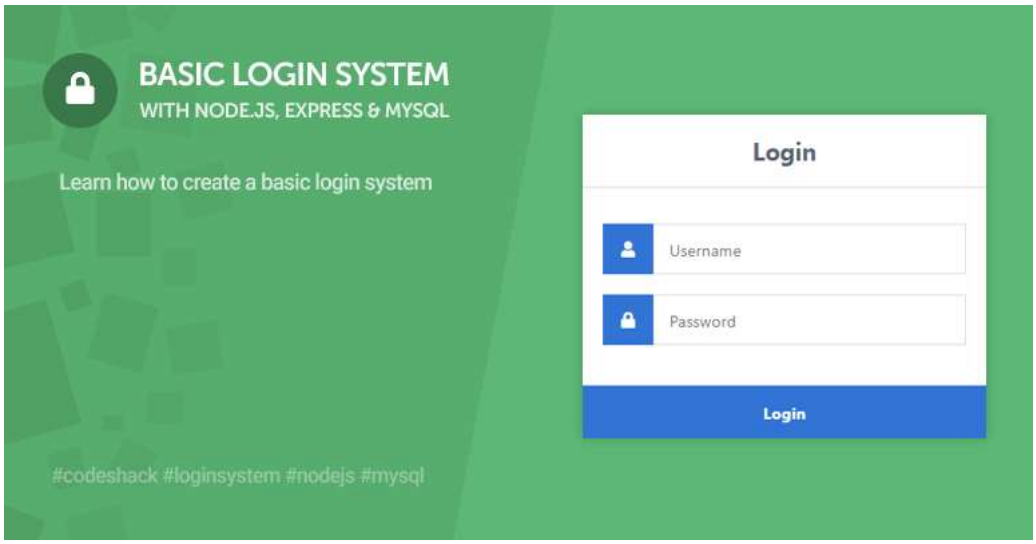
Basic Login System with Node.js, Express, and MySQL

Updated on **February 4, 2022** by **David Adams**



In this tutorial, we'll be developing a login system with Node.js, Express, and MySQL. Node.js will enable us to develop our application with JavaScript, Express is a web framework, and we'll be using MySQL to store and retrieve account details (username, password, etc.).

The login system we'll be developing is similar to that of the PHP login system tutorial, but will not be as comprehensive.

The Advanced package includes additional features and a download link to the source code. For more detailed information and screenshots, click here.

Contents

1. Why create a login system with Node.js as opposed to PHP?

Node.js is a powerful open-source server environment that leverages JavaScript as its core scripting language.

As more people become aware of Node.js, it is becoming increasingly popular in the development of web applications. Therefore, if you plan to develop applications for the future web, I highly suggest you enhance your knowledge with Node.js.

Did you know?

Unlike PHP, Node.js does not depend on Apache or Nginx because Node.js is its own environment.

If you are familiar with JavaScript, you will enjoy developing applications with Node.js, and will be able to adapt very easily.

2. Getting Started

Before we start developing our Node.js login system, we need to install software and packages that our app will depend on, and subsequently create the structure for our app.

2.1. What You Will Learn in this Tutorial

- Establishing a connection to a MySQL database and selecting rows using MySQL queries.
- Implementing GET and POST requests with Node.js and Express.
- Designing a login form with CSS3.
- Implementing validation that will ensure the captured data is valid.
- Leveraging sessions, which will determine whether a user is logged in or not.

2.2. Requirements

- MySQL Server >= 5.6
- Node.js
- Express - Install with command: `npm install express --save`.
- Express Sessions - Install with command: `npm install express-session --save`.
- MySQL for Node.js - Install with command: `npm install mysql --save`.

2.3. Setup & File Structure

Follow the below instructions.

- Create a new directory called `nodelogin`, which can be created anywhere on your environment.
- Open command line as administrator, and navigate to your new directory with the following command: `cd c:\nodeprojects\nodelogin`
- Run the command: `npm init` - it will prompt us to enter a package name, enter: `login`.
- When it prompts to enter the entry point, enter `login.js`.
- We need to install the packages listed in the requirements, so we must execute the commands listed in the requirements above.

A new directory will appear called `node_modules`, which is populated with all the modules we've installed. Don't delete this directory, otherwise it will break our app.

File Structure

```
\-- nodelogin
    |-- login.html
    |-- login.js
```

3. Styling the Login Form with CSS

Cascading style sheets will enable us to structure the login form and make it look more appealing. The stylesheet file consists of properties that are associated with HTML elements.

Edit the ***style.css*** file and add:

```css
* {
    box-sizing: border-box;
    font-family: -apple-system, BlinkMacSystemFont, "segoe ui", roboto, oxygen, ubuntu, cantarell,
    font-size: 16px;
}
body {
    background-color: #435165;
}
.login {
    width: 400px;
    background-color: #ffffff;
    box-shadow: 0 0 9px 0 rgba(0, 0, 0, 0.3);
    margin: 100px auto;
}
.login h1 {
    text-align: center;
    color: #5b6574;
    font-size: 24px;
    padding: 20px 0 20px 0;
    border-bottom: 1px solid #dee0e4;
}
.login form {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    padding-top: 20px;
}
.login form label {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 50px;
    height: 50px;
    background-color: #3274d6;
    color: #ffffff;
}
.login form input[type="password"], .login form input[type="text"] {
    width: 310px;
    height: 50px;
    border: 1px solid #dee0e4;
    margin-bottom: 20px;
    padding: 0 15px;
}
.login form input[type="submit"] {
    width: 100%;
    padding: 15px;
   margin-top: 20px;
    background-color: #3274d6;
    border: 0;
    cursor: pointer;
    font-weight: bold;
    color: #ffffff;
    transition: background-color 0.2s;
}
.login form input[type="submit"]:hover {
```

That's all we need to add to our CSS file.

4. Creating the Login Template with HTML

The login form will consist of an HTML form element and input elements, enabling the user to enter their details and submit them. There is no need to include Node.js code in the template file.

Edit the ***login.html*** file and add:

HTML Copy

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,minimum-scale=1">
        <title>Login</title>
        <!-- the form awesome library is used to add icons to our form -->
        <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
        <!-- include the stylesheet file -->
        <link href="/style.css" rel="stylesheet" type="text/css">
    </head>
    <body>
        <div class="login">
            <h1>Login</h1>
            <form action="/auth" method="post">
                <label for="username">
                    <!-- font awesome icon -->
                    <i class="fas fa-user"></i>
                </label>
                <input type="text" name="username" placeholder="Username" id="username" required>
                <label for="password">
                    <i class="fas fa-lock"></i>
                </label>
                <input type="password" name="password" placeholder="Password" id="password" requir
                <input type="submit" value="Login">
            </form>
        </div>
    </body>
</html>
```

Let's narrow down what each element will do.

- Form — we need to use a form element to submit data to our Node.js app. The action attribute will point to our `auth` route (POST request), which we will create later on.
    - Input (username) — will capture the user's username. In addition, the required attribute is declared to ensure the field is mandatory.
    - Input (password) — will capture the user's password.
    - Input (submit) — button that will be used to submit the form.

The login template will enable users to submit their details, and we'll use Node.js to validate the details. We'll be using a POST request to capture the details, which we can then handle in our Node.js `auth` route.

5. Creating the Login App with Node.js

Now that we have all our basics finished, we can finally start developing our app with Node.js.

JS Copy

```
const mysql = require('mysql');
const express = require('express');
const session = require('express-session');
const path = require('path');
```

The above code will include the MySQL, Express, Express-session, and Path modules, and associate them with the variables we have declared.

Before we implement the database connection code, we need a database to connect to. Therefore, we must execute the below SQL statement either with command line or your preferred MySQL Editor. Make sure the MySQL server is running on port 3306.

SQL Copy

```
CREATE DATABASE IF NOT EXISTS `nodelogin` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
USE `nodelogin`;

CREATE TABLE IF NOT EXISTS `accounts` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(50) NOT NULL,
  `password` varchar(255) NOT NULL,
  `email` varchar(100) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;

INSERT INTO `accounts` (`id`, `username`, `password`, `email`) VALUES (1, 'test', 'test', 'test@te
```

The above SQL statement will create the database (`nodelogin`) and create the `accounts` table. In addition, it will insert a test account that we can use for testing purposes.

We can now connect to our database with the following code:

JS Copy

```
const connection = mysql.createConnection({
    host     : 'localhost',
    user     : 'root',
    password : '',
    database : 'nodelogin'
});
```

The connection details must reflect your database credentials. In most local environments, the default username is `root` , so you might not have to change anything, but in production mode, we highly suggest you change the default username for MySQL and set a strong password.

Express is what we'll use for our web application, which includes packages that are essential for server-side web development, such as sessions and handling HTTP requests.

Add the following code to initialize express:

JS Copy

```
const app = express();
```

After, we need to associate the modules we'll be using with Express:

JS Copy

```
app.use(session({
    secret: 'secret',
```

```
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, 'static')));
```

Make sure to update the secret code variable when declaring the session function as it will used to secure the session data. We'll be using sessions to determine whether the user is logged-in or not. The `json` and `urlencoded` methods will extract the form data from our `login.html` file.

Security Tip

When deploying your project to a production server, ensure you're leveraging SSL as it will help secure the browser cookies that are associated with sessions.

After, we need to declare the login route that will output our `login.html` file to the client using a GET request.

Add the following:

JS Copy

```
// http://localhost:3000/
app.get('/', function(request, response) {
    // Render login template
    response.sendFile(path.join(__dirname + '/login.html'));
});
```

When the client establishes a new connection to our Node.js server, it will output the `login.html` file.

Next, we need to add a new route that will authenticate the user.

Add the following:

JS Copy

```
// http://localhost:3000/auth
app.post('/auth', function(request, response) {
    // Capture the input fields
    let username = request.body.username;
    let password = request.body.password;
    // Ensure the input fields exists and are not empty
    if (username && password) {
        // Execute SQL query that'll select the account from the database based on the specified u
        connection.query('SELECT * FROM accounts WHERE username = ? AND password = ?', [username,
            // If there is an issue with the query, output the error
            if (error) throw error;
            // If the account exists
            if (results.length > 0) {
                // Authenticate the user
                request.session.loggedin = true;
                request.session.username = username;
                // Redirect to home page
                response.redirect('/home');
            } else {
                response.send('Incorrect Username and/or Password!');
            }
            response.end();
        });
    } else {
        response.send('Please enter Username and Password!');
        response.end();
    }
});
```

Remember the action we declared for our form in the login template? We are using the same value for the path in our new route, so when the user submits the form, the `/auth` will be appended to the URL.

When the user submits the form, the code will check if both input fields are not empty and will subsequently select the account from our accounts table in our MySQL database. The user is successfully authenticated and redirected to the home page if the account exists. If not, they will encounter an error message.

The `loggedin` session variable will be used to determine whether the user is logged in or not, and the `username` variable we can use to output on the home page.

We can finally create the home route that will output the user's username.

Add the following:

JS Copy

```
// http://localhost:3000/home
app.get('/home', function(request, response) {
    // If the user is loggedin
    if (request.session.loggedin) {
        // Output username
        response.send('Welcome back, ' + request.session.username + '!');
    } else {
        // Not logged in
        response.send('Please login to view this page!');
    }
    response.end();
});
```

Finally, our Node.js server needs to listen on a port, so for testing purposes, we can use port 3000.

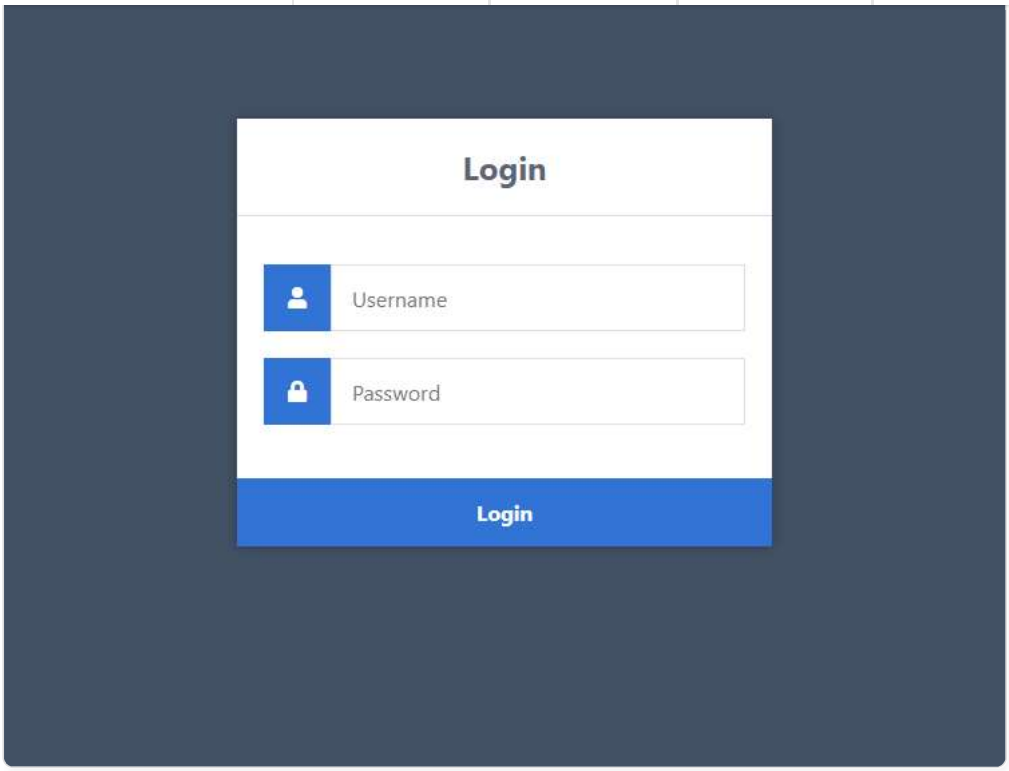
Add the following:

JS Copy

```
app.listen(3000);
```

Ideally, when you deploy your login system to a production server, you would want your server to listen on port 80, so you don't have to specify the port number in the URL.

To start our Node.js app, we can execute the command `node login.js` in command line. If we navigate to `http://localhost:3000/` in our browser, we should see the login form, which should look like the following:



You can proceed to log in with the test account (username: test, password: test). If successful, you will see the username displayed on the screen.

Full login.js Source

JS Copy

```
const mysql = require('mysql');
const express = require('express');
const session = require('express-session');
const path = require('path');

const connection = mysql.createConnection({
    host     : 'localhost',
    user     : 'root',
    password : '',
    database : 'nodelogin'
});

const app = express();

app.use(session({
    secret: 'secret',
    resave: true,
    saveUninitialized: true
}));
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, 'static')));

// http://localhost:3000/
app.get('/', function(request, response) {
    // Render login template
    response.sendFile(path.join(__dirname + '/login.html'));
});
```

```
    let username = request.body.username;
    let password = request.body.password;
    // Ensure the input fields exists and are not empty
    if (username && password) {
        // Execute SQL query that'll select the account from the database based on the specified u
        connection.query('SELECT * FROM accounts WHERE username = ? AND password = ?', [username,
            // If there is an issue with the query, output the error
            if (error) throw error;
            // If the account exists
            if (results.length > 0) {
                // Authenticate the user
                request.session.loggedin = true;
                request.session.username = username;
                // Redirect to home page
                response.redirect('/home');
            } else {
                response.send('Incorrect Username and/or Password!');
            }
            response.end();
        });
    } else {
        response.send('Please enter Username and Password!');
        response.end();
    }
});

// http://localhost:3000/home
app.get('/home', function(request, response) {
    // If the user is loggedin
    if (request.session.loggedin) {
        // Output username
        response.send('Welcome back, ' + request.session.username + '!');
    } else {
        // Not logged in
        response.send('Please login to view this page!');
    }
    response.end();
});

app.listen(3000);
```

Conclusion

You should now have a basic understanding of how a login system works and the fundamentals involved. While we don't recommend deploying the system to a production environment, you can, however, use it as a base in your development projects.

What next? Consider implementing security techniques (password hashing, prepared statements, CSRF, etc.) and the registration aspect.

If you've enjoyed this tutorial, don't forget to like, follow, and share it on social media. We appreciate your support.

Enjoy coding!

If you would like to support us, consider purchasing a package below as it will greatly help us create more quality tutorials and keep our server up and running.

Database SQL file	✔	✔
Login & Registration system	✔	✔
Home page	✔	✔
Profile page	✔	✔
Activate account feature	✔	✔
Edit profile page		✔
Remember me feature		✔
Forgot & Reset Password		✔
AJAX integration		✔
Password encryption		✔
reCAPTCHA protection		✔
CSRF protection		✔
Brute-force protection		✔
Two-factor Authentication		✔
Admin Panel — View Dashboard — Create, edit, and delete accounts — Search, sort, and filter accounts — Manage Email Templates — Edit Settings		✔
Responsive Design (mobile-friendly)		✔
SCSS file	✔	✔
Commented code	✔	✔
Free updates & support (bugs and minor issues)	✔	✔
User Guide	✔	✔
* Payments are processed with PayPal. * Both packages include the tutorial source code. * Advanced package also includes the basic package.		$20.00 PayPal Download Stripe Download Cryptocurrency

For more detailed information regarding the advanced package, click here.

ABOUT AUTHOR

David Adams

Enthusiastic website developer, I've been designing and developing web applications for over 10 years, I enjoy the creativity I put into my projects and enjoy what others bring to the awesome web. My goal is to help newcomers learn the ways of the web.

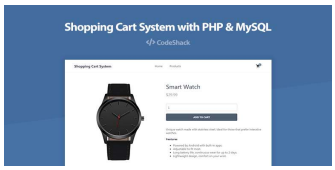
express form login mysql node.js programming sessions tutorials

← 21 Useful PHP Snippets

How to Create a Contact Page with PHP →

RELATED POSTS

Login System with Python Flask and MySQL



Shopping Cart System with PHP and MySQL

Poll and Voting System with PHP and MySQL

104 Comments codeshack Disqus' Privacy Policy Login

Favorite 15 Tweet Share Sort by Newest

Join the discussion…

LOG IN WITH OR SIGN UP WITH DISQUS

Name

Ewout Lagendijk • 2 months ago • edited

bodyparser has since been deprecated, you can replace those two lines with:

```
app.use(express.urlencoded({ extended: true }));
app.use(express.json());
```

Reply • Share ›

David Adams Mod ➚ Ewout Lagendijk • 2 months ago

Thanks for the info! I will look into it.

1 Reply • Share ›

Michal Zachař • 2 months ago

Hi, great tutorial!
Is it possible to use http module instead of express?

Reply • Share ›

David Adams Mod ➚ Michal Zachař • 2 months ago

The problem with using the HTTP module as opposed to Express is that you're missing the essential modules that are required for the login system to function correctly, such as the session

Reply • Share ›

melisa • 3 months ago
Thank you! The tutorial really help me
1 | Reply • Share ›

Boniface Muriga • 4 months ago
thanks! asante sana..
1 | Reply • Share ›

David Ch • 4 months ago • edited
. Thanks!
1 | Reply • Share ›

Booster6969 • 7 months ago • edited
i made a styling in extarnal css. When i open just html with localhost/folder/login.html everything is fine but when i run it thru localhost:3000 it has no styling and no images
| Reply • Share ›

David Adams Mod ➚ Booster6969 • 7 months ago
Try appending the port number to the stylesheet URL.
| Reply • Share ›

DJ Burst Bass • 7 months ago • edited
Hola david ,excelente post ,pienso adentrarme en node . Una pregunta fuera del topic del post.
Cuando el cliente se redirige al home

```
app.get('/home', function(request, response) {
```

Como pudiera darle mas estilo a ese /home? es dentro del mismo login.js o creando otro .js y llamdanolo
| Reply • Share ›

David Adams Mod ➚ DJ Burst Bass • 7 months ago
You can customize the templates with the "style.css" file. If you want to add more content to that file, you can create more elements and such.
1 | Reply • Share ›

ManuTheCoder • 9 months ago
Hi,
I was trying to create the database on repl.it, but I don't know how to...
URL: https://replit.com/@ManuThe...
1 | Reply • Share ›

David Adams Mod ➚ ManuTheCoder • 9 months ago
I'd refer to the docs to see if it supports MySQL. I'm not familiar with the replit website.
| Reply • Share ›

Mike Saunders • a year ago
Hi David thanks for this, tutorial, I want to turn this into an electron desktop app, as this is web based - using localhost how can I change the routes so it will work in a desktop app, thanks
| Reply • Share ›

Avatar This comment was deleted.

David Adams Mod ➚ Guest • a year ago
When you navigate to "http://localhost:3000/" in your browser, do you see the login form? Did you change the action attribute of the HTML form? You could try adding the full URL to this attribute "http://localhost:3000/auth".
1 | Reply • Share ›

Amine Tanou • 2 years ago
thank you so muuuuch <3
How can I have the advanced version ?
| Reply • Share ›

• Reply • Share ›

Jonny • 2 years ago • edited

Hello, need help with install npm Packages "nodemailer"

Full Logs are here: https://forum.netcup.de/ent...

But i have still another Big problem, but dont know "nodemailer" is the reason.
If i login it gave me the error: "Incomplete response received from application"

• Reply • Share ›

David Adams Mod ➚ Jonny • 2 years ago

You need to include the node path when you install nodemailer, your system cannot locate the node binary, this could be a problem with your set-up.

Check this out: https://stackoverflow.com/q...

• Reply • Share ›

Eugene Miller • 2 years ago

Hi, thank you for the tutorial this is very helpful. When I run the code `node login.js` and then connect to localhost:3000 I get the following error.
(This occurs on line 40)

```
if (results.length > 0)
TypeError: Cannot read property 'length' of undefined
```

Any help would be appreciated, thanks.

2 • Reply • Share ›

GhostwayNE ➚ Eugene Miller • 2 years ago

you are getting this error because you are getting undefined value in results

if you do

console.log("query result:", results);

you will get to know the value of results.

Try printing the what is the error with

console.log(error);

This will show you the error:
(Client does not support authentication protocol requested by server; consider upgrading MySQL client)---This was in my case.
then i follow this link to fix this: https://stackoverflow.com/a...

I hope this will help you.

• Reply • Share ›

David Adams Mod ➚ Eugene Miller • 2 years ago

This could be an issue with your MySQL connection or you have not properly set-up the database and tables.

• Reply • Share ›

cidicley cintra • 2 years ago

thanks, I got it. but how to save the name, email, login that comes from the bank in one session to show on all screens until calling the session destroyer. including being able to use the login in an invisible hidden input to register something save in bamco the login of those who made i register

• Reply • Share ›

David Adams Mod ➚ cidicley cintra • 2 years ago

Not sure I understand your question here, if you want to create new session variables you can simply do:

```
request.session.yoursessionname = yoursessionvalue;
```

• Reply • Share ›

cidicley cintra ➚ David Adams • 2 years ago

```
var username = request.body.username;
var password = request.body.password;
var email = [];
if (username && password) {
connection.query('SELECT * FROM accounts WHERE username = ? AND password = ?',
[username, password], function(error, results, fields) {
if (results.length > 0) {
request.session.loggedin = true;

results.forEach(function(row){
//response.send(row.email);
request.session.username = row.username ;
request.session.email = row.email;
request.session.loin = row.login;
```

see more

Reply • Share ›

David Adams Mod ➚ cidicley cintra • 2 years ago

Is this the full source code? Did you add the code from the tutorial?

This could be a problem with your express-session package, you can try and configure it, check here: https://www.npmjs.com/packa...

Reply • Share ›

cidicley cintra ➚ David Adams • 2 years ago

I still haven't got it, do you know any video tutorial that deals with login and password with session in mysql through the node?

Reply • Share ›

David Adams Mod ➚ cidicley cintra • 2 years ago

The express sessions package should be working fine, it is probably a misconfiguration problem with your app.

Reply • Share ›

Tristan • 2 years ago

when I try to login, i get: Cannot POST /auth
any ideas?

Reply • Share ›

David Adams Mod ➚ Tristan • 2 years ago

Did you change the code? Try again with the full source at the end of the post.

Reply • Share ›

Matt • 2 years ago

Before you start using this, please DO NOT use this in your production code. Security is a first priority and without knowledge and proper structure of security with login forms, usernames and passwords are easily exposed and exploited. Use SHA256 or SHA512 and use password salts to encrypt passwords.

Reply • Share ›

David Adams Mod ➚ Matt • 2 years ago

Yes, this is a basic login system to work from, for those that need a more advanced version I recommend they purchase the advanced package, this includes password encryption, and more.

Reply • Share ›

JUAN DAVID CASTAÑO P • 2 years ago

It is running correctly, I leave for here the githubrepository if someone wants to verity, I also leave the database.sql

https://github.com/juanthe7...

Reply • Share ›

Robby Rotten • 2 years ago

Hey, I'm having a problem where I believe mysql is setup correctly & all of that, however the default 'test','test' isn't working, it just says 'Incorrect username & password.'

David Adams Mod ➚ Robby Rotten • 2 years ago • edited

This is probably a MySQL issue, output the error in your queries:

```
if (error) {
    console.log(error);
}
```

This should help you identify any issues.

▲ | ▼ • Reply • Share ›

DEBP ➚ David Adams • a year ago • edited

How would you do if u use sqlite3 to this?

for me it just says Incorrect Username and/or Password!

this is my sql query and i use sqlite3

CREATE TABLE accounts (
username VARCHAR(32),
password VARCHAR(32),
email VARCHAR(32),
id INTEGER UNIQUE PRIMARY KEY AUTOINCREMENT NOT NULL)

INSERT INTO accounts (username, password, email)
VALUES
('test', 'test', 'test@test.com');

▲ | ▼ • Reply • Share ›

Robby Rotten ➚ David Adams • 2 years ago

I'm sorry but I'm not really sure what you're asking me to do with this piece of code. I know it outputs errors, but how shall I implement it?

▲ | ▼ • Reply • Share ›

David Adams Mod ➚ Robby Rotten • 2 years ago

Place the code after this line:

```
connection.query('SELECT * FROM accounts WHERE username = ? AND password = ?', [use
```

This should output any errors to console (if any).

▲ | ▼ • Reply • Share ›

Robby Rotten ➚ David Adams • 2 years ago

Alright there isn't any errors, so it's definitely to do with the mysql connection.

▲ | ▼ • Reply • Share ›

Kenneth Ledeño Pacifico • 2 years ago

Hello is this compatible with SQL Server 2008 R2 ?

▲ | ▼ • Reply • Share ›

David Adams Mod ➚ Kenneth Ledeño Pacifico • 2 years ago

I'm not sure, you'll have to try it.

▲ | ▼ • Reply • Share ›

Kenneth Ledeño Pacifico ➚ David Adams • 2 years ago • edited

Hello sir, I have already tried , but it did not work ,Can you Make a tutorial of SQLServer version of this please?

▲ | ▼ • Reply • Share ›

David Adams Mod ➚ Kenneth Ledeño Pacifico • 2 years ago

I'm not familiar with using SQL server 2008.

▲ | ▼ • Reply • Share ›

Kenneth Ledeño Pacifico ➚ Kenneth Ledeño Pacifico • 2 years ago

Do you have youtube channel or github sir ?

This article is helpful by using with CRUD Operation in Expressjs with MVC Pattern in Express, But Let me know that How to send flash message after inserting, updating, or deleting the data

Reply • Share ›

David Adams Mod ➚ Rapsan jani • 2 years ago

Check out this page: https://stackoverflow.com/q...

Reply • Share ›

Nathan Oliveira • 2 years ago • edited

Thank you, until I can adapt it to my structure that I did (MVC)

1 Reply • Share ›

raj mohan • 2 years ago

hi when i try this with postgresql unsing

connection query

const Pool = require('pg').Pool

const pool = new Pool({

user: 'postgres',

host: 'localhost',

database: 'api',

password: 'wwld',

port: 5432,

})

it shows this error

see more

Reply • Share ›

raj mohan ➚ raj mohan • 2 years ago

how to solve this

Reply • Share ›

Load more comments

Search blog ...

Informative posts straight to your inbox

Join our large group of subscribers

RECENT POSTS

21 JavaScript Code Snippets for Beginners

Hotel Reservation Form with PHP

Event Calendar with PHP

TAGS

tutorials php mysql programming javascript snippets css form php class html login scripting mysqli ajax pdo shopping cart freebies registration event calendar voting system commenting system crud mvc table express jquery hotel reservation form progamming poll system sort sessions python cookie pagination content locker contact gallery system review system ticketing system template columns node.js database flask snippet register