Backend

Llm.js

```
const axios = require('axios');


const { GoogleGenerativeAI } = require("@google/generative-ai");

const apiKey = 'AIzaSyB-gYeps1h7YUPOSCNIDtB0Z4zBGDLzhoA';


const genAI = new GoogleGenerativeAI(apiKey);


const model = genAI.getGenerativeModel({ model: "gemini-1.5-flash"});


exports.generateNextQuestion = async (previousQuestion,userResponse) => {


  const prompt = ` Question: ${previousQuestion} User's response: ${userResponse}. Generate the next question. ` ;
  console.log(prompt,apiKey)
  try {
    const result = await model.generateContent(prompt);
    const response = await result.response;
    const text = response.text();
    console.log(text)
    return text;
  } catch (error) {
    console.error('Error generating question:', error);
    throw error;
  }
};
```

surveyController.js

```
const Survey = require('../models/Survey');
const axios = require('axios');
const { generateNextQuestion } = require('../config/llm');

exports.startSurvey = async (req, res) => {
  // console.log("Working")
  const previousQuestion = "How often do you exercise?";
  return res.json({ question: previousQuestion });
};

exports.getNextQuestion = async (req, res) => {
  const userResponse = req.body.response;
  const previousQuestion=req.body.pre;
  try {
    const nextQuestion = await generateNextQuestion(previousQuestion,userResponse);
    // console.log("nextQuestion",nextQuestion)
    res.json({ question: nextQuestion });
  } catch (error) {
    res.status(500).json({ error: "Error generating the next question" });
  }
};
```

Survey.js

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

const surveySchema = new mongoose.Schema({
  question: { type: String, required: true },
  response: { type: String },
});
```

```
module.exports = mongoose.model('Survey', surveySchema);
```

surveyroutes.js

```
const express = require('express');
const { startSurvey, getNextQuestion } = require('../controllers/surveyController');

const router = express.Router();

router.get('/start', startSurvey);
router.post('/nextQuestion', getNextQuestion);

module.exports = router;
```

index.js

```
const express = require('express');
const mongoose = require('mongoose');
const surveyRoutes = require('./routes/survey');
const dotenv = require('dotenv');
dotenv.config();
const cors = require("cors")
const app = express();

app.use(cors())

app.use(express.json());
app.use('/api/survey', surveyRoutes);

const PORT = process.env.PORT || 8000;
```

```
const connectDB = async () => {
 try {
  await mongoose.connect(process.env.MONGO_URI, {
   useNewUrlParser: true,
   useUnifiedTopology: true,
  });
  // console.log('MongoDB connected');

  app.listen(PORT, () => console.log(` Server running on port ${PORT}` ));
 } catch (error) {
  console.error('server connection error:', error);
  process.exit(1);
 }
};
// app.listen(PORT)

module.exports=connectDB();
```

Frontend

App.js

```
import React from 'react';
import Survey from './components/Survey';

function App() {
 return (
  <div className="App">
   <Survey />
  </div>
 );
}
```

```
export default App;
```

Survey.js

```
import React, { useState } from 'react';
import axios from 'axios';

const Survey = () => {
  const [questions, setQuestions] = useState([]);
  const [currentAnswer, setCurrentAnswer] = useState('');

  const fetchNextQuestion = async (response) => {
    try {
      const res = await axios.post('http://localhost:5000/api/survey/nextQuestion', { response,pre:questions[-1] });
      setQuestions([...questions, { question: res.data.question, response }]);
    } catch (error) {
      console.error('Error fetching next question:', error);
    }
  };

  const handleAnswerSubmit = (e) => {
    e.preventDefault();
    fetchNextQuestion(currentAnswer);
    setCurrentAnswer('');
  };

  const startSurvey = async () => {
    const res = await axios.get('http://localhost:5000/api/survey/start');
    setQuestions([{ question: res.data.question, response: '' }]);
```

```
  };

  return (
    <div>
      {questions.length === 0 && <button onClick={startSurvey}>Start Survey</button>}
      {questions.map((q, index) => (
        <div key={index}>
          <p>{q.question}</p>
          {index === questions.length - 1 && (
            <form onSubmit={handleAnswerSubmit}>
              <input
                type="text"
                value={currentAnswer}
                onChange={(e) => setCurrentAnswer(e.target.value)}
              />
              <button type="submit">Next</button>
            </form>
          )}
        </div>
      ))}
    </div>
  );
};

export default Survey;
```