# AI-Powered Smart Study Mentor Platform - Complete Source Code

## File Structure:

project/

├── Layout.js

├── entities/

│ ├── User.json

│ ├── Subject.json

│ ├── StudyPlan.json

│ ├── Question.json

│ ├── StudySession.json

│ ├── LearningMaterial.json

│ ├── Quiz.json

│ ├── FlashcardSet.json

│ └── Achievement.json

├── pages/

│ ├── Dashboard.js

│ ├── QAAssistant.js

│ ├── StudyPlans.js

│ ├── Subjects.js

│ ├── Progress.js

│ └── Materials.js

└── components/

├── dashboard/

│ ├── WelcomeHeader.jsx

│ ├── QuickActions.jsx

│ ├── UpcomingStudySessions.jsx

│ └── RecentActivity.jsx

├── qa/

│ ├── QuestionForm.jsx

│ └── AnswerDisplay.jsx

└── ui/

└── DynamicIcon.jsx

# ------------------- ENTITIES -------------------

## entities/User.json

{

"name": "User",

"type": "object",

"properties": {

"learning\_style": {

"type": "string",

"enum": ["visual", "auditory", "kinesthetic", "reading\_writing", "balanced"],

"default": "balanced",

"description": "The user's preferred learning style."

},

"learning\_pace": {

"type": "string",

"enum": ["slow", "medium", "fast"],

"default": "medium",

"description": "The user's preferred learning pace."

},

"strengths": {

"type": "array",

"items": { "type": "string" },

"description": "AI-identified academic strengths."

},

"weaknesses": {

"type": "array",

"items": { "type": "string" },

"description": "AI-identified areas for improvement."

}

}

}

## entities/Subject.json

{

"name": "Subject",

"type": "object",

"properties": {

"name": {

"type": "string",

"description": "Subject name (e.g., Mathematics, Physics, History)"

},

"description": {

"type": "string",

"description": "Brief description of the subject"

},

"difficulty\_level": {

"type": "string",

"enum": ["beginner", "intermediate", "advanced"],

"default": "beginner",

"description": "Student's current level in this subject"

},

"weekly\_hours": {

"type": "number",

"description": "Hours per week allocated to this subject",

"default": 5

},

"priority": {

"type": "string",

"enum": ["low", "medium", "high"],

"default": "medium",

"description": "Study priority for this subject"

},

"color": {

"type": "string",

"description": "Color code for visual representation",

"default": "#3b82f6"

}

},

"required": ["name"]

}

## entities/StudyPlan.json

{

"name": "StudyPlan",

"type": "object",

"properties": {

"title": {

"type": "string",

"description": "Study plan title"

},

"subject\_id": {

"type": "string",

"description": "Reference to the subject"

},

"description": {

"type": "string",

"description": "Detailed description of the study plan"

},

"duration\_weeks": {

"type": "number",

"description": "Duration of the study plan in weeks",

"default": 4

},

"daily\_hours": {

"type": "number",

"description": "Recommended daily study hours",

"default": 1

},

"difficulty\_level": {

"type": "string",

"enum": ["beginner", "intermediate", "advanced"],

"default": "beginner"

},

"topics": {

"type": "array",

"description": "List of topics to cover",

"items": {

"type": "object",

"properties": {

"name": {"type": "string"},

"estimated\_hours": {"type": "number"},

"resources": {"type": "array", "items": {"type": "string"}},

"completed": {"type": "boolean", "default": false}

}

}

},

"goals": {

"type": "array",

"description": "Learning objectives",

"items": {"type": "string"}

},

"status": {

"type": "string",

"enum": ["active", "completed", "paused"],

"default": "active"

},

"progress\_percentage": {

"type": "number",

"description": "Completion percentage",

"default": 0

}

},

"required": ["title", "subject\_id"]

}

## entities/Question.json

{

"name": "Question",

"type": "object",

"properties": {

"question\_text": {

"type": "string",

"description": "The student's question"

},

"subject": {

"type": "string",

"description": "Subject category of the question"

},

"difficulty\_level": {

"type": "string",

"enum": ["beginner", "intermediate", "advanced"],

"description": "Estimated difficulty level"

},

"ai\_response": {

"type": "string",

"description": "AI-generated response to the question"

},

"sources": {

"type": "array",

"description": "Sources used for the response",

"items": {"type": "string"}

},

"helpful\_rating": {

"type": "number",

"description": "Student rating of response helpfulness (1-5)",

"minimum": 1,

"maximum": 5

},

"tags": {

"type": "array",

"description": "Tags for categorizing questions",

"items": {"type": "string"}

}

},

"required": ["question\_text"]

}

## entities/StudySession.json

{

"name": "StudySession",

"type": "object",

"properties": {

"subject\_id": {

"type": "string",

"description": "Reference to the subject studied"

},

"study\_plan\_id": {

"type": "string",

"description": "Reference to the study plan"

},

"duration\_minutes": {

"type": "number",

"description": "Duration of study session in minutes"

},

"topics\_covered": {

"type": "array",

"description": "Topics studied in this session",

"items": {"type": "string"}

},

"notes": {

"type": "string",

"description": "Session notes and reflections"

},

"effectiveness\_rating": {

"type": "number",

"description": "How effective the session was (1-5)",

"minimum": 1,

"maximum": 5

},

"challenges\_faced": {

"type": "array",

"description": "Difficulties encountered during study",

"items": {"type": "string"}

},

"completed\_goals": {

"type": "array",

"description": "Goals achieved in this session",

"items": {"type": "string"}

}

},

"required": ["subject\_id", "duration\_minutes"]

}

## entities/LearningMaterial.json

{

"name": "LearningMaterial",

"type": "object",

"properties": {

"title": {

"type": "string",

"description": "Title of the learning material (e.g., 'Chapter 3 Notes')"

},

"file\_url": {

"type": "string",

"description": "URL of the uploaded file"

},

"file\_type": {

"type": "string",

"enum": ["pdf", "text", "image", "lecture\_notes"],

"description": "Type of the uploaded file"

},

"subject\_id": {

"type": "string",

"description": "Reference to the associated subject"

},

"summary": {

"type": "string",

"description": "AI-generated summary of the material"

},

"keywords": {

"type": "array",

"items": { "type": "string" },

"description": "AI-extracted keywords"

}

},

"required": ["title", "file\_url"]

}

## entities/Quiz.json

{

"name": "Quiz",

"type": "object",

"properties": {

"title": { "type": "string" },

"subject\_id": { "type": "string" },

"study\_plan\_topic": { "type": "string" },

"questions": {

"type": "array",

"items": {

"type": "object",

"properties": {

"question\_text": { "type": "string" },

"options": { "type": "array", "items": { "type": "string" } },

"correct\_answer\_index": { "type": "number" },

"explanation": { "type": "string" }

}

}

},

"score": { "type": "number" },

"completed\_date": { "type": "string", "format": "date-time" }

},

"required": ["title", "subject\_id"]

}

## entities/FlashcardSet.json

{

"name": "FlashcardSet",

"type": "object",

"properties": {

"title": { "type": "string" },

"subject\_id": { "type": "string" },

"study\_plan\_topic": { "type": "string" },

"flashcards": {

"type": "array",

"items": {

"type": "object",

"properties": {

"front": { "type": "string" },

"back": { "type": "string" }

}

}

}

},

"required": ["title", "subject\_id"]

}

## entities/Achievement.json

{

"name": "Achievement",

"type": "object",

"properties": {

"title": { "type": "string" },

"description": { "type": "string" },

"icon": { "type": "string", "description": "Lucide icon name" },

"color": { "type": "string", "description": "Tailwind color class" },

"unlocked\_date": { "type": "string", "format": "date-time" }

},

"required": ["title", "icon"]

}

# ------------------- LAYOUT -------------------

## Layout.js

import React from "react";

import { Link, useLocation } from "react-router-dom";

import { createPageUrl } from "@/utils";

import {

Home,

Calendar,

MessageCircleQuestion,

TrendingUp,

BookOpen,

Brain,

User,

Settings,

FileText

} from "lucide-react";

import {

Sidebar,

SidebarContent,

SidebarGroup,

SidebarGroupContent,

SidebarGroupLabel,

SidebarMenu,

SidebarMenuButton,

SidebarMenuItem,

SidebarHeader,

SidebarFooter,

SidebarProvider,

SidebarTrigger,

} from "@/components/ui/sidebar";

const navigationItems = [

{

title: "Dashboard",

url: createPageUrl("Dashboard"),

icon: Home,

},

{

title: "Study Plans",

url: createPageUrl("StudyPlans"),

icon: Calendar,

},

{

title: "AI Q&A Assistant",

url: createPageUrl("QAAssistant"),

icon: MessageCircleQuestion,

},

{

title: "My Materials",

url: createPageUrl("Materials"),

icon: FileText,

},

{

title: "Progress & Insights",

url: createPageUrl("Progress"),

icon: TrendingUp,

},

{

title: "Subjects",

url: createPageUrl("Subjects"),

icon: BookOpen,

},

];

export default function Layout({ children, currentPageName }) {

const location = useLocation();

return (

<SidebarProvider>

<style>{`

:root {

--primary-blue: #1e40af;

--secondary-blue: #3b82f6;

--accent-green: #10b981;

--accent-orange: #f59e0b;

--neutral-50: #f8fafc;

--neutral-100: #f1f5f9;

--neutral-200: #e2e8f0;

--neutral-600: #475569;

--neutral-700: #334155;

--neutral-800: #1e293b;

--neutral-900: #0f172a;

}

body {

font-family: 'Inter', -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;

background: linear-gradient(135deg, var(--neutral-50) 0%, #e0f2fe 100%);

}

.sidebar-gradient {

background: linear-gradient(180deg, var(--primary-blue) 0%, var(--secondary-blue) 100%);

}

.logo-glow {

filter: drop-shadow(0 0 20px rgba(59, 130, 246, 0.3));

}

`}</style>

<div className="min-h-screen flex w-full">

<Sidebar className="border-r-0 shadow-xl">

<div className="sidebar-gradient h-full">

<SidebarHeader className="border-b border-blue-400/30 p-6">

<div className="flex items-center gap-3">

<div className="w-10 h-10 bg-white/20 backdrop-blur-sm rounded-xl flex items-center justify-center logo-glow">

<Brain className="w-6 h-6 text-white" />

</div>

<div>

<h2 className="font-bold text-white text-lg">StudyMentor AI</h2>

<p className="text-xs text-blue-100">Smart Learning Platform</p>

</div>

</div>

</SidebarHeader>

<SidebarContent className="p-3">

<SidebarGroup>

<SidebarGroupLabel className="text-xs font-medium text-blue-100 uppercase tracking-wider px-3 py-3">

Navigation

</SidebarGroupLabel>

<SidebarGroupContent>

<SidebarMenu className="space-y-1">

{navigationItems.map((item) => (

<SidebarMenuItem key={item.title}>

<SidebarMenuButton

asChild

className={\`hover:bg-white/20 text-white hover:text-white transition-all duration-200 rounded-xl mx-2 \${

location.pathname === item.url ? 'bg-white/20 shadow-lg' : ''

}\`}

>

<Link to={item.url} className="flex items-center gap-3 px-4 py-3">

<item.icon className="w-5 h-5" />

<span className="font-medium">{item.title}</span>

</Link>

</SidebarMenuButton>

</SidebarMenuItem>

))}

</SidebarMenu>

</SidebarGroupContent>

</SidebarGroup>

<SidebarGroup className="mt-8">

<SidebarGroupLabel className="text-xs font-medium text-blue-100 uppercase tracking-wider px-3 py-3">

Quick Stats

</SidebarGroupLabel>

<SidebarGroupContent>

<div className="px-5 py-3 space-y-3">

<div className="flex items-center gap-3 text-sm">

<Calendar className="w-4 h-4 text-blue-200" />

<div className="flex-1">

<span className="text-blue-100">Study Streak</span>

<div className="text-white font-semibold">7 days</div>

</div>

</div>

<div className="flex items-center gap-3 text-sm">

<TrendingUp className="w-4 h-4 text-blue-200" />

<div className="flex-1">

<span className="text-blue-100">Progress</span>

<div className="text-white font-semibold">68%</div>

</div>

</div>

</div>

</SidebarGroupContent>

</SidebarGroup>

</SidebarContent>

<SidebarFooter className="border-t border-blue-400/30 p-4">

<div className="flex items-center gap-3 text-white">

<div className="w-10 h-10 bg-white/20 rounded-full flex items-center justify-center">

<User className="w-5 h-5" />

</div>

<div className="flex-1 min-w-0">

<p className="font-medium text-sm truncate">Student</p>

<p className="text-xs text-blue-100 truncate">Learning Journey</p>

</div>

<Settings className="w-4 h-4 text-blue-200 cursor-pointer hover:text-white transition-colors" />

</div>

</SidebarFooter>

</div>

</Sidebar>

<main className="flex-1 flex flex-col">

<header className="bg-white/80 backdrop-blur-sm border-b border-gray-200/50 px-6 py-4 md:hidden shadow-sm">

<div className="flex items-center gap-4">

<SidebarTrigger className="hover:bg-gray-100 p-2 rounded-lg transition-colors duration-200" />

<h1 className="text-xl font-bold text-gray-900">StudyMentor AI</h1>

</div>

</header>

<div className="flex-1 overflow-auto">

{children}

</div>

</main>

</div>

</SidebarProvider>

);

}

# ------------------- PAGES -------------------

## pages/Dashboard.js

import React, { useState, useEffect } from "react";

import { Subject, StudyPlan, Question, StudySession } from "@/entities/all";

import WelcomeHeader from "../components/dashboard/WelcomeHeader";

import QuickActions from "../components/dashboard/QuickActions";

import UpcomingStudySessions from "../components/dashboard/UpcomingStudySessions";

import RecentActivity from "../components/dashboard/RecentActivity";

export default function Dashboard() {

const [subjects, setSubjects] = useState([]);

const [studyPlans, setStudyPlans] = useState([]);

const [recentQuestions, setRecentQuestions] = useState([]);

const [isLoading, setIsLoading] = useState(true);

useEffect(() => {

loadDashboardData();

}, []);

const loadDashboardData = async () => {

setIsLoading(true);

try {

const [subjectsData, studyPlansData, questionsData] = await Promise.all([

Subject.list('-updated\_date', 5),

StudyPlan.list('-updated\_date', 3),

Question.list('-created\_date', 5)

]);

setSubjects(subjectsData);

setStudyPlans(studyPlansData);

setRecentQuestions(questionsData);

} catch (error) {

console.error('Error loading dashboard data:', error);

}

setIsLoading(false);

};

return (

<div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-indigo-50 p-4 md:p-8">

<div className="max-w-7xl mx-auto">

<WelcomeHeader />

<div className="grid grid-cols-1 lg:grid-cols-3 gap-8 mb-8">

<div className="lg:col-span-2">

<QuickActions />

</div>

<div>

<UpcomingStudySessions />

</div>

</div>

<div className="grid grid-cols-1 lg:grid-cols-2 gap-8">

<RecentActivity />

<div className="bg-white/80 backdrop-blur-sm rounded-2xl p-6 shadow-lg border-0">

<h3 className="text-xl font-bold text-gray-900 mb-4">Study Progress Overview</h3>

<div className="space-y-4">

<div className="flex items-center justify-between">

<span className="text-gray-600">Weekly Goal</span>

<span className="font-semibold text-blue-600">68% Complete</span>

</div>

<div className="w-full bg-gray-200 rounded-full h-3">

<div className="bg-gradient-to-r from-blue-500 to-blue-600 h-3 rounded-full" style={{width: '68%'}}></div>

</div>

<div className="flex items-center justify-between mt-6">

<span className="text-gray-600">Study Streak</span>

<span className="font-semibold text-green-600">7 Days</span>

</div>

<div className="w-full bg-gray-200 rounded-full h-3">

<div className="bg-gradient-to-r from-green-500 to-green-600 h-3 rounded-full" style={{width: '87%'}}></div>

</div>

<div className="mt-6 p-4 bg-gradient-to-r from-blue-50 to-indigo-50 rounded-xl">

<p className="text-sm text-blue-800 font-medium">

🎉 Great progress! You're on track to achieve your weekly study goals.

</p>

</div>

</div>

</div>

</div>

</div>

</div>

);

}

## pages/QAAssistant.js

import React, { useState, useEffect } from "react";

import { Question, LearningMaterial } from "@/entities/all";

import { InvokeLLM } from "@/integrations/Core";

import QuestionForm from "../components/qa/QuestionForm";

import AnswerDisplay from "../components/qa/AnswerDisplay";

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Badge } from "@/components/ui/badge";

import { MessageCircleQuestion, History, TrendingUp, BrainCircuit } from 'lucide-react';

import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";

import { Label } from "@/components/ui/label";

export default function QAAssistant() {

const [questions, setQuestions] = useState([]);

const [currentQuestion, setCurrentQuestion] = useState(null);

const [isLoading, setIsLoading] = useState(false);

const [recentQuestions, setRecentQuestions] = useState([]);

const [materials, setMaterials] = useState([]);

const [selectedMaterialId, setSelectedMaterialId] = useState('');

useEffect(() => {

loadData();

}, []);

const loadData = async () => {

const [recent, materialsData] = await Promise.all([

Question.list('-created\_date', 10),

LearningMaterial.list('-created\_date')

]);

setRecentQuestions(recent);

setMaterials(materialsData);

};

const handleSubmitQuestion = async (questionData) => {

setIsLoading(true);

setCurrentQuestion(null);

try {

let prompt = \`You are an expert academic tutor. Please provide a comprehensive, accurate answer to this \${questionData.difficulty\_level}-level \${questionData.subject} question:

Question: \${questionData.question\_text}\`;

let file\_urls = [];

if (selectedMaterialId) {

const selectedMaterial = materials.find(m => m.id === selectedMaterialId);

if(selectedMaterial) {

prompt += \`\n\nBase your answer primarily on the provided context from the user's study material titled "\${selectedMaterial.title}".\`;

file\_urls.push(selectedMaterial.file\_url);

}

} else {

prompt += \`\n\nPlease provide a general answer based on your broad knowledge.\`

}

const aiResponse = await InvokeLLM({

prompt: prompt,

file\_urls: file\_urls.length > 0 ? file\_urls : undefined,

add\_context\_from\_internet: file\_urls.length === 0,

});

const newQuestion = await Question.create({

...questionData,

ai\_response: aiResponse,

sources: ["Academic AI Assistant", "Educational Database"],

tags: [questionData.subject.toLowerCase(), questionData.difficulty\_level]

});

setCurrentQuestion(newQuestion);

setQuestions(prev => [newQuestion, ...prev]);

await loadData();

} catch (error) {

console.error('Error processing question:', error);

}

setIsLoading(false);

};

const handleRating = async (questionId, rating) => {

await Question.update(questionId, { helpful\_rating: rating });

setCurrentQuestion(prev => prev ? { ...prev, helpful\_rating: rating } : null);

};

const selectRecentQuestion = (question) => {

setCurrentQuestion(question);

};

return (

<div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-indigo-50 p-4 md:p-8">

<div className="max-w-6xl mx-auto">

<div className="mb-8">

<h1 className="text-3xl md:text-4xl font-bold text-gray-900 mb-2">

AI Study Assistant

</h1>

<p className="text-xl text-gray-600">

Get instant, personalized answers to your academic questions

</p>

</div>

<div className="grid grid-cols-1 lg:grid-cols-3 gap-8">

<div className="lg:col-span-2 space-y-6">

<Card className="shadow-xl border-0 bg-white/90 backdrop-blur-sm">

<CardHeader>

<CardTitle className="flex items-center gap-2 text-xl font-bold text-gray-900">

<BrainCircuit className="w-6 h-6 text-blue-600" />

Context-Aware Assistance

</CardTitle>

</CardHeader>

<CardContent>

<div className="space-y-2">

<Label htmlFor="material-context">Use My Study Material (Optional)</Label>

<Select value={selectedMaterialId} onValueChange={setSelectedMaterialId}>

<SelectTrigger id="material-context">

<SelectValue placeholder="General knowledge (no specific material)" />

</SelectTrigger>

<SelectContent>

<SelectItem value={null}>General knowledge</SelectItem>

{materials.map(material => (

<SelectItem key={material.id} value={material.id}>

{material.title}

</SelectItem>

))}

</SelectContent>

</Select>

<p className="text-xs text-gray-500">Select one of your uploaded materials to get answers specific to your course content.</p>

</div>

</CardContent>

</Card>

<QuestionForm

onSubmitQuestion={handleSubmitQuestion}

isLoading={isLoading}

/>

{currentQuestion && (

<AnswerDisplay

questionData={currentQuestion}

onRating={handleRating}

/>

)}

{!currentQuestion && !isLoading && (

<Card className="border-0 bg-white/60 backdrop-blur-sm">

<CardContent className="p-12 text-center">

<MessageCircleQuestion className="w-16 h-16 text-blue-400 mx-auto mb-4" />

<h3 className="text-xl font-semibold text-gray-700 mb-2">

Ask Your First Question

</h3>

<p className="text-gray-500">

Submit a question above to get started with your AI study assistant

</p>

</CardContent>

</Card>

)}

</div>

<div className="space-y-6">

<Card className="shadow-lg border-0 bg-white/90 backdrop-blur-sm">

<CardHeader className="pb-4">

<CardTitle className="flex items-center gap-2 text-lg font-bold text-gray-900">

<History className="w-5 h-5 text-blue-600" />

Recent Questions

</CardTitle>

</CardHeader>

<CardContent>

<div className="space-y-3">

{recentQuestions.length > 0 ? (

recentQuestions.map((question) => (

<div

key={question.id}

onClick={() => selectRecentQuestion(question)}

className="p-3 rounded-lg border border-gray-200 hover:border-blue-300 hover:bg-blue-50 cursor-pointer transition-all duration-200"

>

<p className="font-medium text-sm text-gray-900 mb-2 line-clamp-2">

{question.question\_text}

</p>

<div className="flex items-center gap-2">

<Badge variant="outline" className="text-xs">

{question.subject}

</Badge>

<span className="text-xs text-gray-500">

{new Date(question.created\_date).toLocaleDateString()}

</span>

</div>

</div>

))

) : (

<p className="text-gray-500 text-sm text-center py-4">

No recent questions yet

</p>

)}

</div>

</CardContent>

</Card>

<Card className="shadow-lg border-0 bg-white/90 backdrop-blur-sm">

<CardHeader className="pb-4">

<CardTitle className="flex items-center gap-2 text-lg font-bold text-gray-900">

<TrendingUp className="w-5 h-5 text-green-600" />

Quick Stats

</CardTitle>

</CardHeader>

<CardContent>

<div className="space-y-4">

<div className="flex items-center justify-between">

<span className="text-gray-600">Questions Asked</span>

<span className="font-semibold text-2xl text-blue-600">

{questions.length}

</span>

</div>

<div className="flex items-center justify-between">

<span className="text-gray-600">Average Rating</span>

<span className="font-semibold text-2xl text-green-600">

4.8/5

</span>

</div>

<div className="flex items-center justify-between">

<span className="text-gray-600">Subjects Covered</span>

<span className="font-semibold text-2xl text-purple-600">

{new Set(recentQuestions.map(q => q.subject)).size}

</span>

</div>

</div>

</CardContent>

</Card>

</div>

</div>

</div>

</div>

);

}

## pages/StudyPlans.js

import React, { useState, useEffect } from "react";

import { StudyPlan, Subject, User, Quiz, FlashcardSet } from "@/entities/all";

import { InvokeLLM } from "@/integrations/Core";

import { Button } from "@/components/ui/button";

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Badge } from "@/components/ui/badge";

import { Progress } from "@/components/ui/progress";

import { Calendar, Clock, Target, Plus, BookOpen, CheckCircle, Zap, HelpCircle } from 'lucide-react';

import { Input } from "@/components/ui/input";

import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";

import { Textarea } from "@/components/ui/textarea";

import { Label } from "@/components/ui/label";

import {

Dialog,

DialogContent,

DialogHeader,

DialogTitle,

DialogTrigger,

} from "@/components/ui/dialog";

export default function StudyPlans() {

const [studyPlans, setStudyPlans] = useState([]);

const [subjects, setSubjects] = useState([]);

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

const [showCreateForm, setShowCreateForm] = useState(false);

const [isCreating, setIsCreating] = useState(false);

const [formData, setFormData] = useState({

title: '',

subject\_id: '',

description: '',

duration\_weeks: 4,

daily\_hours: 1,

difficulty\_level: 'intermediate',

goals: ''

});

const [activeQuiz, setActiveQuiz] = useState(null);

const [quizScore, setQuizScore] = useState(null);

const [selectedAnswers, setSelectedAnswers] = useState({});

useEffect(() => {

loadData();

}, []);

const loadData = async () => {

const [plansData, subjectsData, userData] = await Promise.all([

StudyPlan.list('-created\_date'),

Subject.list(),

User.me()

]);

setStudyPlans(plansData);

setSubjects(subjectsData);

setUser(userData);

};

const handleCreatePlan = async (e) => {

e.preventDefault();

setIsCreating(true);

try {

const selectedSubject = subjects.find(s => s.id === formData.subject\_id);

const prompt = \`Create a detailed, adaptive study plan for a student studying "\${selectedSubject?.name}".

Student Profile:

- Learning Style: \${user?.learning\_style || 'balanced'}

- Learning Pace: \${user?.learning\_pace || 'medium'}

Plan Requirements:

- Title: \${formData.title}

- Duration: \${formData.duration\_weeks} weeks

- Daily study hours: \${formData.daily\_hours}

- Difficulty level: \${formData.difficulty\_level}

- Core Goals: \${formData.goals}

- Description: \${formData.description}

Your Task:

Generate a structured study plan broken down by week or major topic. For each topic, provide:

1. 'name': The topic name.

2. 'estimated\_hours': A reasonable number of hours.

3. 'resources': Suggest a mix of resource types (e.g., "Read Textbook Chapter 5", "Watch Khan Academy video on derivatives", "Complete 10 practice problems") that cater to a \${user?.learning\_style} learner.

4. 'completed': Set to false.

Also, expand the user's core goals into a more detailed list of learning objectives.

Format your entire response as a single JSON object.\`;

const aiResponse = await InvokeLLM({

prompt: prompt,

add\_context\_from\_internet: true,

response\_json\_schema: {

type: "object",

properties: {

topics: {

type: "array",

items: {

type: "object",

properties: {

name: { type: "string" },

estimated\_hours: { type: "number" },

resources: { type: "array", items: { type: "string" } },

completed: { type: "boolean" }

}

}

},

detailed\_goals: { type: "array", items: { type: "string" } }

}

}

});

const newPlan = await StudyPlan.create({

...formData,

goals: formData.goals.split(',').map(g => g.trim()),

topics: aiResponse.topics || [],

status: 'active',

progress\_percentage: 0

});

setStudyPlans(prev => [newPlan, ...prev]);

setShowCreateForm(false);

setFormData({

title: '',

subject\_id: '',

description: '',

duration\_weeks: 4,

daily\_hours: 1,

difficulty\_level: 'intermediate',

goals: ''

});

} catch (error) {

console.error('Error creating study plan:', error);

}

setIsCreating(false);

};

const updateProgress = async (planId, topicIndex) => {

const plan = studyPlans.find(p => p.id === planId);

if (!plan) return;

const updatedTopics = [...plan.topics];

updatedTopics[topicIndex].completed = !updatedTopics[topicIndex].completed;

const completedCount = updatedTopics.filter(t => t.completed).length;

const progressPercentage = Math.round((completedCount / updatedTopics.length) \* 100);

await StudyPlan.update(planId, {

topics: updatedTopics,

progress\_percentage: progressPercentage

});

setStudyPlans(prev => prev.map(p =>

p.id === planId

? { ...p, topics: updatedTopics, progress\_percentage: progressPercentage }

: p

));

};

const generateQuiz = async (plan, topic) => {

const prompt = \`Generate a 5-question multiple-choice quiz about the topic "\${topic.name}" from a study plan for "\${subjects.find(s => s.id === plan.subject\_id)?.name}". The difficulty should be \${plan.difficulty\_level}. For each question, provide the question text, 4 options, the index of the correct answer (0-3), and a brief explanation for the correct answer. Format the response as a JSON object.\`;

const quizData = await InvokeLLM({

prompt,

response\_json\_schema: {

type: "object",

properties: {

questions: {

type: "array",

items: {

type: "object",

properties: {

question\_text: { type: "string" },

options: { type: "array", items: { type: "string" } },

correct\_answer\_index: { type: "number" },

explanation: { type: "string" }

}

}

}

}

}

});

const newQuiz = {

title: \`Quiz: \${topic.name}\`,

subject\_id: plan.subject\_id,

study\_plan\_topic: topic.name,

questions: quizData.questions

};

setActiveQuiz(newQuiz);

setQuizScore(null);

setSelectedAnswers({});

};

const handleQuizSubmit = () => {

let score = 0;

activeQuiz.questions.forEach((q, index) => {

if (selectedAnswers[index] === q.correct\_answer\_index) {

score++;

}

});

const finalScore = (score / activeQuiz.questions.length) \* 100;

setQuizScore(finalScore);

};

const getStatusColor = (status) => {

switch (status) {

case 'active': return 'bg-green-100 text-green-800 border-green-200';

case 'completed': return 'bg-blue-100 text-blue-800 border-blue-200';

case 'paused': return 'bg-yellow-100 text-yellow-800 border-yellow-200';

default: return 'bg-gray-100 text-gray-800 border-gray-200';

}

};

return (

<div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-indigo-50 p-4 md:p-8">

<div className="max-w-7xl mx-auto">

<div className="flex flex-col md:flex-row justify-between items-start md:items-center mb-8 gap-4">

<div>

<h1 className="text-3xl md:text-4xl font-bold text-gray-900 mb-2">

Study Plans

</h1>

<p className="text-xl text-gray-600">

Personalized learning paths tailored to your goals

</p>

</div>

<Button

onClick={() => setShowCreateForm(true)}

className="bg-gradient-to-r from-blue-600 to-indigo-600 hover:from-blue-700 hover:to-indigo-700 shadow-lg"

>

<Plus className="w-4 h-4 mr-2" />

Create Study Plan

</Button>

</div>

{showCreateForm && (

<Card className="mb-8 shadow-xl border-0 bg-white/95 backdrop-blur-sm">

<CardHeader>

<CardTitle className="flex items-center gap-2">

<Target className="w-6 h-6 text-blue-600" />

Create New Study Plan

</CardTitle>

</CardHeader>

<CardContent>

<form onSubmit={handleCreatePlan} className="space-y-6">

<div className="grid grid-cols-1 md:grid-cols-2 gap-6">

<div className="space-y-2">

<Label htmlFor="title">Plan Title</Label>

<Input

id="title"

value={formData.title}

onChange={(e) => setFormData({...formData, title: e.target.value})}

placeholder="e.g., Master Calculus Fundamentals"

required

/>

</div>

<div className="space-y-2">

<Label htmlFor="subject">Subject</Label>

<Select

value={formData.subject\_id}

onValueChange={(value) => setFormData({...formData, subject\_id: value})}

required

>

<SelectTrigger>

<SelectValue placeholder="Select subject" />

</SelectTrigger>

<SelectContent>

{subjects.map((subject) => (

<SelectItem key={subject.id} value={subject.id}>

{subject.name}

</SelectItem>

))}

</SelectContent>

</Select>

</div>

</div>

<div className="space-y-2">

<Label htmlFor="description">Description</Label>

<Textarea

id="description"

value={formData.description}

onChange={(e) => setFormData({...formData, description: e.target.value})}

placeholder="Describe what you want to achieve with this study plan..."

className="h-24"

/>

</div>

<div className="grid grid-cols-1 md:grid-cols-3 gap-6">

<div className="space-y-2">

<Label htmlFor="duration">Duration (weeks)</Label>

<Input

id="duration"

type="number"

min="1"

max="52"

value={formData.duration\_weeks}

onChange={(e) => setFormData({...formData, duration\_weeks: parseInt(e.target.value)})}

/>

</div>

<div className="space-y-2">

<Label htmlFor="daily\_hours">Daily Hours</Label>

<Input

id="daily\_hours"

type="number"

min="0.5"

max="12"

step="0.5"

value={formData.daily\_hours}

onChange={(e) => setFormData({...formData, daily\_hours: parseFloat(e.target.value)})}

/>

</div>

<div className="space-y-2">

<Label htmlFor="difficulty">Difficulty Level</Label>

<Select

value={formData.difficulty\_level}

onValueChange={(value) => setFormData({...formData, difficulty\_level: value})}

>

<SelectTrigger>

<SelectValue />

</SelectTrigger>

<SelectContent>

<SelectItem value="beginner">Beginner</SelectItem>

<SelectItem value="intermediate">Intermediate</SelectItem>

<SelectItem value="advanced">Advanced</SelectItem>

</SelectContent>

</Select>

</div>

</div>

<div className="space-y-2">

<Label htmlFor="goals">Learning Goals (comma-separated)</Label>

<Input

id="goals"

value={formData.goals}

onChange={(e) => setFormData({...formData, goals: e.target.value})}

placeholder="e.g., Master derivatives, Understand limits, Solve complex problems"

/>

</div>

<div className="flex justify-end gap-3">

<Button

type="button"

variant="outline"

onClick={() => setShowCreateForm(false)}

disabled={isCreating}

>

Cancel

</Button>

<Button

type="submit"

disabled={isCreating}

className="bg-gradient-to-r from-blue-600 to-indigo-600 hover:from-blue-700 hover:to-indigo-700"

>

{isCreating ? (

<div className="flex items-center gap-2">

<div className="w-4 h-4 border-2 border-white/30 border-t-white rounded-full animate-spin"></div>

Creating...

</div>

) : (

'Create Plan'

)}

</Button>

</div>

</form>

</CardContent>

</Card>

)}

<div className="grid grid-cols-1 lg:grid-cols-2 gap-8">

{studyPlans.map((plan) => (

<Card key={plan.id} className="shadow-xl border-0 bg-white/95 backdrop-blur-sm">

<CardHeader className="pb-4">

<div className="flex items-start justify-between">

<div className="flex-1">

<CardTitle className="text-xl font-bold text-gray-900 mb-2">

{plan.title}

</CardTitle>

<div className="flex items-center gap-2 flex-wrap mb-3">

<Badge className={\`\${getStatusColor(plan.status)} text-xs border\`}>

{plan.status}

</Badge>

<Badge variant="outline" className="text-xs">

{subjects.find(s => s.id === plan.subject\_id)?.name || 'Unknown Subject'}

</Badge>

<Badge variant="outline" className="text-xs">

{plan.difficulty\_level}

</Badge>

</div>

</div>

<div className="text-right">

<div className="text-2xl font-bold text-blue-600">

{plan.progress\_percentage}%

</div>

<div className="text-xs text-gray-500">Complete</div>

</div>

</div>

<Progress value={plan.progress\_percentage} className="h-2" />

</CardHeader>

<CardContent className="space-y-4">

<p className="text-gray-600 text-sm">{plan.description}</p>

<div className="grid grid-cols-3 gap-4 text-center">

<div className="p-3 bg-blue-50 rounded-lg">

<Calendar className="w-4 h-4 text-blue-600 mx-auto mb-1" />

<div className="text-sm font-semibold text-blue-600">

{plan.duration\_weeks} weeks

</div>

</div>

<div className="p-3 bg-green-50 rounded-lg">

<Clock className="w-4 h-4 text-green-600 mx-auto mb-1" />

<div className="text-sm font-semibold text-green-600">

{plan.daily\_hours}h/day

</div>

</div>

<div className="p-3 bg-purple-50 rounded-lg">

<BookOpen className="w-4 h-4 text-purple-600 mx-auto mb-1" />

<div className="text-sm font-semibold text-purple-600">

{plan.topics?.length || 0} topics

</div>

</div>

</div>

{plan.topics && plan.topics.length > 0 && (

<div className="space-y-2">

<h4 className="font-semibold text-gray-900">Topics</h4>

<div className="space-y-2 max-h-48 overflow-y-auto p-1">

{plan.topics.map((topic, index) => (

<div key={index} className={\`p-2 rounded-lg border \${

topic.completed ? 'bg-green-50 border-green-200' : 'bg-gray-50 border-gray-200'

}\`}>

<div className="flex items-center gap-3">

<button onClick={() => updateProgress(plan.id, index)} className={\`w-5 h-5 rounded-full border-2 flex items-center justify-center flex-shrink-0 \${

topic.completed ? 'bg-green-500 border-green-500 text-white' : 'border-gray-300 hover:border-green-400'

}\`}>{topic.completed && <CheckCircle className="w-3 h-3" />}</button>

<div className="flex-1">

<div className={\`text-sm font-medium \${topic.completed ? 'text-green-800 line-through' : 'text-gray-900'}\`}>{topic.name}</div>

<div className="text-xs text-gray-500">{topic.estimated\_hours}h estimated</div>

</div>

<Dialog onOpenChange={(open) => !open && setActiveQuiz(null)}>

<DialogTrigger asChild>

<Button variant="ghost" size="sm" className="text-xs" onClick={() => generateQuiz(plan, topic)}>

<Zap className="w-3 h-3 mr-1" /> Quiz Me

</Button>

</DialogTrigger>

<DialogContent className="sm:max-w-2xl">

<DialogHeader>

<DialogTitle className="text-2xl">{activeQuiz?.title}</DialogTitle>

</DialogHeader>

{activeQuiz && (

<div className="space-y-6 max-h-[70vh] overflow-y-auto p-2">

{activeQuiz.questions.map((q, qIndex) => (

<div key={qIndex} className="p-4 border rounded-lg">

<p className="font-semibold mb-3">{qIndex + 1}. {q.question\_text}</p>

<div className="space-y-2">

{q.options.map((option, oIndex) => (

<div key={oIndex} className={\`flex items-center gap-2 p-2 rounded-md border cursor-pointer \${selectedAnswers[qIndex] === oIndex ? 'bg-blue-100 border-blue-300' : 'hover:bg-gray-50'}\`}

onClick={() => setSelectedAnswers(prev => ({...prev, [qIndex]: oIndex}))}>

<input type="radio" name={\`q\${qIndex}\`} value={oIndex} checked={selectedAnswers[qIndex] === oIndex} readOnly className="form-radio h-4 w-4 text-blue-600"/>

<label>{option}</label>

</div>

))}

</div>

{quizScore !== null && (

<div className={\`mt-3 p-2 rounded-md text-sm \${selectedAnswers[qIndex] === q.correct\_answer\_index ? 'bg-green-100 text-green-800' : 'bg-red-100 text-red-800'}\`}>

<p className="font-bold">Correct Answer: {q.options[q.correct\_answer\_index]}</p>

<p>{q.explanation}</p>

</div>

)}

</div>

))}

{quizScore === null ? (

<Button onClick={handleQuizSubmit} className="w-full">Submit Quiz</Button>

) : (

<div className="text-center p-4 bg-blue-50 rounded-lg">

<p className="text-xl font-bold">Your Score: {quizScore.toFixed(0)}%</p>

</div>

)}

</div>

)}

</DialogContent>

</Dialog>

</div>

</div>

))}

</div>

</div>

)}

</CardContent>

</Card>

))}

</div>

{studyPlans.length === 0 && !showCreateForm && (

<Card className="border-0 bg-white/60 backdrop-blur-sm">

<CardContent className="p-12 text-center">

<Target className="w-16 h-16 text-blue-400 mx-auto mb-4" />

<h3 className="text-xl font-semibold text-gray-700 mb-2">

Create Your First Study Plan

</h3>

<p className="text-gray-500 mb-6">

Let our AI create a personalized study plan based on your goals and schedule

</p>

<Button

onClick={() => setShowCreateForm(true)}

className="bg-gradient-to-r from-blue-600 to-indigo-600 hover:from-blue-700 hover:to-indigo-700"

>

<Plus className="w-4 h-4 mr-2" />

Get Started

</Button>

</CardContent>

</Card>

)}

</div>

</div>

);

}

## pages/Subjects.js

import React, { useState, useEffect } from "react";

import { Subject } from "@/entities/all";

import { Button } from "@/components/ui/button";

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Badge } from "@/components/ui/badge";

import { Input } from "@/components/ui/input";

import { Label } from "@/components/ui/label";

import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";

import { Textarea } from "@/components/ui/textarea";

import { Plus, BookOpen, Clock, TrendingUp, Edit, Trash2 } from 'lucide-react';

export default function Subjects() {

const [subjects, setSubjects] = useState([]);

const [showCreateForm, setShowCreateForm] = useState(false);

const [editingSubject, setEditingSubject] = useState(null);

const [formData, setFormData] = useState({

name: '',

description: '',

difficulty\_level: 'beginner',

weekly\_hours: 5,

priority: 'medium',

color: '#3b82f6'

});

const colors = [

{ value: '#3b82f6', name: 'Blue' },

{ value: '#10b981', name: 'Green' },

{ value: '#f59e0b', name: 'Yellow' },

{ value: '#ef4444', name: 'Red' },

{ value: '#8b5cf6', name: 'Purple' },

{ value: '#f97316', name: 'Orange' },

{ value: '#06b6d4', name: 'Cyan' },

{ value: '#84cc16', name: 'Lime' }

];

useEffect(() => {

loadSubjects();

}, []);

const loadSubjects = async () => {

const data = await Subject.list('-updated\_date');

setSubjects(data);

};

const handleSubmit = async (e) => {

e.preventDefault();

if (editingSubject) {

await Subject.update(editingSubject.id, formData);

} else {

await Subject.create(formData);

}

setShowCreateForm(false);

setEditingSubject(null);

setFormData({

name: '',

description: '',

difficulty\_level: 'beginner',

weekly\_hours: 5,

priority: 'medium',

color: '#3b82f6'

});

loadSubjects();

};

const handleEdit = (subject) => {

setEditingSubject(subject);

setFormData({

name: subject.name,

description: subject.description || '',

difficulty\_level: subject.difficulty\_level,

weekly\_hours: subject.weekly\_hours,

priority: subject.priority,

color: subject.color

});

setShowCreateForm(true);

};

const handleDelete = async (subjectId) => {

if (window.confirm('Are you sure you want to delete this subject?')) {

await Subject.delete(subjectId);

loadSubjects();

}

};

const getPriorityColor = (priority) => {

switch (priority) {

case 'high': return 'bg-red-100 text-red-800 border-red-200';

case 'medium': return 'bg-yellow-100 text-yellow-800 border-yellow-200';

case 'low': return 'bg-green-100 text-green-800 border-green-200';

default: return 'bg-gray-100 text-gray-800 border-gray-200';

}

};

const getDifficultyColor = (difficulty) => {

switch (difficulty) {

case 'beginner': return 'bg-blue-100 text-blue-800 border-blue-200';

case 'intermediate': return 'bg-purple-100 text-purple-800 border-purple-200';

case 'advanced': return 'bg-orange-100 text-orange-800 border-orange-200';

default: return 'bg-gray-100 text-gray-800 border-gray-200';

}

};

return (

<div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-indigo-50 p-4 md:p-8">

<div className="max-w-7xl mx-auto">

<div className="flex flex-col md:flex-row justify-between items-start md:items-center mb-8 gap-4">

<div>

<h1 className="text-3xl md:text-4xl font-bold text-gray-900 mb-2">

My Subjects

</h1>

<p className="text-xl text-gray-600">

Manage your academic subjects and study preferences

</p>

</div>

<Button

onClick={() => setShowCreateForm(true)}

className="bg-gradient-to-r from-blue-600 to-indigo-600 hover:from-blue-700 hover:to-indigo-700 shadow-lg"

>

<Plus className="w-4 h-4 mr-2" />

Add Subject

</Button>

</div>

{showCreateForm && (

<Card className="mb-8 shadow-xl border-0 bg-white/95 backdrop-blur-sm">

<CardHeader>

<CardTitle className="flex items-center gap-2">

<BookOpen className="w-6 h-6 text-blue-600" />

{editingSubject ? 'Edit Subject' : 'Add New Subject'}

</CardTitle>

</CardHeader>

<CardContent>

<form onSubmit={handleSubmit} className="space-y-6">

<div className="grid grid-cols-1 md:grid-cols-2 gap-6">

<div className="space-y-2">

<Label htmlFor="name">Subject Name</Label>

<Input

id="name"

value={formData.name}

onChange={(e) => setFormData({...formData, name: e.target.value})}

placeholder="e.g., Mathematics, Physics, History"

required

/>

</div>

<div className="space-y-2">

<Label htmlFor="difficulty">Difficulty Level</Label>

<Select

value={formData.difficulty\_level}

onValueChange={(value) => setFormData({...formData, difficulty\_level: value})}

>

<SelectTrigger>

<SelectValue />

</SelectTrigger>

<SelectContent>

<SelectItem value="beginner">Beginner</SelectItem>

<SelectItem value="intermediate">Intermediate</SelectItem>

<SelectItem value="advanced">Advanced</SelectItem>

</SelectContent>

</Select>

</div>

</div>

<div className="space-y-2">

<Label htmlFor="description">Description</Label>

<Textarea

id="description"

value={formData.description}

onChange={(e) => setFormData({...formData, description: e.target.value})}

placeholder="Brief description of what you're studying in this subject..."

className="h-24"

/>

</div>

<div className="grid grid-cols-1 md:grid-cols-3 gap-6">

<div className="space-y-2">

<Label htmlFor="weekly\_hours">Weekly Hours</Label>

<Input

id="weekly\_hours"

type="number"

min="1"

max="40"

value={formData.weekly\_hours}

onChange={(e) => setFormData({...formData, weekly\_hours: parseInt(e.target.value)})}

/>

</div>

<div className="space-y-2">

<Label htmlFor="priority">Priority</Label>

<Select

value={formData.priority}

onValueChange={(value) => setFormData({...formData, priority: value})}

>

<SelectTrigger>

<SelectValue />

</SelectTrigger>

<SelectContent>

<SelectItem value="low">Low Priority</SelectItem>

<SelectItem value="medium">Medium Priority</SelectItem>

<SelectItem value="high">High Priority</SelectItem>

</SelectContent>

</Select>

</div>

<div className="space-y-2">

<Label htmlFor="color">Color</Label>

<Select

value={formData.color}

onValueChange={(value) => setFormData({...formData, color: value})}

>

<SelectTrigger>

<SelectValue />

</SelectTrigger>

<SelectContent>

{colors.map((color) => (

<SelectItem key={color.value} value={color.value}>

<div className="flex items-center gap-2">

<div

className="w-4 h-4 rounded-full border border-gray-300"

style={{ backgroundColor: color.value }}

></div>

{color.name}

</div>

</SelectItem>

))}

</SelectContent>

</Select>

</div>

</div>

<div className="flex justify-end gap-3">

<Button

type="button"

variant="outline"

onClick={() => {

setShowCreateForm(false);

setEditingSubject(null);

setFormData({

name: '',

description: '',

difficulty\_level: 'beginner',

weekly\_hours: 5,

priority: 'medium',

color: '#3b82f6'

});

}}

>

Cancel

</Button>

<Button

type="submit"

className="bg-gradient-to-r from-blue-600 to-indigo-600 hover:from-blue-700 hover:to-indigo-700"

>

{editingSubject ? 'Update Subject' : 'Add Subject'}

</Button>

</div>

</form>

</CardContent>

</Card>

)}

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">

{subjects.map((subject) => (

<Card key={subject.id} className="shadow-xl border-0 bg-white/95 backdrop-blur-sm hover:shadow-2xl transition-all duration-200">

<CardHeader className="pb-4">

<div className="flex items-start justify-between">

<div className="flex items-center gap-3">

<div

className="w-4 h-16 rounded-full"

style={{ backgroundColor: subject.color }}

></div>

<div className="flex-1">

<CardTitle className="text-xl font-bold text-gray-900 mb-2">

{subject.name}

</CardTitle>

<div className="flex items-center gap-2 flex-wrap">

<Badge className={\`\${getDifficultyColor(subject.difficulty\_level)} text-xs border\`}>

{subject.difficulty\_level}

</Badge>

<Badge className={\`\${getPriorityColor(subject.priority)} text-xs border\`}>

{subject.priority} priority

</Badge>

</div>

</div>

</div>

<div className="flex items-center gap-1">

<Button

variant="ghost"

size="icon"

onClick={() => handleEdit(subject)}

className="text-gray-400 hover:text-blue-600"

>

<Edit className="w-4 h-4" />

</Button>

<Button

variant="ghost"

size="icon"

onClick={() => handleDelete(subject.id)}

className="text-gray-400 hover:text-red-600"

>

<Trash2 className="w-4 h-4" />

</Button>

</div>

</div>

</CardHeader>

<CardContent className="space-y-4">

{subject.description && (

<p className="text-gray-600 text-sm">{subject.description}</p>

)}

<div className="grid grid-cols-2 gap-4">

<div className="text-center p-3 bg-blue-50 rounded-lg">

<Clock className="w-5 h-5 text-blue-600 mx-auto mb-1" />

<div className="text-sm font-semibold text-blue-600">

{subject.weekly\_hours}h/week

</div>

<div className="text-xs text-gray-500">Study Time</div>

</div>

<div className="text-center p-3 bg-green-50 rounded-lg">

<TrendingUp className="w-5 h-5 text-green-600 mx-auto mb-1" />

<div className="text-sm font-semibold text-green-600">

85%

</div>

<div className="text-xs text-gray-500">Progress</div>

</div>

</div>

<div className="pt-2">

<div className="flex items-center justify-between text-xs text-gray-500 mb-1">

<span>Weekly Progress</span>

<span>12/15 hours</span>

</div>

<div className="w-full bg-gray-200 rounded-full h-2">

<div

className="h-2 rounded-full transition-all duration-300"

style={{

backgroundColor: subject.color,

width: '80%'

}}

></div>

</div>

</div>

</CardContent>

</Card>

))}

</div>

{subjects.length === 0 && !showCreateForm && (

<Card className="border-0 bg-white/60 backdrop-blur-sm">

<CardContent className="p-12 text-center">

<BookOpen className="w-16 h-16 text-blue-400 mx-auto mb-4" />

<h3 className="text-xl font-semibold text-gray-700 mb-2">

Add Your First Subject

</h3>

<p className="text-gray-500 mb-6">

Start by adding the subjects you want to study and track your progress

</p>

<Button

onClick={() => setShowCreateForm(true)}

className="bg-gradient-to-r from-blue-600 to-indigo-600 hover:from-blue-700 hover:to-indigo-700"

>

<Plus className="w-4 h-4 mr-2" />

Add Subject

</Button>

</CardContent>

</Card>

)}

</div>

</div>

);

}

## pages/Progress.js

import React, { useState, useEffect } from "react";

import { StudySession, Subject, StudyPlan, Question, Achievement } from "@/entities/all";

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Badge } from "@/components/ui/badge";

import { Progress } from "@/components/ui/progress";

import { TrendingUp, Calendar, Clock, Target, Award, BookOpen, Brain } from 'lucide-react';

import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Legend, ResponsiveContainer, LineChart, Line } from 'recharts';

import DynamicIcon from '@/components/ui/DynamicIcon';

export default function ProgressPage() {

const [studySessions, setStudySessions] = useState([]);

const [subjects, setSubjects] = useState([]);

const [studyPlans, setStudyPlans] = useState([]);

const [questions, setQuestions] = useState([]);

const [achievements, setAchievements] = useState([]);

const [stats, setStats] = useState({

totalHours: 0,

weeklyGoal: 20,

streak: 7,

completedPlans: 0

});

useEffect(() => {

loadProgressData();

}, []);

const loadProgressData = async () => {

const [sessionsData, subjectsData, plansData, questionsData, achievementsData] = await Promise.all([

StudySession.list('-created\_date', 50),

Subject.list(),

StudyPlan.list(),

Question.list('-created\_date', 100),

Achievement.list('-unlocked\_date')

]);

setStudySessions(sessionsData);

setSubjects(subjectsData);

setStudyPlans(plansData);

setQuestions(questionsData);

setAchievements(achievementsData);

const totalMinutes = sessionsData.reduce((sum, session) => sum + session.duration\_minutes, 0);

const totalHours = Math.round((totalMinutes / 60) \* 10) / 10;

const completedPlans = plansData.filter(plan => plan.status === 'completed').length;

setStats({

totalHours,

weeklyGoal: 20,

streak: 7,

completedPlans

});

};

const getWeeklyStudyData = () => {

const data = Array(7).fill(0).map((\_, i) => {

const d = new Date();

d.setDate(d.getDate() - i);

return { name: d.toLocaleDateString('en-US', { weekday: 'short' }), hours: 0 };

}).reverse();

studySessions.forEach(session => {

const sessionDate = new Date(session.created\_date);

const diffDays = Math.floor((new Date() - sessionDate) / (1000 \* 60 \* 60 \* 24));

if (diffDays < 7) {

const dayIndex = 6 - diffDays;

data[dayIndex].hours += session.duration\_minutes / 60;

}

});

return data.map(d => ({ ...d, hours: Math.round(d.hours \* 10) / 10 }));

};

const getWeeklyProgress = () => {

const now = new Date();

const weekStart = new Date(now.setDate(now.getDate() - now.getDay()));

const weekSessions = studySessions.filter(session =>

new Date(session.created\_date) >= weekStart

);

const weeklyHours = weekSessions.reduce((sum, session) =>

sum + session.duration\_minutes, 0) / 60;

return Math.round((weeklyHours / stats.weeklyGoal) \* 100);

};

const getSubjectStats = () => {

return subjects.map(subject => {

const subjectSessions = studySessions.filter(session =>

session.subject\_id === subject.id

);

const totalMinutes = subjectSessions.reduce((sum, session) =>

sum + session.duration\_minutes, 0);

const subjectPlans = studyPlans.filter(plan =>

plan.subject\_id === subject.id

);

const avgProgress = subjectPlans.length > 0

? subjectPlans.reduce((sum, plan) => sum + plan.progress\_percentage, 0) / subjectPlans.length

: 0;

return {

...subject,

totalHours: Math.round((totalMinutes / 60) \* 10) / 10,

sessionsCount: subjectSessions.length,

avgProgress: Math.round(avgProgress)

};

}).sort((a, b) => b.totalHours - a.totalHours);

};

const getAchievements = () => {

return achievements;

};

const weeklyStudyData = getWeeklyStudyData();

const weeklyProgress = getWeeklyProgress();

const subjectStats = getSubjectStats();

const unlockedAchievements = getAchievements();

return (

<div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-indigo-50 p-4 md:p-8">

<div className="max-w-7xl mx-auto">

<div className="mb-8">

<h1 className="text-3xl md:text-4xl font-bold text-gray-900 mb-2">

Progress & Insights

</h1>

<p className="text-xl text-gray-600">

Analyze your learning patterns, track progress, and celebrate achievements.

</p>

</div>

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6 mb-8">

<Card className="shadow-lg border-0 bg-white/90 backdrop-blur-sm">

<CardContent className="p-6">

<div className="flex items-center gap-3">

<div className="p-3 bg-blue-100 rounded-xl">

<Clock className="w-6 h-6 text-blue-600" />

</div>

<div>

<p className="text-sm text-gray-600">Total Study Time</p>

<p className="text-2xl font-bold text-gray-900">{stats.totalHours}h</p>

</div>

</div>

</CardContent>

</Card>

<Card className="shadow-lg border-0 bg-white/90 backdrop-blur-sm">

<CardContent className="p-6">

<div className="flex items-center gap-3">

<div className="p-3 bg-green-100 rounded-xl">

<TrendingUp className="w-6 h-6 text-green-600" />

</div>

<div>

<p className="text-sm text-gray-600">Study Streak</p>

<p className="text-2xl font-bold text-gray-900">{stats.streak} days</p>

</div>

</div>

</CardContent>

</Card>

<Card className="shadow-lg border-0 bg-white/90 backdrop-blur-sm">

<CardContent className="p-6">

<div className="flex items-center gap-3">

<div className="p-3 bg-purple-100 rounded-xl">

<Target className="w-6 h-6 text-purple-600" />

</div>

<div>

<p className="text-sm text-gray-600">Completed Plans</p>

<p className="text-2xl font-bold text-gray-900">{stats.completedPlans}</p>

</div>

</div>

</CardContent>

</Card>

<Card className="shadow-lg border-0 bg-white/90 backdrop-blur-sm">

<CardContent className="p-6">

<div className="flex items-center gap-3">

<div className="p-3 bg-orange-100 rounded-xl">

<Brain className="w-6 h-6 text-orange-600" />

</div>

<div>

<p className="text-sm text-gray-600">Questions Asked</p>

<p className="text-2xl font-bold text-gray-900">{questions.length}</p>

</div>

</div>

</CardContent>

</Card>

</div>

<div className="grid grid-cols-1 lg:grid-cols-3 gap-8 mb-8">

<Card className="lg:col-span-2 shadow-xl border-0 bg-white/95 backdrop-blur-sm">

<CardHeader>

<CardTitle className="flex items-center gap-2">

<Calendar className="w-6 h-6 text-blue-600" />

Weekly Study Hours

</CardTitle>

</CardHeader>

<CardContent>

<ResponsiveContainer width="100%" height={300}>

<BarChart data={weeklyStudyData}>

<CartesianGrid strokeDasharray="3 3" />

<XAxis dataKey="name" />

<YAxis />

<Tooltip />

<Legend />

<Bar dataKey="hours" fill="#3b82f6" name="Study Hours" />

</BarChart>

</ResponsiveContainer>

</CardContent>

</Card>

<Card className="shadow-xl border-0 bg-white/95 backdrop-blur-sm">

<CardHeader>

<CardTitle className="flex items-center gap-2">

<Award className="w-6 h-6 text-yellow-600" />

Unlocked Achievements

</CardTitle>

</CardHeader>

<CardContent>

<div className="space-y-3">

{unlockedAchievements.map((achievement, index) => (

<div key={index} className="flex items-center gap-3">

<div className={\`p-2 rounded-full \${achievement.color}\`}>

<DynamicIcon name={achievement.icon} className="w-5 h-5" />

</div>

<div>

<h3 className="font-semibold text-gray-900 text-sm">{achievement.title}</h3>

<p className="text-xs text-gray-600">{achievement.description}</p>

</div>

</div>

))}

{unlockedAchievements.length === 0 && (

<div className="text-center py-8">

<Award className="w-12 h-12 text-gray-300 mx-auto mb-3" />

<p className="text-gray-500">Keep studying to unlock achievements!</p>

</div>

)}

</div>

</CardContent>

</Card>

</div>

<Card className="shadow-xl border-0 bg-white/95 backdrop-blur-sm">

<CardHeader>

<CardTitle className="flex items-center gap-2">

<BookOpen className="w-6 h-6 text-blue-600" />

Subject Progress

</CardTitle>

</CardHeader>

<CardContent>

<div className="space-y-4">

{subjectStats.map((subject) => (

<div key={subject.id} className="p-4 rounded-xl bg-gradient-to-r from-gray-50 to-gray-100">

<div className="flex items-center justify-between mb-3">

<div className="flex items-center gap-3">

<div

className="w-4 h-12 rounded-full"

style={{ backgroundColor: subject.color }}

></div>

<div>

<h3 className="font-semibold text-gray-900">{subject.name}</h3>

<p className="text-sm text-gray-500">

{subject.totalHours}h • {subject.sessionsCount} sessions

</p>

</div>

</div>

<div className="text-right">

<div className="text-2xl font-bold text-blue-600">

{subject.avgProgress}%

</div>

<div className="text-xs text-gray-500">Average Progress</div>

</div>

</div>

<Progress value={subject.avgProgress} className="h-2" />

</div>

))}

{subjectStats.length === 0 && (

<div className="text-center py-8">

<BookOpen className="w-12 h-12 text-gray-300 mx-auto mb-3" />

<p className="text-gray-500">Add subjects to track your progress</p>

</div>

)}

</div>

</CardContent>

</Card>

</div>

</div>

);

}

## pages/Materials.js

import React, { useState, useEffect } from "react";

import { LearningMaterial, Subject } from "@/entities/all";

import { UploadFile, InvokeLLM } from "@/integrations/Core";

import { Button } from "@/components/ui/button";

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Input } from "@/components/ui/input";

import { Label } from "@/components/ui/label";

import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";

import { Plus, FileText, Upload, Trash2, BrainCircuit } from 'lucide-react';

import { Badge } from "@/components/ui/badge";

export default function Materials() {

const [materials, setMaterials] = useState([]);

const [subjects, setSubjects] = useState([]);

const [showUploadForm, setShowUploadForm] = useState(false);

const [isUploading, setIsUploading] = useState(false);

const [file, setFile] = useState(null);

const [title, setTitle] = useState('');

const [subjectId, setSubjectId] = useState('');

useEffect(() => {

loadData();

}, []);

const loadData = async () => {

const [materialsData, subjectsData] = await Promise.all([

LearningMaterial.list('-created\_date'),

Subject.list()

]);

setMaterials(materialsData);

setSubjects(subjectsData);

};

const handleFileChange = (e) => {

if (e.target.files.length > 0) {

setFile(e.target.files[0]);

}

};

const handleUpload = async (e) => {

e.preventDefault();

if (!file || !title || !subjectId) {

alert("Please fill all fields and select a file.");

return;

}

setIsUploading(true);

try {

const { file\_url } = await UploadFile({ file });

const prompt = \`Summarize the following document and extract up to 10 relevant keywords. The document is titled "\${title}".\`;

const aiResponse = await InvokeLLM({

prompt: prompt,

file\_urls: [file\_url],

response\_json\_schema: {

type: "object",

properties: {

summary: { type: "string" },

keywords: { type: "array", items: { type: "string" } }

}

}

});

await LearningMaterial.create({

title,

file\_url,

file\_type: file.type.includes('pdf') ? 'pdf' : 'text',

subject\_id: subjectId,

summary: aiResponse.summary,

keywords: aiResponse.keywords

});

setShowUploadForm(false);

setFile(null);

setTitle('');

setSubjectId('');

await loadData();

} catch (error) {

console.error("Error uploading material:", error);

alert("Failed to upload material.");

} finally {

setIsUploading(false);

}

};

const handleDelete = async (materialId) => {

if (window.confirm("Are you sure you want to delete this material?")) {

await LearningMaterial.delete(materialId);

await loadData();

}

};

return (

<div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-indigo-50 p-4 md:p-8">

<div className="max-w-7xl mx-auto">

<div className="flex flex-col md:flex-row justify-between items-start md:items-center mb-8 gap-4">

<div>

<h1 className="text-3xl md:text-4xl font-bold text-gray-900 mb-2">My Materials</h1>

<p className="text-xl text-gray-600">Upload and manage your course notes, textbooks, and lectures.</p>

</div>

<Button

onClick={() => setShowUploadForm(true)}

className="bg-gradient-to-r from-blue-600 to-indigo-600 hover:from-blue-700 hover:to-indigo-700 shadow-lg"

>

<Upload className="w-4 h-4 mr-2" />

Upload Material

</Button>

</div>

{showUploadForm && (

<Card className="mb-8 shadow-xl border-0 bg-white/95 backdrop-blur-sm">

<CardHeader>

<CardTitle className="flex items-center gap-2">

<FileText className="w-6 h-6 text-blue-600" /> Upload New Material

</CardTitle>

</CardHeader>

<CardContent>

<form onSubmit={handleUpload} className="space-y-6">

<div className="grid grid-cols-1 md:grid-cols-2 gap-6">

<div className="space-y-2">

<Label htmlFor="title">Material Title</Label>

<Input id="title" value={title} onChange={(e) => setTitle(e.target.value)} placeholder="e.g., Physics Chapter 5 Notes" required />

</div>

<div className="space-y-2">

<Label htmlFor="subject">Subject</Label>

<Select value={subjectId} onValueChange={setSubjectId} required>

<SelectTrigger><SelectValue placeholder="Select subject" /></SelectTrigger>

<SelectContent>

{subjects.map(s => <SelectItem key={s.id} value={s.id}>{s.name}</SelectItem>)}

</SelectContent>

</Select>

</div>

</div>

<div className="space-y-2">

<Label htmlFor="file">File (PDF, TXT, etc.)</Label>

<Input id="file" type="file" onChange={handleFileChange} required />

</div>

<div className="flex justify-end gap-3">

<Button type="button" variant="outline" onClick={() => setShowUploadForm(false)} disabled={isUploading}>Cancel</Button>

<Button type="submit" disabled={isUploading}>

{isUploading ? "Uploading..." : "Upload and Process"}

</Button>

</div>

</form>

</CardContent>

</Card>

)}

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">

{materials.map((material) => (

<Card key={material.id} className="shadow-lg border-0 bg-white/95 backdrop-blur-sm flex flex-col">

<CardHeader className="flex flex-row items-start justify-between">

<div className="flex items-center gap-3">

<div className="p-3 bg-blue-100 rounded-lg">

<FileText className="w-5 h-5 text-blue-600" />

</div>

<div className="flex-1">

<CardTitle className="text-lg font-bold text-gray-900 leading-tight">{material.title}</CardTitle>

<p className="text-sm text-gray-500">{subjects.find(s => s.id === material.subject\_id)?.name}</p>

</div>

</div>

<Button variant="ghost" size="icon" className="text-gray-400 hover:text-red-600" onClick={() => handleDelete(material.id)}>

<Trash2 className="w-4 h-4" />

</Button>

</CardHeader>

<CardContent className="flex-grow">

<p className="text-sm text-gray-600 mb-4 line-clamp-3">{material.summary || "No summary available."}</p>

<div className="flex flex-wrap gap-2">

{(material.keywords || []).map(keyword => <Badge key={keyword} variant="secondary">{keyword}</Badge>)}

</div>

</CardContent>

</Card>

))}

</div>

{materials.length === 0 && !showUploadForm && (

<Card className="border-0 bg-white/60 backdrop-blur-sm">

<CardContent className="p-12 text-center">

<FileText className="w-16 h-16 text-blue-400 mx-auto mb-4" />

<h3 className="text-xl font-semibold text-gray-700 mb-2">Upload Your First Study Material</h3>

<p className="text-gray-500">Add your notes and textbooks to get context-aware help from the AI.</p>

</CardContent>

</Card>

)}

</div>

</div>

);

}

# ------------------- COMPONENTS -------------------

## components/dashboard/WelcomeHeader.jsx

import React from 'react';

import { Calendar, Clock, Target } from 'lucide-react';

import { Card, CardContent } from "@/components/ui/card";

export default function WelcomeHeader({ userName = "Student" }) {

const currentDate = new Date();

const timeOfDay = currentDate.getHours() < 12 ? 'Morning' : currentDate.getHours() < 18 ? 'Afternoon' : 'Evening';

return (

<div className="relative overflow-hidden bg-gradient-to-r from-blue-600 via-blue-700 to-indigo-800 rounded-2xl p-8 mb-8 text-white">

<div className="absolute top-0 right-0 w-64 h-64 transform translate-x-32 -translate-y-32">

<div className="w-full h-full bg-white/10 rounded-full blur-3xl"></div>

</div>

<div className="absolute bottom-0 left-0 w-48 h-48 transform -translate-x-24 translate-y-24">

<div className="w-full h-full bg-blue-400/20 rounded-full blur-2xl"></div>

</div>

<div className="relative z-10">

<div className="flex flex-col md:flex-row justify-between items-start md:items-center mb-6">

<div>

<h1 className="text-3xl md:text-4xl font-bold mb-2">

Good {timeOfDay}, {userName}! 👋

</h1>

<p className="text-blue-100 text-lg">

Ready to continue your learning journey today?

</p>

</div>

<div className="mt-4 md:mt-0 text-right">

<div className="text-blue-100 text-sm mb-1">Today's Date</div>

<div className="text-xl font-semibold">

{currentDate.toLocaleDateString('en-US', {

weekday: 'long',

month: 'short',

day: 'numeric'

})}

</div>

</div>

</div>

<div className="grid grid-cols-1 md:grid-cols-3 gap-4">

<Card className="bg-white/10 backdrop-blur-sm border-white/20">

<CardContent className="flex items-center gap-3 p-4">

<Calendar className="w-8 h-8 text-blue-200" />

<div>

<div className="text-sm text-blue-100">Study Streak</div>

<div className="text-2xl font-bold">7 Days</div>

</div>

</CardContent>

</Card>

<Card className="bg-white/10 backdrop-blur-sm border-white/20">

<CardContent className="flex items-center gap-3 p-4">

<Clock className="w-8 h-8 text-green-200" />

<div>

<div className="text-sm text-blue-100">Hours Today</div>

<div className="text-2xl font-bold">2.5h</div>

</div>

</CardContent>

</Card>

<Card className="bg-white/10 backdrop-blur-sm border-white/20">

<CardContent className="flex items-center gap-3 p-4">

<Target className="w-8 h-8 text-orange-200" />

<div>

<div className="text-sm text-blue-100">Weekly Goal</div>

<div className="text-2xl font-bold">68%</div>

</div>

</CardContent>

</Card>

</div>

</div>

</div>

);

}

## components/dashboard/QuickActions.jsx

import React from 'react';

import { Button } from "@/components/ui/button";

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Link } from "react-router-dom";

import { createPageUrl } from "@/utils";

import { MessageCircleQuestion, Calendar, BookOpen, TrendingUp, Plus, Brain } from 'lucide-react';

export default function QuickActions() {

const actions = [

{

title: "Ask AI Assistant",

description: "Get instant answers to your academic questions",

icon: MessageCircleQuestion,

link: createPageUrl("QAAssistant"),

color: "bg-blue-500 hover:bg-blue-600",

textColor: "text-blue-600"

},

{

title: "Create Study Plan",

description: "Generate a personalized study schedule",

icon: Calendar,

link: createPageUrl("StudyPlans"),

color: "bg-green-500 hover:bg-green-600",

textColor: "text-green-600"

},

{

title: "Add Subject",

description: "Start studying a new subject",

icon: BookOpen,

link: createPageUrl("Subjects"),

color: "bg-purple-500 hover:bg-purple-600",

textColor: "text-purple-600"

},

{

title: "View Progress",

description: "Track your learning achievements",

icon: TrendingUp,

link: createPageUrl("Progress"),

color: "bg-orange-500 hover:bg-orange-600",

textColor: "text-orange-600"

}

];

return (

<Card className="shadow-lg border-0 bg-white/80 backdrop-blur-sm">

<CardHeader className="pb-4">

<CardTitle className="flex items-center gap-2 text-xl font-bold text-gray-900">

<Brain className="w-6 h-6 text-blue-600" />

Quick Actions

</CardTitle>

</CardHeader>

<CardContent>

<div className="grid grid-cols-1 md:grid-cols-2 gap-4">

{actions.map((action, index) => (

<Link key={index} to={action.link} className="group">

<div className="p-4 rounded-xl border border-gray-200 hover:border-gray-300 transition-all duration-200 hover:shadow-md bg-white/50 hover:bg-white/80">

<div className="flex items-start gap-3">

<div className={\`p-2 rounded-lg \${action.color} transition-colors duration-200\`}>

<action.icon className="w-5 h-5 text-white" />

</div>

<div className="flex-1">

<h3 className={\`font-semibold \${action.textColor} group-hover:text-gray-900 transition-colors\`}>

{action.title}

</h3>

<p className="text-sm text-gray-600 mt-1">

{action.description}

</p>

</div>

</div>

</div>

</Link>

))}

</div>

</CardContent>

</Card>

);

}

## components/dashboard/UpcomingStudySessions.jsx

import React from 'react';

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Badge } from "@/components/ui/badge";

import { Clock, BookOpen, ArrowRight } from 'lucide-react';

import { Button } from "@/components/ui/button";

export default function UpcomingStudySessions() {

const upcomingSessions = [

{

id: 1,

subject: "Mathematics",

topic: "Calculus - Derivatives",

time: "2:00 PM",

duration: "60 min",

difficulty: "intermediate",

color: "bg-blue-500"

},

{

id: 2,

subject: "Physics",

topic: "Quantum Mechanics",

time: "4:30 PM",

duration: "90 min",

difficulty: "advanced",

color: "bg-purple-500"

},

{

id: 3,

subject: "Chemistry",

topic: "Organic Reactions",

time: "7:00 PM",

duration: "45 min",

difficulty: "beginner",

color: "bg-green-500"

}

];

const getDifficultyColor = (difficulty) => {

switch (difficulty) {

case 'beginner': return 'bg-green-100 text-green-800 border-green-200';

case 'intermediate': return 'bg-yellow-100 text-yellow-800 border-yellow-200';

case 'advanced': return 'bg-red-100 text-red-800 border-red-200';

default: return 'bg-gray-100 text-gray-800 border-gray-200';

}

};

return (

<Card className="shadow-lg border-0 bg-white/80 backdrop-blur-sm">

<CardHeader className="pb-4">

<CardTitle className="flex items-center gap-2 text-xl font-bold text-gray-900">

<Clock className="w-6 h-6 text-blue-600" />

Today's Schedule

</CardTitle>

</CardHeader>

<CardContent>

<div className="space-y-4">

{upcomingSessions.map((session) => (

<div key={session.id} className="flex items-center gap-4 p-4 rounded-xl bg-white/60 border border-gray-200 hover:shadow-md transition-all duration-200">

<div className={\`w-1 h-16 \${session.color} rounded-full\`}></div>

<div className="flex-1">

<div className="flex items-center gap-2 mb-2">

<BookOpen className="w-4 h-4 text-gray-500" />

<span className="font-semibold text-gray-900">{session.subject}</span>

<Badge className={\`\${getDifficultyColor(session.difficulty)} text-xs border\`}>

{session.difficulty}

</Badge>

</div>

<p className="text-gray-600 text-sm mb-2">{session.topic}</p>

<div className="flex items-center gap-4 text-xs text-gray-500">

<span className="flex items-center gap-1">

<Clock className="w-3 h-3" />

{session.time}

</span>

<span>{session.duration}</span>

</div>

</div>

<Button variant="ghost" size="icon" className="text-blue-600 hover:text-blue-700 hover:bg-blue-50">

<ArrowRight className="w-4 h-4" />

</Button>

</div>

))}

</div>

</CardContent>

</Card>

);

}

## components/dashboard/RecentActivity.jsx

import React from 'react';

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Badge } from "@/components/ui/badge";

import { MessageCircleQuestion, BookOpen, CheckCircle, Clock } from 'lucide-react';

export default function RecentActivity() {

const activities = [

{

id: 1,

type: "question",

content: "Asked about quadratic equations",

subject: "Mathematics",

time: "2 hours ago",

icon: MessageCircleQuestion,

color: "text-blue-600"

},

{

id: 2,

type: "study",

content: "Completed Linear Algebra chapter",

subject: "Mathematics",

time: "4 hours ago",

icon: CheckCircle,

color: "text-green-600"

},

{

id: 3,

type: "session",

content: "45-minute study session",

subject: "Physics",

time: "Yesterday",

icon: Clock,

color: "text-purple-600"

},

{

id: 4,

type: "subject",

content: "Added new subject: Chemistry",

subject: "Chemistry",

time: "2 days ago",

icon: BookOpen,

color: "text-orange-600"

}

];

return (

<Card className="shadow-lg border-0 bg-white/80 backdrop-blur-sm">

<CardHeader className="pb-4">

<CardTitle className="flex items-center gap-2 text-xl font-bold text-gray-900">

<Clock className="w-6 h-6 text-blue-600" />

Recent Activity

</CardTitle>

</CardHeader>

<CardContent>

<div className="space-y-4">

{activities.map((activity) => (

<div key={activity.id} className="flex items-start gap-3 p-3 rounded-lg hover:bg-white/60 transition-colors duration-200">

<div className={\`p-2 rounded-lg bg-gray-50 \${activity.color}\`}>

<activity.icon className="w-4 h-4" />

</div>

<div className="flex-1">

<p className="text-gray-900 font-medium text-sm">

{activity.content}

</p>

<div className="flex items-center gap-2 mt-1">

<Badge variant="outline" className="text-xs">

{activity.subject}

</Badge>

<span className="text-xs text-gray-500">{activity.time}</span>

</div>

</div>

</div>

))}

</div>

</CardContent>

</Card>

);

}

## components/qa/QuestionForm.jsx

import React, { useState } from 'react';

import { Button } from "@/components/ui/button";

import { Textarea } from "@/components/ui/textarea";

import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Badge } from "@/components/ui/badge";

import { Brain, Send, Sparkles } from 'lucide-react';

export default function QuestionForm({ onSubmitQuestion, isLoading }) {

const [question, setQuestion] = useState('');

const [subject, setSubject] = useState('');

const [difficulty, setDifficulty] = useState('');

const subjects = [

'Mathematics', 'Physics', 'Chemistry', 'Biology', 'Computer Science',

'History', 'Literature', 'Geography', 'Economics', 'Psychology'

];

const handleSubmit = (e) => {

e.preventDefault();

if (!question.trim()) return;

onSubmitQuestion({

question\_text: question,

subject: subject || 'General',

difficulty\_level: difficulty || 'intermediate'

});

};

return (

<Card className="shadow-xl border-0 bg-white/90 backdrop-blur-sm">

<CardHeader className="pb-4">

<CardTitle className="flex items-center gap-2 text-2xl font-bold text-gray-900">

<Brain className="w-7 h-7 text-blue-600" />

Ask Your Question

<Sparkles className="w-5 h-5 text-yellow-500" />

</CardTitle>

<p className="text-gray-600">Get instant, accurate answers from our AI study assistant</p>

</CardHeader>

<CardContent>

<form onSubmit={handleSubmit} className="space-y-6">

<div className="space-y-2">

<label className="text-sm font-medium text-gray-700">Your Question</label>

<Textarea

placeholder="What would you like to learn about? Be as specific as possible for better answers..."

value={question}

onChange={(e) => setQuestion(e.target.value)}

className="min-h-32 text-base resize-none border-gray-200 focus:border-blue-500 focus:ring-blue-500"

disabled={isLoading}

/>

</div>

<div className="grid grid-cols-1 md:grid-cols-2 gap-4">

<div className="space-y-2">

<label className="text-sm font-medium text-gray-700">Subject</label>

<Select value={subject} onValueChange={setSubject} disabled={isLoading}>

<SelectTrigger className="border-gray-200 focus:border-blue-500">

<SelectValue placeholder="Select subject" />

</SelectTrigger>

<SelectContent>

{subjects.map((subj) => (

<SelectItem key={subj} value={subj}>{subj}</SelectItem>

))}

</SelectContent>

</Select>

</div>

<div className="space-y-2">

<label className="text-sm font-medium text-gray-700">Difficulty Level</label>

<Select value={difficulty} onValueChange={setDifficulty} disabled={isLoading}>

<SelectTrigger className="border-gray-200 focus:border-blue-500">

<SelectValue placeholder="Select difficulty" />

</SelectTrigger>

<SelectContent>

<SelectItem value="beginner">

<div className="flex items-center gap-2">

<Badge className="bg-green-100 text-green-800 text-xs">Beginner</Badge>

<span>Basic concepts</span>

</div>

</SelectItem>

<SelectItem value="intermediate">

<div className="flex items-center gap-2">

<Badge className="bg-yellow-100 text-yellow-800 text-xs">Intermediate</Badge>

<span>Standard level</span>

</div>

</SelectItem>

<SelectItem value="advanced">

<div className="flex items-center gap-2">

<Badge className="bg-red-100 text-red-800 text-xs">Advanced</Badge>

<span>Complex topics</span>

</div>

</SelectItem>

</SelectContent>

</Select>

</div>

</div>

<Button

type="submit"

disabled={!question.trim() || isLoading}

className="w-full bg-gradient-to-r from-blue-600 to-indigo-600 hover:from-blue-700 hover:to-indigo-700 text-white font-semibold py-3 rounded-xl transition-all duration-200 shadow-lg hover:shadow-xl"

>

{isLoading ? (

<div className="flex items-center gap-2">

<div className="w-4 h-4 border-2 border-white/30 border-t-white rounded-full animate-spin"></div>

Thinking...

</div>

) : (

<div className="flex items-center gap-2">

<Send className="w-4 h-4" />

Get Answer

</div>

)}

</Button>

</form>

</CardContent>

</Card>

);

}

## components/qa/AnswerDisplay.jsx

import React, { useState } from 'react';

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Badge } from "@/components/ui/badge";

import { Button } from "@/components/ui/button";

import { Separator } from "@/components/ui/separator";

import { CheckCircle, XCircle, ThumbsUp, ThumbsDown, Copy, Share, BookOpen } from 'lucide-react';

export default function AnswerDisplay({ questionData, onRating }) {

const [copied, setCopied] = useState(false);

const [userRating, setUserRating] = useState(questionData.helpful\_rating);

const copyToClipboard = async () => {

await navigator.clipboard.writeText(questionData.ai\_response);

setCopied(true);

setTimeout(() => setCopied(false), 2000);

};

const handleRating = (rating) => {

setUserRating(rating);

onRating(questionData.id, rating);

};

const getDifficultyColor = (difficulty) => {

switch (difficulty) {

case 'beginner': return 'bg-green-100 text-green-800 border-green-200';

case 'intermediate': return 'bg-yellow-100 text-yellow-800 border-yellow-200';

case 'advanced': return 'bg-red-100 text-red-800 border-red-200';

default: return 'bg-gray-100 text-gray-800 border-gray-200';

}

};

return (

<Card className="shadow-xl border-0 bg-white/95 backdrop-blur-sm">

<CardHeader className="pb-4">

<div className="flex items-start justify-between">

<div className="flex-1">

<CardTitle className="text-xl font-bold text-gray-900 mb-3">

{questionData.question\_text}

</CardTitle>

<div className="flex items-center gap-2 flex-wrap">

<Badge variant="outline" className="bg-blue-50 text-blue-700 border-blue-200">

<BookOpen className="w-3 h-3 mr-1" />

{questionData.subject}

</Badge>

<Badge className={\`\${getDifficultyColor(questionData.difficulty\_level)} text-xs border\`}>

{questionData.difficulty\_level}

</Badge>

<span className="text-xs text-gray-500">

{new Date(questionData.created\_date).toLocaleDateString()}

</span>

</div>

</div>

</div>

</CardHeader>

<CardContent className="space-y-6">

<Separator />

<div>

<h4 className="font-semibold text-gray-900 mb-3 flex items-center gap-2">

<CheckCircle className="w-5 h-5 text-green-600" />

AI Answer

</h4>

<div className="bg-gradient-to-r from-blue-50 to-indigo-50 rounded-xl p-6 border border-blue-100">

<div className="prose prose-sm max-w-none text-gray-800 leading-relaxed">

{questionData.ai\_response.split('\n').map((paragraph, index) => (

<p key={index} className="mb-3 last:mb-0">

{paragraph}

</p>

))}

</div>

</div>

</div>

{questionData.sources && questionData.sources.length > 0 && (

<div>

<h4 className="font-semibold text-gray-900 mb-3">Sources & References</h4>

<div className="space-y-2">

{questionData.sources.map((source, index) => (

<div key={index} className="flex items-center gap-2 text-sm text-blue-600 hover:text-blue-800">

<BookOpen className="w-4 h-4" />

<span>{source}</span>

</div>

))}

</div>

</div>

)}

<Separator />

<div className="flex items-center justify-between">

<div className="flex items-center gap-4">

<span className="text-sm font-medium text-gray-700">Was this helpful?</span>

<div className="flex items-center gap-2">

<Button

variant={userRating >= 4 ? "default" : "outline"}

size="sm"

onClick={() => handleRating(5)}

className="flex items-center gap-1"

>

<ThumbsUp className="w-4 h-4" />

Yes

</Button>

<Button

variant={userRating <= 2 ? "default" : "outline"}

size="sm"

onClick={() => handleRating(1)}

className="flex items-center gap-1"

>

<ThumbsDown className="w-4 h-4" />

No

</Button>

</div>

</div>

<div className="flex items-center gap-2">

<Button

variant="outline"

size="sm"

onClick={copyToClipboard}

className="flex items-center gap-1"

>

<Copy className="w-4 h-4" />

{copied ? 'Copied!' : 'Copy'}

</Button>

<Button variant="outline" size="sm" className="flex items-center gap-1">

<Share className="w-4 h-4" />

Share

</Button>

</div>

</div>

</CardContent>

</Card>

);

}

## components/ui/DynamicIcon.jsx

import React from 'react';

import {

HelpCircle,

Award,

Book,

BrainCircuit,

Calendar,

CheckCircle,

Clock,

Rocket,

Star,

Zap,

TrendingUp,

Target

} from 'lucide-react';

const iconMap = {

HelpCircle,

Award,

Book,

BrainCircuit,

Calendar,

CheckCircle,

Clock,

Rocket,

Star,

Zap,

TrendingUp,

Target

};

const DynamicIcon = ({ name, ...props }) => {

const IconComponent = iconMap[name];

if (!IconComponent) {

// Fallback icon if the requested icon name is not found

return <iconMap.HelpCircle {...props} />;

}

return <IconComponent {...props} />;

};

export default DynamicIcon;