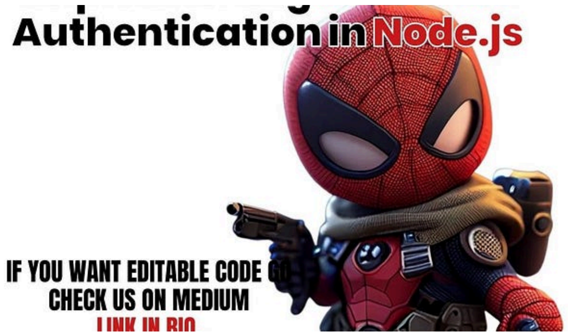
Reply

Younis Jad
over 1 year ago

nice basic auth using google

Reply

More from Tasadduq Ali



Tasadduq Ali

Implementing Authentication in NodeJS App using OAuth2.0

Implementing OAuth 2.0 Authentication in Node.js

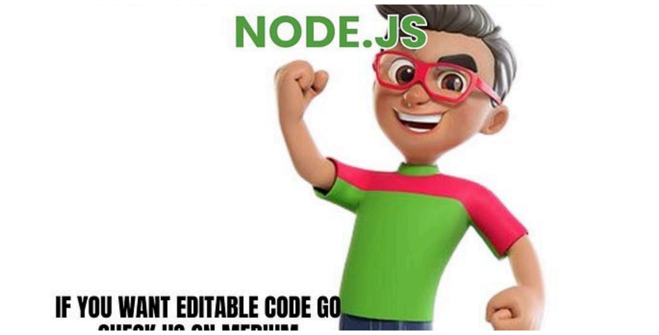
Jul 9, 2023 79 2

Tasadduq Ali

How to implement login with Facebook in Node.Js

In this article, we have discussed the whole implementation guide of implementing the F...

Jul 17, 2023 7 2



Tasadduq Ali

How to implement Login with Apple in Node JS

Step 1: Set Up an Apple Developer Account

Jul 26, 2023 136 2

Tasadduq Ali

How to Become an AI Engineer using JavaScript: Roadmap

Before starting. If you want to join the Full Stack Development Mentorship Bootcamp...

Jun 5, 2023 3

See all from Tasadduq Ali

Recommended from Medium

Ægir Máni Hauksson

Next-Auth with a custom authentication backend

Jul 28 60 1

Lovish Kumar

Encrypt and Decrypt JWT Token using RSA Algorithm in Node.js

In modern applications, JWT (JSON Web Tokens) are widely used for authentication...

Sep 5 13

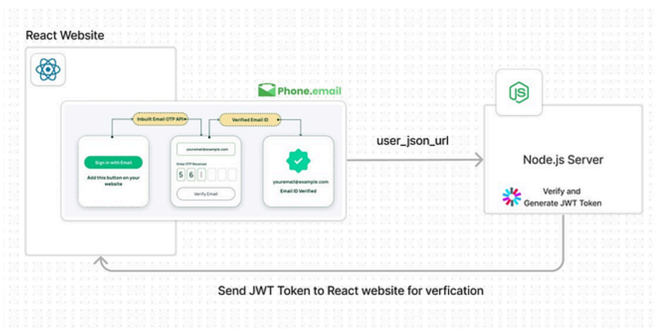
Lists

Stories to Help You Grow as a Software Developer
19 stories · 1506 saves

General Coding Knowledge
20 stories · 1790 saves

Medium's Huge List of Publications Accepting...
377 stories · 4056 saves

Natural Language Processing
1842 stories · 1466 saves


React Website
Phone.email
user_json_url
Node.js Server
Verify and Generate JWT Token
Send JWT Token to React website for verfication

PhoneEmail @ https://phone.email

How to Implement OTP Verification in Reactjs & Nodejs Under 2...

In less than two minutes, we will learn how to incorporate OTP verification into a React.js...

Jun 10 143 1

In Stackademic by Deepak Chaudhari

Node.js(Express): req.body vs req.params vs req.query

Node.js (Express): The Differences Between req.body, req.params, and req.query"

Jun 19 533 9

Always Free
24 GB RAM + 4 CPU + 200 GB
FREE
@harendraverma2 @harendra21 @harendra21

Harendra

How I Am Using a Lifetime 100% Free Server

Get a server with 24 GB RAM + 4 CPU + 200 GB Storage + Always Free

Oct 26 6.4K 93

In JavaScript in Plain Engli... by ✨ Upeksha Hera...

Deploy Node.js backend for free on Vercel

Deploying any frontends for free is not a big deal. You have plenty of options like Vercel,...

Oct 29 199 6

See more recommendations