DETAILED EXPLANATION OF EACH STEP IMPLEMENTATION OF CODE

**Authentication in NodeJS With Express and Mongo**

**1. Introduction**

**Authentication** - It is a process of identifying user identity.

User Authentication contains various steps, please check out this flowchart to know more. We will be using this flow to build the authentication system in our application.

**2. Prerequisites**

You should have prior knowledge of javascript basics, nodejs. Knowledge of ES6 syntax is a plus. And, at last **nodejs** should be installed on your system.

**3. Packages Required**

You will be needing these following 'npm' packages.

1. **express**  
   Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications
2. **express-validator**  
   To Validate the body data on the server in the express framework, we will be using this library. It's a server-side data validation library. So, even if a malicious user bypasses the client-side verification, the server-side data validation will catch it and throw an error.
3. **body-parser**  
   It is nodejs middleware for parsing the body data.
4. **bcryptjs**  
   This library will be used to hash the password and then store it to database.This way even app administrators can't access the account of a user.
5. **jsonwebtoken**  
   **jsonwebtoken** will be used to encrypt our data payload on registration and return a token. We can use that **token** to authenticate ourselves to secured pages like the dashboard. There would also an option to set the validity of those token, so you can specify how much time that token will last.
6. **mongoose**  
   Mongoose is a MongoDB object modeling tool designed to work in an asynchronous environment. Mongoose supports both promises and callbacks.

**4. Initiate Project**

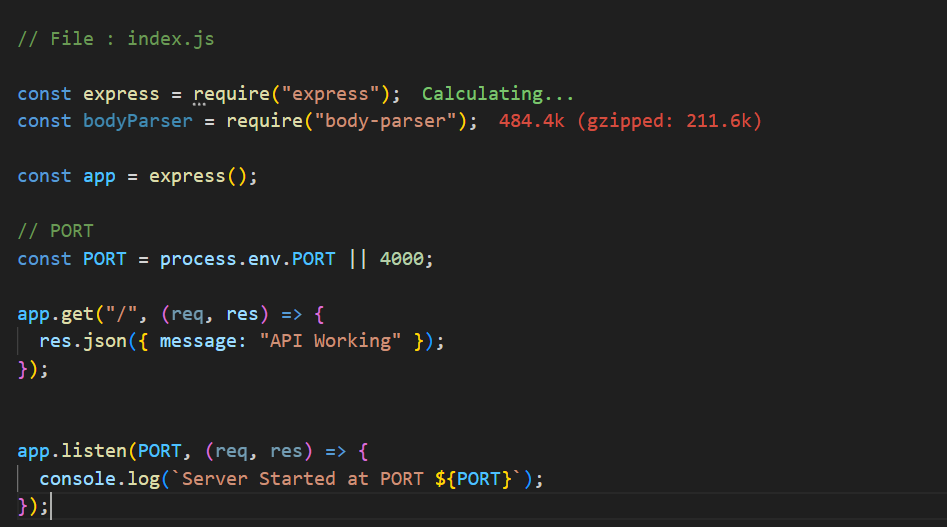
We will start by creating a node project. So, Create a new folder with the name 'node-auth' and follow the steps below. All the project files should be inside the 'user-auth' folder.

****

*npm init* will ask you some basic information about project. Now, you have created the node project, it's time to install the required packages. So, go ahead and install the packages by running the below command.



Now, create a file ***index.js*** and add this code.



If you type node index.js in the terminal, the server will start at PORT 4000.

You have successfully set up your NodeJS app application. It's time to set up the database to add more functionality.

**5. Setup MongoDB Database**

We will be using MongoDB Database to store our users. You can use either a cloud MongoDB server or a local MongoDB server.

In this CodeLab, we will be using a Cloud MongoDB server

So, First, go ahead and signup on mLab. And follow the below steps.

1. After successful signup, Click on Create Button on home page.
2. give name
3. click on Users and add a user by clicking on Add Database User.and Now, choose any cloud provider for example AWS. In the Plan Type choose the free
4. Select the region(any) and click continue. its free of cost.
5. Enter a DB name(any). I am using user-auth. Click continue and then submit the order on the next page.
6. Now, You will be re-directed to the homepage. Select your DB i.e user-auth.
7. Copy the standard MongoDB URI.
8. Now, you need to add a user to your database. From the 5 tabs below,

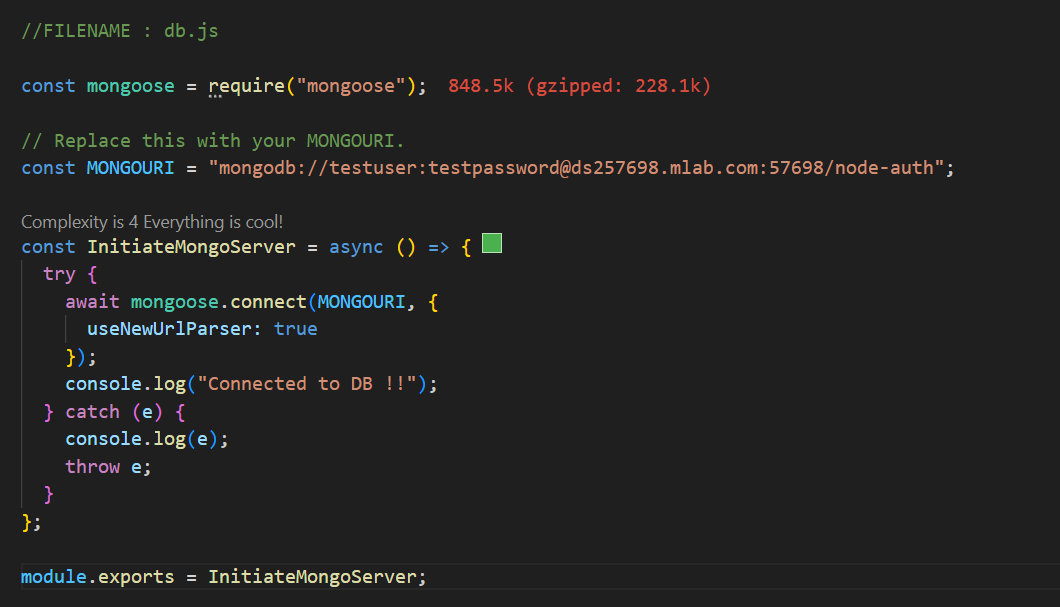
Now, you have got your database user. Replace the && with your DB username and password.

**mongodb+srv://username:<password>@cluster0.qnudkim.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0**

**6. Configure User Model**

Let's go and first create a config folder. This folder will keep the database connection information.

Create a file named**: db.js** in **config**

****

Now, we are done the database connection. Let's create the User Model to save our registered users.

Go ahead and create a new folder named **model**. Inside the model folder, create a new file **User.js**.

We will be using **mongoose** to create UserSchema.

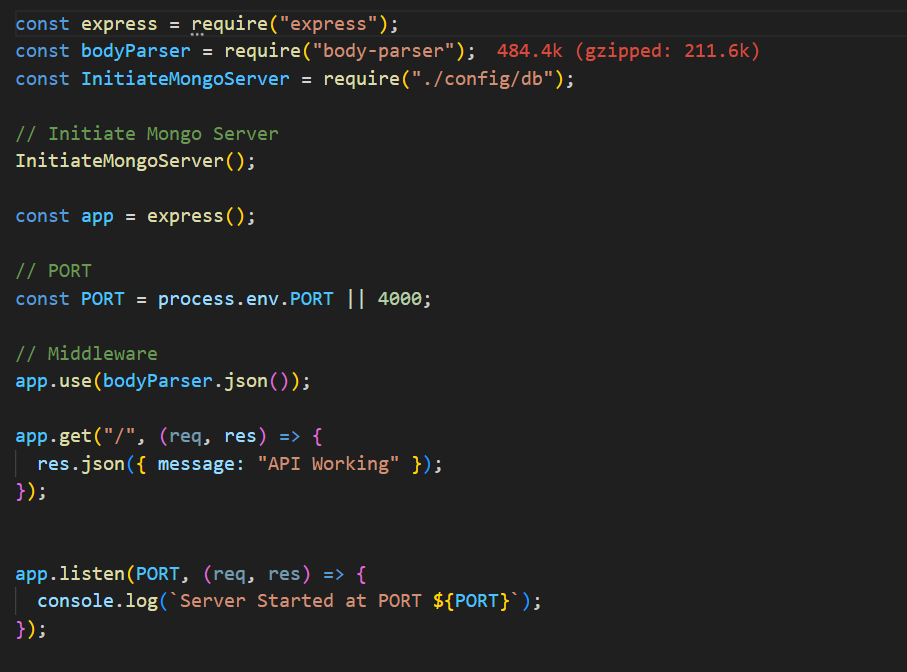
**User.js**

A screen shot of a computer

Description automatically generated

Now, we are done with **Database Connection**, **User Schema**. So, let's go ahead and update our index.js to connect our API to the database.

**index.js**



We can successfully connected your app to the MongoDB server.

Now, the next thing we have to do is make a /user/signup route to register a new user. We will see this in the next section.

**7. User Signup**

The Route for user registration will be **'/user/signup'**.

Create a folder named routes. In the 'routes' folder, create a file named user.js

**routes/user.js**

A screen shot of a computer program

Description automatically generated

Conti..

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

Now, we have created the user registration in **'routes/user.js'**. So, we need to import this in **index.js** to make it work.

So, the updated **index** file code should look like this.  
**index.js**

A screen shot of a computer program

Description automatically generated

Let's start the user registration using postman. A postman is a tool for API testing.  
you can find postman API testing tool **extension in vs code** or u can go to website and registor and find the tokens

A screenshot of a computer

Description automatically generated

**8. User Login**

Now, it's time to implement the Login router which will be mounted on '/user/login'.

Here is the code snippet for login functionality. Add the below code snippet in **user.js**



A screen shot of a computer program

Description automatically generated

**9. Get Logged In User**

Now, your [User Signup](https://dev.to/dipakkr/implementing-authentication-in-nodejs-with-express-and-jwt-codelab-1-j5i#7-user-signup) and [User Login](https://dev.to/dipakkr/implementing-authentication-in-nodejs-with-express-and-jwt-codelab-1-j5i#7-user-login) is working, and you are getting a token in return.

So, our next task will be to Retrieve the LoggedIn user using the **token**. Let's go and add this functionality.

The /user/me route will return your user if you pass the token in the header. In the file **route.js**, add the below code snippet.

A screenshot of a computer program

Description automatically generated

As you can see, we added the auth middleware as a parameter in the **/user/me** GET route, so let's define **auth** function.

Go ahead and create a new folder named **middleware**. Inside this folder, create a file named **auth.js**

This auth middleware will be used to verify the token, retrieve user based on the token payload.

**middleware/auth.js**

A computer screen shot of text

Description automatically generated

You have successfully created an authentication API in nodejs. Now, You can go ahead and test the **/user/me** endpoint after logging in.