Neuro

9YCOvAZNAiUqkMvTgOmo4VjtPPLe4ntCnAb9eBWXqr8WhbNmzm0QPqiHrf3JzBHp api dingo-neuroki

curl --location --request POST 'https://eu-west-2.aws.data.mongodb-api.com/app/data-tiobkyq/endpoint/data/v1/action/findOne' \

--header 'Content-Type: application/json' \

--header 'Access-Control-Request-Headers: \*' \

--header 'api-key: 9YCOvAZNAiUqkMvTgOmo4VjtPPLe4ntCnAb9eBWXqr8WhbNmzm0QPqiHrf3JzBHp' \

--header 'Accept: application/ejson' \

--data-raw '{

"collection":"questions",

"database":"test",

"dataSource":"Neuro",

"projection": {"\_id": 1}

}'

**Key Points to Verify:**

1. **MongoDB Connection**: Ensure that your MongoDB URI in the .env file is correct.
2. **Environment Variables**: Make sure you have the following in your .env file:

PORT=3002

MONGO\_URI=your-mongodb-connection-string

EMAIL\_USER=your-email@gmail.com

EMAIL\_PASS=your-email-password

1. **React Build**: Ensure that your React app is built and the static files are in the my-app/build directory.
2. **Model Definitions**: Ensure your Mongoose models (monthlysubscription, user, Questions) are correctly defined and located in the appropriate directory (./models/).

**Using Postman:**

To test your API endpoints, you can use Postman:

1. **Open Postman**.
2. **Create a New Request**:
   * **URL**: http://localhost:3002/api/questions
   * **Method**: GET
   * **Send**: Click "Send" and check the response.

For POST requests (e.g., /subscribe), you can:

1. **Create a New Request**:
   * **URL**: http://localhost:3002/subscribe
   * **Method**: POST
   * **Headers**:
     + Key: Content-Type
     + Value: application/json
   * **Body**:

json

Copy code

{

"email": "test@example.com"

}

* + **Send**: Click "Send" and check the response.

These steps should help you verify that your server and API are functioning correctly.

A small test website-application which is based Node.js, MongoDB and express.js. The side project is to test my: creativity, problem-solving abilities, common-sense, code-learning abilities and how these side hustling projects small ideas can be brought to life.

I am looking forward to Neuro.

Neuro is a site that welcomes anybody who wishes to take psychology test and give feedback tests as per their answers. The test will be checked by an AI assistant who would be under supervision and will hence give you a free response constructive feedback.

No Psychologist career is targeted here, it is just for my learning.

MDB: RZYAeKdm0Z87I7N6

mongodb+srv://faobilal28:RZYAeKdm0Z87I7N6@neuro.2x1dbxm.mongodb.net/

mongodb+srv://faobilal28:<password>@neuro.2x1dbxm.mongodb.net/

**function processResults(answers) {**

**// Implement your supervised learning algorithm here**

**// For demonstration, we'll return a dummy result**

**return "Your personality type is X based on your answers.";**

**} //112-116**

IN MY NAV-BAR-LEFT it is written other tests which is a drop down which has knowledge test, charcater test and personality test, Character test is where the major role of peela is to be so peela has multiple other siblings called: including peela :-> are: Peela Neela Jaamni Naaranjee Surkhh Siyah Sabzz Saafed

where each of these have a personality and once the test is perfomed end result is given to user as their executive result for example Jaamni, are given a code thats all time available using that they can play any game they want being that character the site has already asked them to subscribe on it, so the user is already registered via that one email and can only login with the option with sign-in withh google where the subcription is already done, fetches that and opens there account.

no need of asking user or password directly linked with gmail.

they can play only as that character each character to have a special power

CHARACTER TEST

**Summary**

1. Create the Character Test page with questions to determine the user's character.  
   **1.1. Character Definitions** Define each character's personality and special power. This information will be used to match users to characters.

**1.2. Character Test Page** Create a page or component for the Character Test. This will present questions or scenarios to users to determine their match.

1. Integrate Google Sign-In for user authentication and fetch their subscription data.  
   **2.1. Google Sign-In Integration** Integrate Google Sign-In to authenticate users. Ensure that users are automatically registered and logged in using their Google account.

**2.2. Subscription Handling** When users sign in, fetch their subscription data from your database and provide access to their character's unique features.

1. Use the character data to allow users to play games with unique features.  
   **3.1. Character-Based Access** Use the character code provided after the test to allow users to play games as their character.

**3.2. Special Powers** Incorporate the special powers of characters into the gameplay mechanics.

1. Ensure Peela and its siblings are animated properly on the home page.  
   **4.1. Animated Peela and Siblings** Ensure Peela and its siblings are animated on the home page as described earlier.

**4.2. Continuous Character Display** Have Peela and siblings continuously move across the screen as if they are on a news ticker.

<https://chatgpt.com/share/acee106a-7441-48d1-8dcc-a963712166d2>

Today was the dead line and I could not do it.

Connectivity of the components

**require**('dotenv').**config**();

const **express** = **require**('express');

const path = **require**('path');

const bodyParser = **require**('body-parser');

const **mongoose** = **require**('mongoose');

const cron = **require**('node-cron');

const nodemailer = **require**('nodemailer');

const app = **express**();

app.**use**(bodyParser.**urlencoded**({ extended: true }));

app.**use**(bodyParser.**json**());

*// MongoDB connection*

const mongoUri = process.env.MONGO\_URI;

if (!mongoUri) {

  console.**error**('MongoDB URI is not set in the environment variables');

  process.**exit**(1);

}

**mongoose**.**connect**(mongoUri, { useNewUrlParser: true, useUnifiedTopology: true })

  .**then**(() => console.**log**('MongoDB connected...'))

  .**catch**(err => {

    console.**error**('MongoDB connection error:', err);

    process.**exit**(1);

  });

*// Models*

const User = **require**('./models/user');

const Question = **require**('./models/question'); *// Ensure this path is correct*

const Result = **require**('./models/result'); *// Ensure this path is correct*

*// Nodemailer setup*

const transporter = nodemailer.**createTransport**({

  service: 'Gmail',

  auth: {

    user: process.env.EMAIL\_USER,

    pass: process.env.EMAIL\_PASS

  }

});

*// API routes*

app.**post**('/subscribe', async (req, res) => {

  const { email } = req.body;

  let user = await User.**findOne**({ email: email });

  if (!user) {

    user = new **User**({ email: email });

    await user.**save**();

    res.**send**('Subscription successful! You can now take the knowledge test.');

  } else {

    res.**send**('You are already registered.');

  }

});

app.**post**('/submit-test', async (req, res) => {

  const { email, score } = req.body;

  let user = await User.**findOne**({ email: email });

  if (!user) {

    return res.**status**(400).**send**('User not found. Please register first.');

  }

  const lastResult = user.testResults.length > 0 ? user.testResults[user.testResults.length - 1] : null;

  if (!lastResult || score > lastResult.score) {

    user.testResults.**push**({ score: score });

    await user.**save**();

    res.**send**('Test submitted successfully! Your score has been updated.');

  } else {

    res.**send**('Test submitted successfully! Your previous score is higher.');

  }

});

app.**get**('/api/questions', async (req, res) => {

  try {

    const { userId } = req.query;

    const alreadyAskedQuestions = await Result.**find**({ userId }).**select**('responses').**lean**();

    const askedQuestionIds = alreadyAskedQuestions.**flatMap**(result => result.responses.**map**(response => response.questionId));

    const newQuestions = await Question.**find**({ \_id: { $nin: askedQuestionIds } }).**limit**(10);

    res.**json**(newQuestions);

  } catch (err) {

    res.**status**(500).**send**('Error fetching questions');

  }

});

*// Catch-all handler to serve React's index.html for any route not handled by API routes*

app.**get**('\*', (req, res) => {

  res.**sendFile**(path.**join**(\_\_dirname, 'neuro-app/build', 'index.html'));

});

app.**use**(**express**.**static**(path.**join**(\_\_dirname, 'neuro-app/build')));

app.**listen**(3000, () => {

  console.**log**('Server started on port 3000');

});