**Documentation for Authentication and Password Reset API**

This document provides detailed instructions on how to use and interact with the provided Node.js code for user authentication and password reset. The API is built using Express, MongoDB, bcrypt, jwt, and nodemailer. Follow the steps below to understand the functionality and integration of this code.

Table of Contents

1 Installation

2 Configuration

3 MongoDB Configuration

4 JWT Secret Key

5 Email Service Provider

6 Endpoints

=> 1. Register

=> 2. Login

=> 3. Forget Password (Send Reset Link)

=> 4. Reset Password

=> 5. Get User Info

=> 1 Usage Examples

6 Middleware

=> 1. Authenticate Token

7 Running the Server

8 Conclusion

**Installation**

Clone the repository:

bash

Copy code

git clone <repository-url>

Navigate to the project directory:

bash

Copy code

cd <project-directory>

Install dependencies:

bash

Copy code

npm install

Configuration

MongoDB Configuration

Ensure MongoDB is installed and running on your system.

Open app.js and modify the MongoDB connection string:

javascript

Copy code

mongoose.connect('mongodb://localhost/your\_database\_name', { useNewUrlParser: true, useUnifiedTopology: true });

Replace 'mongodb://localhost/your\_database\_name' with your MongoDB connection string.

JWT Secret Key

Open app.js and set your JWT secret key:

javascript

Copy code

const token = jwt.sign({ email: user.email }, 'your\_secret\_key');

Replace 'your\_secret\_key' with a secure secret key for JWT.

Email Service Provider

Open app.js and configure nodemailer with your email service provider:

javascript

Copy code

const transporter = nodemailer.createTransport({

service: 'your\_email\_service\_provider', // e.g., 'gmail'

auth: {

user: 'your\_email@example.com',

pass: 'your\_email\_password',

},

});

Replace 'your\_email\_service\_provider', 'your\_email@example.com', and 'your\_email\_password' with your email service provider details.

Endpoints

1. Register

Endpoint: POST /register

Request Body:

email (String, required): User's email address.

password (String, required): User's password.

Response:

Success (Status Code: 201):

json

Copy code

{ "message": "User registered successfully" }

Failure (Status Code: 500):

json

Copy code

{ "error": "Internal Server Error" }

2. Login

Endpoint: POST /login

Request Body:

email (String, required): User's email address.

password (String, required): User's password.

Response:

Success (Status Code: 200):

json

Copy code

{ "token": "your\_generated\_token" }

Invalid Credentials (Status Code: 401):

json

Copy code

{ "error": "Invalid credentials" }

Failure (Status Code: 500):

json

Copy code

{ "error": "Internal Server Error" }

3. Forget Password (Send Reset Link)

Endpoint: POST /forget-password

Request Body:

email (String, required): User's email address.

Response:

User Not Found (Status Code: 404):

json

Copy code

{ "error": "User not found" }

Success (Status Code: 200):

json

Copy code

{ "message": "Reset link sent successfully" }

Failure (Status Code: 500):

json

Copy code

{ "error": "Internal Server Error" }

4. Reset Password

Endpoint: POST /reset-password

Request Body:

newPassword (String, required): User's new password.

Response:

Success (Status Code: 200):

json

Copy code

{ "message": "Password reset successful" }

Failure (Status Code: 500):

json

Copy code

{ "error": "Internal Server Error" }

5. Get User Info

Endpoint: GET /user-info

Request Headers:

Authorization (String, required): Bearer token obtained during login.

Response:

Success (Status Code: 200):

json

Copy code

{ "email": "user@example.com" }

Unauthorized (Status Code: 401):

json

Copy code

{ "error": "Unauthorized" }

Forbidden (Status Code: 403):

json

Copy code

{ "error": "Forbidden" }

Usage Examples

Register a User:

bash

Copy code

curl -X POST -H "Content-Type: application/json" -d '{"email": "user@example.com", "password": "securepassword"}' http://localhost:3000/register

Login and Obtain Token:

bash

Copy code

curl -X POST -H "Content-Type: application/json" -d '{"email": "user@example.com", "password": "securepassword"}' http://localhost:3000/login

Output: {"token": "your\_generated\_token"}

Forget Password (Send Reset Link):

bash

Copy code

curl -X POST -H "Content-Type: application/json" -d '{"email": "user@example.com"}' http://localhost:3000/forget-password

Reset Password:

bash

Copy code

curl -X POST -H "Content-Type: application/json" -H "Authorization: Bearer your\_generated\_token" -d '{"newPassword": "newsecurepassword"}' http://localhost:3000/reset-password

Get User Info:

bash

Copy code

curl -H "Authorization: Bearer your\_generated\_token" http://localhost:3000/user-info

Middleware

1. Authenticate Token

This middleware ensures that the request has a valid JWT token in the Authorization header.

Usage: Applied to protected routes (e.g., /user-info, /reset-password).

Behavior:

If the token is missing, returns 401 Unauthorized.

If the token is invalid, returns 403 Forbidden.

If the token is valid, sets req.user to the decoded user information and allows the request to proceed.

Running the Server

To start the server, run the following command:

bash

Copy code

node app.js

The server will run on port 3000 by default. You can modify the port in the app.js file if needed.

**Conclusion**

This API provides basic user authentication and password reset functionality. Customize the configuration based on your environment, and use the provided endpoints and examples to integrate these features into your application.

**Nithin settibathula**

**7032460727**