Save user   
import { connectDB } from "@/lib/db"

import Student from "@/models/studentModel";

import { currentUser } from "@clerk/nextjs/server";

export async function GET() {

    try {

        await connectDB();

        const clerkUser=await currentUser()

         if (!clerkUser) {

      return Response.json({ error: "Unauthorized" }, { status: 401 });

    }

    const existingStudent=await Student.findOne({

        clerkId:clerkUser.id

    })

    if(!existingStudent){

        const newStudent=await Student.create({

                 clerkId: clerkUser.id,

        email: clerkUser.emailAddresses[0].emailAddress,

        name: clerkUser.fullName,

        })

        return Response.json({ student: newStudent, status: "created" });

    }

    // console.log(existingStudent)

   return Response.json({ student: existingStudent, status: "existing" });

    } catch (error) {

        console.log(error)

        console.error("error saving the user ",error)

    }

}

Subjects/create/id   
import { connectDB } from "@/lib/db";

import Student from "@/models/studentModel";

import Subject from "@/models/subjectModel";

export async function PATCH(req, { params }) {

  await connectDB();

  const { id } = await params;

console.log(id)

  const { name } = await req.json();

  try {

    const updatedSubject = await Subject.findByIdAndUpdate(

      id,

      {

        ...(name && { name:name }),

      },

      { new: true }

    );

    return new Response(JSON.stringify({ subject: updatedSubject }), {

      status: 200,

    });

  } catch (error) {

    console.error("PATCH error:", error);

    return new Response(JSON.stringify({ error: "Internal Server Error" }), {

      status: 500,

    });

  }

}

subjects/create   
import { connectDB } from "@/lib/db";

import Student from "@/models/studentModel";

import Subject from "@/models/subjectModel";

import { currentUser } from "@clerk/nextjs/server";

export async function POST(req) {

  try {

    await connectDB();

    const clerkUser = await currentUser();

    if (!clerkUser) {

      return new Response(JSON.stringify({ error: "Unauthorized" }), { status: 401 });

    }

    const thisStudent = await Student.findOne({ clerkId: clerkUser.id });

    const reqBody = await req.json();

    const { name } = reqBody;

    const existingSubject = await Subject.findOne({ name });

    if (existingSubject) {

      // Check if student is already linked to the subject

      const isAlreadyLinked = existingSubject.student.some(

        (stdId) => stdId.toString() === thisStudent.\_id.toString()

      );

      if (!isAlreadyLinked) {

        existingSubject.student.push(thisStudent.\_id);

        await existingSubject.save();

        thisStudent.subjects.push(existingSubject.\_id);

        await thisStudent.save();

        return new Response(

          JSON.stringify({ subject: existingSubject, status: "linked-to-existing" }),

          { status: 200 }

        );

      }

      // Student already linked to existing subject

      return new Response(

        JSON.stringify({ error: "Subject already linked to student" }),

        { status: 400 }

      );

    }

    // If subject does not exist, create a new one

    const newSubject = await Subject.create({

      name,

      student: [thisStudent.\_id],

    });

    thisStudent.subjects.push(newSubject.\_id);

    await thisStudent.save();

    return new Response(

      JSON.stringify({ subject: newSubject, status: "created-new" }),

      { status: 201 }

    );

  } catch (error) {

    console.error("Error while creating subject:", error);

    return new Response(JSON.stringify({ error: "Internal Server Error" }), { status: 500 });

  }

}

Subjects/goals/create/id  
import { connectDB } from "@/lib/db";

import Goal from "@/models/goalModel";

import Student from "@/models/studentModel";

import Subject from "@/models/subjectModel";

export async function PATCH(req, { params }) {

  await connectDB();

  const { id } = await params;

console.log(id)

  const { title, deadline, description } = await req.json();

  try {

    const updatedGoal = await Goal.findByIdAndUpdate(

      id,

      {

        ...(title && { title }),

        ...(deadline && { deadline }),

        ...(description && { description }),

      },

      { new: true }

    );

    return new Response(JSON.stringify({ goal: updatedGoal }), {

      status: 200,

    });

  } catch (error) {

    console.error("PATCH error:", error);

    return new Response(JSON.stringify({ error: "Internal Server Error" }), {

      status: 500,

    });

  }

}

export async function DELETE(req,{params}){

    await connectDB();

    const {id}= params;

    console.log("ye hai id", params)

    try {

        const deletedGoal=await Goal.findByIdAndDelete(id);

        console.log(deletedGoal)

        if(!deletedGoal) {

            console.log("j")

            return new Response(JSON.stringify({

                erorr:"Goal not found or error deleting goal"

            }),{status:404})

        }

        // removing from students

        await Student.updateOne(

            {\_id:deletedGoal.student},

            { $pull: { goals: id } }

        )

        // removing from subjects

        await Subject.updateOne(

            {\_id:deletedGoal.subject},

            { $pull: { goals: id } }

        )

          return new Response(JSON.stringify({ status: "deleted", id }), {

      status: 200,

    });

    } catch (error) {

         console.error("DELETE error:", err);

    return new Response(JSON.stringify({ error: "Failed to delete goal" }), {

      status: 500,

    });

    }

}

Subjects/goals/creatr  
import { connectDB } from "@/lib/db";

import Goal from "@/models/goalModel";

import Student from "@/models/studentModel";

import Subject from "@/models/subjectModel";

import { currentUser } from "@clerk/nextjs/server";

// import { connect } from "mongoose";

export  async function POST(req){

try {

    await connectDB();

    const reqBody=await req.json();

    const {subject,title,deadline,description}=reqBody;

    console.log("request body",reqBody)

    const clerkUser = await currentUser();

    if (!clerkUser) {

      return new Response(JSON.stringify({ error: "Unauthorized" }), { status: 401 });

    }

   const thisStudent = await Student.findOne({ clerkId: clerkUser.id }).populate("subjects").populate("goals")

   console.log("thisStudent",thisStudent)

   const thisSubject=await Subject.findOne({name:subject})

   const goal={subject:thisSubject.\_id,

    title,

    deadline,

    student:thisStudent.\_id,

    description};

   const createGoal=await Goal.create(goal);

   console.log("naya goal" ,createGoal)

thisSubject.goals.push(createGoal);

thisStudent.goals.push(createGoal)

await thisSubject.save()

await thisStudent.save()

  return new Response(

      JSON.stringify({ goal: createGoal, status: "created-new" }),

      { status: 201 }

    );

} catch (error) {

    console.error("Error while creating the goal:", error);

    return new Response(JSON.stringify({ error: "Internal Server Error" }), { status: 500 });

}

}

export async function PATCH(req) {

  try {

    await connectDB();

    const body = await req.json();

    const { goalId, title, deadline, description } = body;

    if (!goalId) {

      return new Response(JSON.stringify({ error: "Missing goalId" }), {

        status: 400,

      });

    }

    // Find and update the goal

    const updatedGoal = await Goal.findByIdAndUpdate(

      goalId,

      {

        ...(title && { title }),

        ...(deadline && { deadline }),

        ...(description && { description }),

      },

      { new: true } // Return the updated document

    );

    if (!updatedGoal) {

      return new Response(JSON.stringify({ error: "Goal not found" }), {

        status: 404,

      });

    }

    return new Response(JSON.stringify({ goal: updatedGoal }), {

      status: 200,

    });

  } catch (error) {

    console.error("Error while updating goal:", error);

    return new Response(JSON.stringify({ error: "Internal Server Error" }), {

      status: 500,

    });

  }

}

Subject/goals/update

import { connectDB } from "@/lib/db";

import Goal from "@/models/goalModel";

export async function PATCH(req) {

  await connectDB();

  try {

    const reqBody = await req.json();

    const { goalId, status } = reqBody;

    console.log("Request Body:", reqBody);

    const goal=await Goal.findByIdAndUpdate(

      goalId,

      {

        ...(status !== undefined && { completionStatus: status }),

      },

      { new: true }

    );

    console.log(goal)

    return new Response(JSON.stringify({ goal:goal, message: "Goal updated successfully" }), {

      status: 200,

    });

  } catch (error) {

    console.error("Error updating goal:", error);

    return new Response(JSON.stringify({ error: "Failed to update goal" }), {

      status: 500,

    });

  }

}

Subjects/users  
import { connectDB } from "@/lib/db";

import Student from "@/models/studentModel";

import { currentUser } from "@clerk/nextjs/server";

import Subject from "@/models/subjectModel";

import Goal from "@/models/goalModel";

import Session from "@/models/sessionModel";

export async function GET(){

    try {

        await connectDB();

    const clerkUser = await currentUser();

    console.log("clerkUser",clerkUser)

    if (!clerkUser) {

      return new Response(JSON.stringify({ error: "Unauthorized" }), { status: 401 });

    }

    const thisStudent = await Student.findOne({ clerkId: clerkUser.id })

  .populate("goals")

  .populate("subjects")

  .populate("sessions")

    return Response.json({ student: thisStudent, status: "existing" });

    } catch (error) {

        console.error("Error while getting data:", error);

    return new Response(JSON.stringify({ error: "Internal Server Error" }), { status: 500 });

    }

}

sessions/general  
import { connectDB } from "@/lib/db";

import Session from "@/models/sessionModel";

import Student from "@/models/studentModel";

import { currentUser } from "@clerk/nextjs/server";

export async function POST(req){

    await connectDB();

    try {

        const reqBody=await req.json()

        const {date,startTime,endTime}= reqBody;

        console.log(reqBody);

        const clerkUser = await currentUser();

         if (!clerkUser) {

      return new Response(JSON.stringify({ error: "Unauthorized" }), { status: 401 });

        }

    const thisStudent = await Student.findOne({ clerkId: clerkUser.id });

if(!thisStudent) throw new Error("no studenet fid")

    const newSession=await Session.create({

        student:thisStudent,

        date,

        startTime,

        endTime

    })

    if(!newSession) throw new Error("failed cresting the session")

        thisStudent.sessions.push(newSession)

    await thisStudent.save();

 return new Response(JSON.stringify({ session: newSession }), {

      status: 200,

    });

  } catch (error) {

    console.error("Error while creating sessiion:", error);

    return new Response(JSON.stringify({ error: "Internal Server Error" }), { status: 500 });

  }

}

subjects/fix-goals  
import { connectDB } from "@/lib/db";

import Student from "@/models/studentModel";

export async function PATCH() {

  try {

    await connectDB();

    const result = await Student.updateMany(

      { goals: { $exists: false } },

      { $set: { goals: [] } }

    );

    return Response.json({

      message: "Goals field initialized",

      matched: result.matchedCount,

      modified: result.modifiedCount,

    });

  } catch (error) {

    console.error("Fix goals error:", error);

    return new Response(JSON.stringify({ error: "Failed to update students" }), { status: 500 });

  }

}

Usercontext  
'use client';

import { createContext, useContext, useState, useEffect } from 'react';

const UserContext = createContext();

export const UserProvider = ({ children }) => {

  const [user, setUser] = useState(null);

  const [loading, setLoading] = useState(true);

  useEffect(() => {

    const fetchUser = async () => {

      try {

        const res = await fetch('/api/subjects/user');

        const data = await res.json();

        setUser(data);

        console.log(data)

      } catch (err) {

        console.error('Failed to fetch user', err);

      } finally {

        setLoading(false);

      }

    };

    fetchUser();

  }, []);

  return (

    <UserContext.Provider value={{ user, setUser, loading }}>

      {children}

    </UserContext.Provider>

  );

};

export const useUser = () => useContext(UserContext);

db  
import mongoose from "mongoose"

//yaha db connection dena hai

export async function connectDB() {

    const  DBNAME="/WORKFLOW"

    const name=process.env.MONGO\_DB\_URI+DBNAME

    try {

        mongoose.connect(name)

        const connection=mongoose.connection

        connection.on('connected',()=>{

            console.log("Mongo Db jud gya")

        })

        connection.on('error',(err)=>{

            console.log("Mongo Db uddd gya, db is not running"+err)

            process.exit()

        })

    } catch (error) {

        console.log("something went wrong is connceting to db ye error",error)

    }

}

// middleware.js

import { clerkMiddleware } from "@clerk/nextjs/server";

export default clerkMiddleware();

// Adjust route matching here

export const config = {

  matcher: [

    "/((?!\_next|.\*\\..\*|favicon.ico).\*)", // skip static files and \_next

    "/", // include root route

    "/(api|trpc)(.\*)", // include API and TRPC

  ],

};

// models/Session.js

import mongoose from 'mongoose';

const sessionSchema = new mongoose.Schema(

  {

    student:{

       type: mongoose.Schema.Types.ObjectId,

      ref: 'Student',

      required:true

    },

    goal: {

      type: mongoose.Schema.Types.ObjectId,

      ref: 'Goal',

    },

    date: {

      type: Date,

      required: true,

    },

    startTime: {

      type: String,

      required: true,

    },

    endTime: {

      type: String,

      required: true,

    },

  },

  {

    timestamps: true,

  }

);

const Session=mongoose.models.Session || mongoose.model('Session', sessionSchema);

export default Session

import mongoose from "mongoose";

const goalSchema = new mongoose.Schema(

  {

    subject: {

      type: mongoose.Schema.Types.ObjectId,

      ref: "Subject",

      required: true,

    },

    student: {

      type: mongoose.Schema.Types.ObjectId,

      ref: "Student",

    },

    deadline: {

      type: Date,

      required: true,

    },

    completionStatus: {

      type: String,

      default: "pending",

    },

    title:{

        type:String,

        required:true

    },

    description:{

      type:String,

      required:true,

    },

    sessions:[{

       type: mongoose.Schema.Types.ObjectId,

      ref: "Session",

    }]

  },

  {

    timestamps: true,

  }

);

// Use existing model if already compiled, otherwise create new one

const Goal = mongoose.models.Goal || mongoose.model("Goal", goalSchema);

export default Goal;

import mongoose from "mongoose";

const studentSchema=new mongoose.Schema({

    clerkId:{

        type:String,

        required:true,

        unique:true

    },

    name:{

           type:String,

        required:true

    },

    email:{

           type:String,

        required:true

    },

    subjects:[{

      type:mongoose.Schema.Types.ObjectId,

      ref:"Subject"

    }],

    goals:[{

      type:mongoose.Schema.Types.ObjectId,

      ref:"Goal"

    }],

    sessions:[

      {

         type: mongoose.Schema.Types.ObjectId,

      ref: 'Session',

      }

    ]

})

const Student=mongoose.models.Student || mongoose.model("Student",studentSchema)

export default Student

import mongoose from "mongoose";

const subjectSchema=new mongoose.Schema({

    name:{

           type:String,

        required:true,

        unique:true

    },

    student:[{

      type:mongoose.Schema.Types.ObjectId,

      ref:"Student",

      required:true

    }],

    topics:[{

        type:mongoose.Schema.Types.ObjectId,

      ref:"Topic"

    }],

    goals:[{

        type:mongoose.Schema.Types.ObjectId,

      ref:"Goal"

    }],

})

const Subject=mongoose.models.Subject || mongoose.model("Subject",subjectSchema)

export default Subject

import mongoose from 'mongoose';

const topicSchema = new mongoose.Schema(

  {

    title: {

      type: String,

      required: true,

    },

    difficulty: {

      type: String, // You can also use enum if you want to restrict values

      enum: ['Easy', 'Medium', 'Hard'],

      required: true,

    },

    subject: {

      type: mongoose.Schema.Types.ObjectId,

      ref: 'Subject',

      required: true,

    },

  },

  {

    timestamps: true,

  }

);

export default mongoose.models.Topic || mongoose.model('Topic', topicSchema);