To set up a simple project with JWT authentication for login and password change, using Node.js, Express, MySQL, and React, here is a step-by-step guide along with the code. This guide will provide a single-file backend and single terminal commands to run everything.

**Step 1: Set Up the Backend**

* 1. **Create the backend directory and initialize the project:**

mkdir jwt-auth-app

cd jwt-auth-app

mkdir server

cd server

npm init -y

**1.2. Install necessary backend dependencies:**

bash

Copy code

**1.3. Create a single backend file (server.js):**

Create the server.js file with the following code:

// server.js

const express = require('express');

const jwt = require('jsonwebtoken');

const mysql = require('mysql2');

const dotenv = require('dotenv');

// Initialize dotenv to read environment variables

dotenv.config();

// Create a MySQL connection pool

const pool = mysql.createPool({

  host: process.env.DB\_HOST,

  user: process.env.DB\_USER,

  password: process.env.DB\_PASSWORD,

  database: process.env.DB\_NAME,

  waitForConnections: true,

  connectionLimit: 10,

  queueLimit: 0

});

const db = pool.promise();

let refreshTokens = [];

app.post("/token", (req, res) => {

  const refreshToken = req.body.token;

  if (refreshToken == null) return res.sendStatus(401);

  if (!refreshTokens.includes(refreshToken)) return res.sendStatus(403);

  jwt.verify(refreshToken, process.env.REFRESH\_TOKEN\_SECRET, (err, user) => {

    if (err) return res.sendStatus(403);

    const accessToken = generateAccessToken({ name: user.name });

    res.json({ accessToken: accessToken });

  });

});

app.delete("/logout", (req, res) => {

  const tokenToDelete = req.headers["token"];

  refreshTokens = refreshTokens.filter((token) => token !== tokenToDelete);

  res.sendStatus(204);

});

function generateAccessToken({ user }) {

  return jwt.sign({ user }, process.env.ACCESS\_TOKEN\_SECRET, {

    expiresIn: "24h",

  });

}

function authenticateToken(req, res, next) {

  const authHeader = req.headers["authorization"];

  const token = authHeader && authHeader.split(" ")[1];

  if (token == null) return res.sendStatus(401);

  jwt.verify(token, process.env.ACCESS\_TOKEN\_SECRET, (err, user) => {

    // console.log(err);

    if (err) return res.sendStatus(403);

    // console.log(req.user, user);

    req.user = user;

    next();

  });

}

// Initialize Express app

const app = express();

app.use(express.json());  // To parse JSON request bodies

app.post("/user", (req, res) => {

  const sql = "SELECT \* FROM register WHERE `email` = ? AND `password` = ?";

  db.query(sql, [req.body.username, req.body.password], (err, data) => {

    if (err) {

      return res.json("Error");

    }

    if (data.length > 0) {

      const accessToken = generateAccessToken({data})

      // console.log(accessToken)

      const refreshToken = jwt.sign({data}, process.env.REFRESH\_TOKEN\_SECRET)

      refreshTokens.push(refreshToken)

      // const token = jwt.sign({data}, secretKey, { expiresIn: '1h' });

      return res.json({accessToken, data});

      // return res.json(data);

    } else {

      return res.json("Fail");

    }

  });

});

app.post("/admin", (req, res) => {

  const sql = "SELECT \* FROM admin WHERE `email` = ? AND `password` = ?";

  db.query(sql, [req.body.username, req.body.password], (err, data) => {

    if (err) {

      return res.json("Error");

    }

    if (data.length > 0) {

      const accessToken = generateAccessToken({data})

      // console.log(accessToken)

      const refreshToken = jwt.sign({data}, process.env.REFRESH\_TOKEN\_SECRET)

      refreshTokens.push(refreshToken)

      // const token = jwt.sign({data}, secretKey, { expiresIn: '1h' });

      return res.json({accessToken, data});

      // return res.json(data);

    } else {

      return res.json("Fail");

    }

  });

});

app.get("/admin",authenticateToken, (req, res) => {

  const sql ="SELECT \* FROM admin";;

  db.query(sql, (err, data) => {

    if (err) {

      return res.json("Error");

    }

    if (data.length > 0) {

      return res.json(data);

    } else {

      return res.json("Fail");

    }

  });

});

app.get("/user",authenticateToken, (req, res) => {

  const sql =  "SELECT \* FROM register";

  db.query(sql, (err, data) => {

    if (err) {

      return res.json("Error");

    }

    if (data.length > 0) {

      return res.json(data);

    } else {

      return res.json("Fail");

    }

  });

});

// Start the server

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => {

  console.log(`Server running on port ${PORT}`);

});

**Step 2: Create the MySQL Database**

Run the following SQL commands to set up the database and tables:

CREATE DATABASE jwt\_auth\_app;

USE jwt\_auth\_app;

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL

);

CREATE TABLE refresh\_tokens (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

refresh\_token VARCHAR(255) UNIQUE NOT NULL,

FOREIGN KEY (user\_id) REFERENCES users(id)

);

### Step 4: Run the Server

Start the backend server by running:

node server.js

### Step 5: Set Up the Frontend (React)

**5.1. Create a React app:**

cd ..

npx create-react-app client

cd client

**5.2. Install Axios for making HTTP requests:**

npm install axios

**5.3. Create basic React components for login and password change (src/components/Login.js and src/components/ChangePassword.js) with appropriate Axios calls.**

1. **Login**
2. **changpassword**

**Step 6: Test the App**

* Use Postman or a similar tool to test the login endpoint (POST /login).
* Implement and test the React frontend for login and password change functionality.

**Additional Terminal Commands for JWT Token Generation:**

### Sample .env File

ini

# Database connection settings

DB\_HOST=localhost

DB\_USER=root

DB\_PASSWORD=your\_password

DB\_NAME=jwt\_auth\_app

# JWT secret key (used for signing and verifying tokens)

JWT\_SECRET=your\_jwt\_secret

# Port the server should run on

PORT=5000

### Step-by-Step Guide to Generate a JWT Secret Using Node.js

1. **Open Visual Studio Code**.
2. **Open the Terminal**: Use the shortcut Ctrl+` (backtick) or go to View > Terminal.
3. **Run the following command** to generate a secure JWT secret:

**node -e "console.log(require('crypto').randomBytes(32).toString('base64'))"**