GenAi Interactive Learning Games

PROJECT REPORT

Submitted to:

Submitted by:

Fiza Babariya

Karan Soni

Date Of Submission: 5th April 2025

Institution:

PARUL UNIVERSITY, VADODARA, GUJARAT (INDIA)

Acknowledgement

We would like to express our sincere gratitude to Intel for giving us the opportunity to work on our project, GenAI Interactive Learning Games. This experience has allowed us to explore the intersection of AI and education, leveraging cutting-edge technologies to enhance interactive learning.

We extend our deepest appreciation to Parul University for providing us with the resources, knowledge, and encouragement necessary for this project. A special thanks to our faculty mentors for their invaluable guidance and unwavering support throughout this journey.

We also acknowledge the efforts and dedication of our project team, Fiza Babariya and Karan Soni, whose collaboration, innovation, and hard work have been the driving force behind this project.

Finally, we would like to thank our peers, friends, and family for their continuous encouragement, motivation, and insightful feedback, which have been instrumental in shaping our work.

Abstract

The shift toward digital education has highlighted the need for interactive and engaging learning solutions that go beyond traditional teaching methods. Many students face difficulties in maintaining focus, tracking their progress, and retaining information effectively. To bridge this gap, we introduce EduTrack, an intelligent web-based learning platform that combines gamification, real-time analytics, and adaptive learning strategies.

EduTrack enhances the educational experience by incorporating quizzes, flashcards, achievement badges, and progress tracking, making learning both structured and enjoyable. The platform dynamically adapts to individual learning patterns, offering a personalized approach that caters to diverse needs—whether for exam preparation, self-learning, or instructional support. By fostering motivation through rewards and data-driven insights, EduTrack transforms conventional study habits into an interactive and efficient learning journey.

Introduction

In today's ever-evolving educational landscape, the need for innovative and interactive learning tools has become more significant than ever. Traditional educational methods often fall short in terms of engagement, retention, and personalization. Students frequently encounter challenges in tracking their academic progress, maintaining motivation, and managing large volumes of study material effectively. Recognizing these challenges, we present EduTrack – an interactive web-based educational platform designed to make learning more dynamic, engaging, and effective.

EduTrack leverages the power of flashcards, quizzes, gamification, and real-time analytics to enhance the learning experience. It offers a personalized and responsive interface that adapts to different learning needs. With features such as experience points (XP), achievement badges, dynamic flashcard generation, and performance tracking, EduTrack transforms routine learning into an engaging and rewarding activity. The platform is ideal for students preparing for exams, educators looking for effective teaching tools, and self-learners striving for structured progress.

Problem statement: GenAI Interactive Learning games

This project focuses on developing an AI-driven educational game platform that leverages Generative AI (GenAI) to create dynamic content, challenges, and learning scenarios tailored to individual users.

Objectives:

- AI-Generated Content: Automatically generate questions, mini-stories, and puzzles based on user progress.
- Adaptive Learning Paths: Adjust game difficulty and topics in real-time to match the learner's skill level.
- Gamification Elements: Incorporate leveling up, badges, and leaderboards to enhance engagement and motivation.
- Natural Language Interaction: Enable users to ask questions, give answers, and influence the game's narrative using AI-powered chat interfaces.
- Personalized Recommendations: Use AI to suggest next challenges or educational topics based on player performance and preferences.

Methodology

Tech Stack Used:

2.1 Frontend

- Next.js 14: A React framework with support for Server and Client Components, App Router, and code splitting.
- Tailwind CSS: A utility-first CSS framework for building responsive, customizable interfaces.
- shadcn/ui: A component library used for accessible and styled UI components.
- Lucide Icons: Scalable and customizable icon library.
- Framer Motion: A library used to add smooth, interactive animations.
- Canvas-confetti: For celebration animations like quiz success confetti.

2.2 State & Data Management

- React Hooks (useState, useEffect): For managing component-level state.
- localStorage: To persist data such as scores, XP, and flashcards across sessions.

2.3 Hosting & Deployment

- Vercel: Offers instant deployment, automatic CI/CD, and a global CDN for optimal speed and performance.

2.4 Backend (Planned Future Enhancements)

- Node.js & Express: For building RESTful APIs.
- MongoDB: NoSQL database to store flashcards, quizzes, users, and achievements.

System Architecture

EduTrack follows a modular and scalable architecture that separates concerns and allows independent development of features.

The major architectural components include:

- Frontend Interface: Built using Next.js with Tailwind for styling.
- Gamification Engine: Manages XP, badges, levels, and progress visualization.
- Flashcard Generator: Dynamically generates topic-based flashcards.
- Quiz Generator: Creates multiple-choice quizzes from flashcard data.
- Dashboard Analytics: Provides performance insights and progress tracking.
- Authentication Layer: Allows role-based access control and secure login sessions.

Development Approach

4.1 Component-Based Development

The platform is designed using reusable components. Layout components (Header, Footer), interactive elements (Flashcards, Quizzes), and feedback elements (ProgressBar, ConfettiExplosion) are all developed as standalone modules. This promotes maintainability and scalability.

4.2 Agile Methodology

EduTrack was developed using an agile development model. Weekly sprints were held to incrementally build features like UI, Flashcard flipping, quiz logic, gamification, and the dashboard. This approach allowed for continuous feedback and rapid iterations.

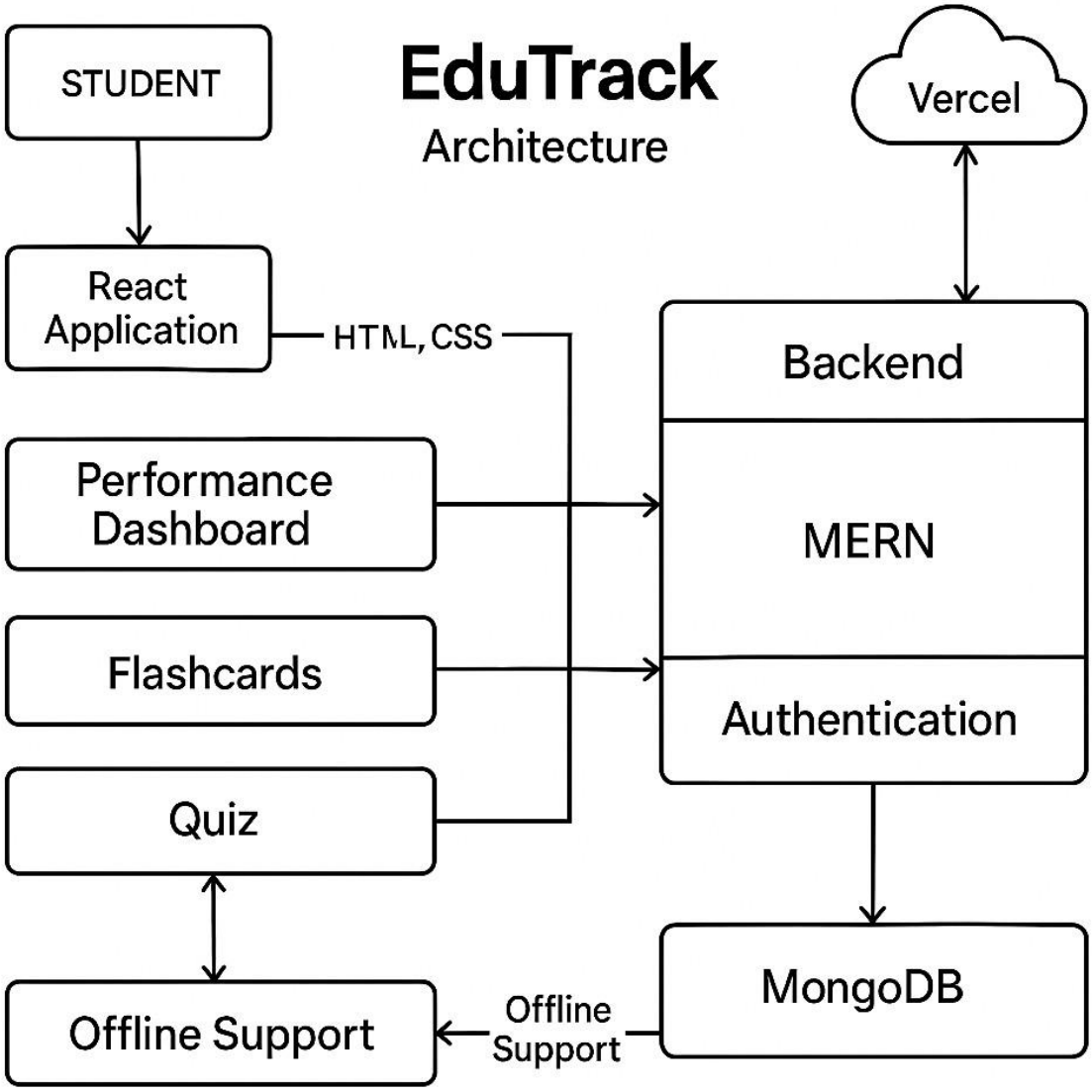
4.3 Responsive Design

The user interface follows a mobile-first design philosophy. With the help of Flexbox, CSS Grid, and responsive breakpoints, the application adapts smoothly across different screen sizes, offering a seamless experience on both desktop and mobile devices.

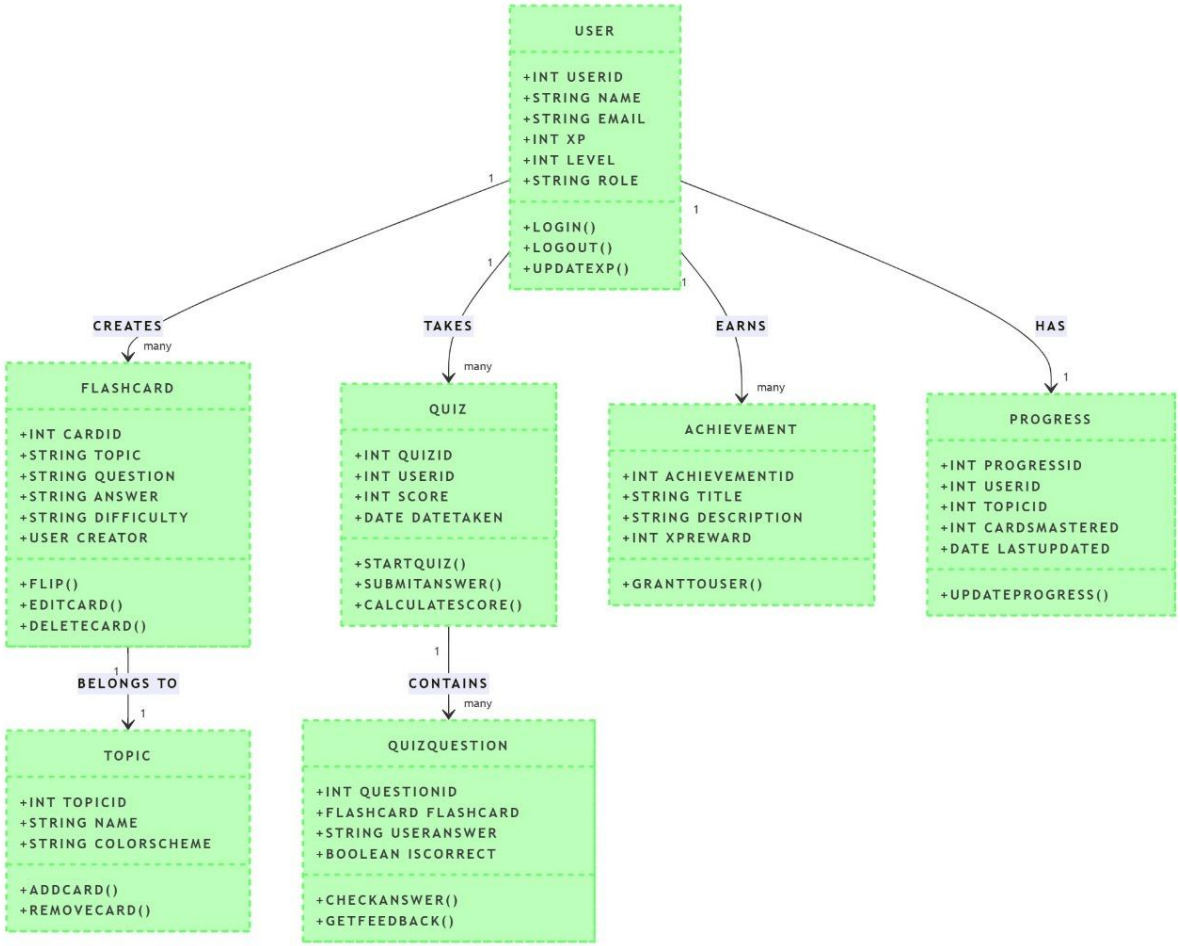
4.4 User-Centric Design

User experience was prioritized at every development stage. Key features such as easy navigation, intuitive controls, and visual feedback were added to improve usability and satisfaction.

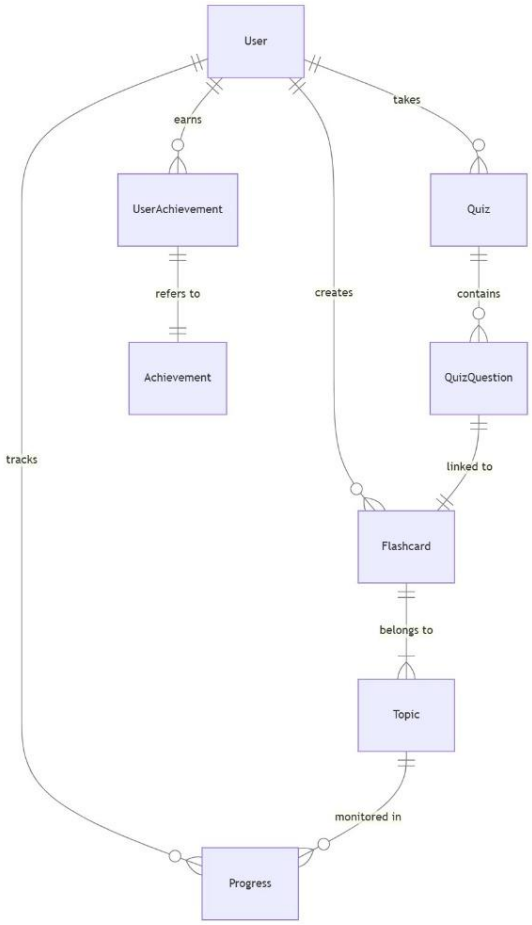
Architecture Diagram



Class Diagram



ER Diagram


User
earns
takes
UserAchievement
Quiz
refers to
creates
contains
Achievement
QuizQuestion
tracks
linked to
Flashcard
belongs to
Topic
monitored in
Progress

Directory Structure

/project-root

|— app

| ├— create

| ├— dashboard

| ├— globals.css

| ├— layout.tsx

| ├— page.tsx

|

|— components

| ├— ui

| | ├— animated-card.tsx

| | ├— confetti-explosion.tsx

| | ├— floating-element.tsx

| | ├— theme-provider.tsx

|

|— data

| ├— flashcard-topics.ts

|

|— hooks

| ├— use-mobile.tsx

| ├— use-toast.ts

|

|— lib

| ├— utils.ts

|

|— public

| ├— placeholder-logo.png

| ├— placeholder-logo.svg

| ├— placeholder-user.jpg

| ├— placeholder.jpg

| ├── placeholder.svg

|

|── styles

| ├── globals.css

|

|── .gitignore

|── components.json

|── image-1.png

|── image-2.png

|── image-3.png

|── image-4.png

|── image-5.png

|── image.png

|── next.config.mjs

|── package.json

|── pnpm-lock.yaml

|── postcss.config.mjs

|── README.md

|── tailwind.config.js

|── tsconfig.js**on**

Links

Rest code on Github Link : https://github.com/qwertykaran/flashcard.git

Video link : video link

Implementation Code

Page.tsx file:

```
"use client"


import { useState, useEffect } from "react"
import { useSearchParams } from "next/navigation"
import {
  ArrowLeft,
  ArrowRight,
  Sparkles,
  Trophy,
  Star,
  CheckCircle,
  XCircle,
  Zap,
  Award,
  Flag,
  Lightbulb,
} from "lucide-react"
import Link from "next/link"
import { motion, AnimatePresence } from "framer-motion"
import { Button } from "@/components/ui/button"
import { Card, CardContent } from "@/components/ui/card"
import { Input } from "@/components/ui/input"
import { Label } from "@/components/ui/label"
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs"
import { Progress } from "@/components/ui/progress"
import { Badge } from "@/components/ui/badge"
```

```
import { Tooltip, TooltipContent, TooltipProvider, TooltipTrigger } from "@/components/ui/tooltip"
import { ConfettiExplosion } from "@/components/confetti-explosion"
import { topicData, topicIcons } from "@/data/flashcard-topics"

// Define checkpoints for the learning ladder
const checkpoints = [
  { id: 1, name: "Beginner", cards: 1, icon: Star, color: "text-blue-500" },
  { id: 2, name: "Explorer", cards: 2, icon: Zap, color: "text-green-500" },
  { id: 3, name: "Scholar", cards: 3, icon: Award, color: "text-yellow-500" },
  { id: 4, name: "Expert", cards: 4, icon: Trophy, color: "text-orange-500" },
  { id: 5, name: "Master", cards: 5, icon: Flag, color: "text-red-500" },
]

// Function to get background styles based on topic and difficulty
const getCardBackground = (topic, difficulty) => {
  const backgrounds = {
    monuments: {
      easy: "bg-gradient-to-br from-blue-50 to-blue-100 border-blue-200",
      medium: "bg-gradient-to-br from-blue-100 to-blue-200 border-blue-300",
      hard: "bg-gradient-to-br from-blue-200 to-blue-300 border-blue-400",
    },
    space: {
      easy: "bg-gradient-to-br from-indigo-50 to-purple-100 border-purple-200",
      medium: "bg-gradient-to-br from-indigo-100 to-purple-200 border-purple-300",
      hard: "bg-gradient-to-br from-indigo-200 to-purple-300 border-purple-400",
    },
    animals: {
      easy: "bg-gradient-to-br from-green-50 to-emerald-100 border-emerald-200",
      medium: "bg-gradient-to-br from-green-100 to-emerald-200 border-emerald-300",
```

```
  hard: "bg-gradient-to-br from-green-200 to-emerald-300 border-emerald-400",
},
plants: {
  easy: "bg-gradient-to-br from-emerald-50 to-green-100 border-green-200",
  medium: "bg-gradient-to-br from-emerald-100 to-green-200 border-green-300",
  hard: "bg-gradient-to-br from-emerald-200 to-green-300 border-green-400",
},
math: {
  easy: "bg-gradient-to-br from-red-50 to-orange-100 border-orange-200",
  medium: "bg-gradient-to-br from-red-100 to-orange-200 border-orange-300",
  hard: "bg-gradient-to-br from-red-200 to-orange-300 border-orange-400",
},
geography: {
  easy: "bg-gradient-to-br from-cyan-50 to-blue-100 border-blue-200",
  medium: "bg-gradient-to-br from-cyan-100 to-blue-200 border-blue-300",
  hard: "bg-gradient-to-br from-cyan-200 to-blue-300 border-blue-400",
},
literature: {
  easy: "bg-gradient-to-br from-amber-50 to-yellow-100 border-yellow-200",
  medium: "bg-gradient-to-br from-amber-100 to-yellow-200 border-yellow-300",
  hard: "bg-gradient-to-br from-amber-200 to-yellow-300 border-yellow-400",
},
biology: {
  easy: "bg-gradient-to-br from-lime-50 to-green-100 border-green-200",
  medium: "bg-gradient-to-br from-lime-100 to-green-200 border-green-300",
  hard: "bg-gradient-to-br from-lime-200 to-green-300 border-green-400",
},
music: {
  easy: "bg-gradient-to-br from-fuchsia-50 to-pink-100 border-pink-200",
```

```
  medium: "bg-gradient-to-br from-fuchsia-100 to-pink-200 border-pink-300",
  hard: "bg-gradient-to-br from-fuchsia-200 to-pink-300 border-pink-400",
},
art: {
  easy: "bg-gradient-to-br from-rose-50 to-pink-100 border-pink-200",
  medium: "bg-gradient-to-br from-rose-100 to-pink-200 border-pink-300",
  hard: "bg-gradient-to-br from-rose-200 to-pink-300 border-pink-400",
},
history: {
  easy: "bg-gradient-to-br from-amber-50 to-orange-100 border-orange-200",
  medium: "bg-gradient-to-br from-amber-100 to-orange-200 border-orange-300",
  hard: "bg-gradient-to-br from-amber-200 to-orange-300 border-orange-400",
},
geology: {
  easy: "bg-gradient-to-br from-stone-50 to-stone-100 border-stone-200",
  medium: "bg-gradient-to-br from-stone-100 to-stone-200 border-stone-300",
  hard: "bg-gradient-to-br from-stone-200 to-stone-300 border-stone-400",
},
nutrition: {
  easy: "bg-gradient-to-br from-lime-50 to-yellow-100 border-yellow-200",
  medium: "bg-gradient-to-br from-lime-100 to-yellow-200 border-yellow-300",
  hard: "bg-gradient-to-br from-lime-200 to-yellow-300 border-yellow-400",
},
brain: {
  easy: "bg-gradient-to-br from-violet-50 to-purple-100 border-purple-200",
  medium: "bg-gradient-to-br from-violet-100 to-purple-200 border-purple-300",
  hard: "bg-gradient-to-br from-violet-200 to-purple-300 border-purple-400",
},
physics: {
```

```
    easy: "bg-gradient-to-br from-sky-50 to-blue-100 border-blue-200",
    medium: "bg-gradient-to-br from-sky-100 to-blue-200 border-blue-300",
    hard: "bg-gradient-to-br from-sky-200 to-blue-300 border-blue-400",
  },
  chemistry: {
    easy: "bg-gradient-to-br from-teal-50 to-cyan-100 border-cyan-200",
    medium: "bg-gradient-to-br from-teal-100 to-cyan-200 border-cyan-300",
    hard: "bg-gradient-to-br from-teal-200 to-cyan-300 border-cyan-400",
  },
  weather: {
    easy: "bg-gradient-to-br from-blue-50 to-cyan-100 border-cyan-200",
    medium: "bg-gradient-to-br from-blue-100 to-cyan-200 border-cyan-300",
    hard: "bg-gradient-to-br from-blue-200 to-cyan-300 border-cyan-400",
  },
  // Default fallback
  default: {
    easy: "bg-gradient-to-br from-slate-50 to-slate-100 border-slate-200",
    medium: "bg-gradient-to-br from-slate-100 to-slate-200 border-slate-300",
    hard: "bg-gradient-to-br from-slate-200 to-slate-300 border-slate-400",
  },
 }

 // Get the background for the topic and difficulty, or use default if not found
 const topicBg = backgrounds[topic] || backgrounds.default
 return topicBg[difficulty] || topicBg.medium
}

// Function to get decorative elements based on topic
const getTopicDecoration = (topic) => {
```

```
const decorations = {
  monuments: "after:content-['🏛'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
  space: "after:content-['🚀'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
  animals: "after:content-['🦁'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
  plants: "after:content-['🌿'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
  math: "after:content-['📊'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
  geography: "after:content-['🌎'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
  literature: "after:content-['📚'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
  biology: "after:content-['🧬'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
  music: "after:content-['🎵'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
  art: "after:content-['🎨'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
  history: "after:content-['⏳'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
  geology: "after:content-['🏔'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
  nutrition: "after:content-['🍎'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
  brain: "after:content-['🧠'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
  physics: "after:content-['⚛'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
  chemistry: "after:content-['🧪'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
```

```
    weather: "after:content-['☁️'] after:absolute after:bottom-4 after:right-4 after:opacity-10 after:text-5xl",
    default: "",
  }

  return decorations[topic] || decorations.default
}

// Quiz questions based on the flashcards
const generateQuiz = (flashcards) => {
  return flashcards.map((card) => {
    return {
      question: card.question,
      options: generateOptions(card.answer),
      correctAnswer: card.answer,
      difficulty: card.difficulty,
    }
  })
}

// Generate random options for multiple choice questions
const generateOptions = (correctAnswer) => {
  // In a real app, we would generate better distractors
  // For now, we'll just create some dummy options
  const options = [
    correctAnswer,
    "This is an incorrect option",
    "This is another incorrect option",
    "This is a third incorrect option",
```

```
  ]

  // Shuffle the options
  return options.sort(() => Math.random() - 0.5)
}

export default function CreatePage() {
  const searchParams = useSearchParams()
  const initialTopic = searchParams.get("topic") || ""

  const [topic, setTopic] = useState(initialTopic)
  const [generatedTopic, setGeneratedTopic] = useState("")
  const [flashcards, setFlashcards] = useState([])
  const [quiz, setQuiz] = useState([])
  const [currentCardIndex, setCurrentCardIndex] = useState(0)
  const [isFlipped, setIsFlipped] = useState(false)
  const [currentQuizIndex, setCurrentQuizIndex] = useState(0)
  const [selectedAnswer, setSelectedAnswer] = useState("")
  const [score, setScore] = useState(0)
  const [quizCompleted, setQuizCompleted] = useState(false)
  const [activeTab, setActiveTab] = useState("input")

  // Gamification state
  const [masteredCards, setMasteredCards] = useState([])
  const [currentLevel, setCurrentLevel] = useState(0)
  const [xp, setXp] = useState(0)
  const [showFeedback, setShowFeedback] = useState(false)
  const [isCorrect, setIsCorrect] = useState(false)
  const [checkpointReached, setCheckpointReached] = useState(false)
```

```
const [currentCheckpoint, setCurrentCheckpoint] = useState(null)
const [showConfetti, setShowConfetti] = useState(false)
const [tipVisible, setTipVisible] = useState(false)

// Show random study tips
useEffect(() => {
  const interval = setInterval(() => {
    setTipVisible(true)
    setTimeout(() => setTipVisible(false), 5000)
  }, 15000)

  return () => clearInterval(interval)
}, [])

// Auto-generate if topic is provided in URL
useEffect(() => {
  if (initialTopic && Object.keys(topicData).includes(initialTopic)) {
    setTopic(initialTopic)
    handleGenerate(initialTopic)
  }
}, [initialTopic])

const handleGenerate = (topicToGenerate = topic) => {
  // Normalize the topic to lowercase for matching
  const normalizedTopic = topicToGenerate.toLowerCase().trim()

  // Find the closest match in our sample data
  let matchedTopic = null
  for (const key in topicData) {
```

```
  if (normalizedTopic.includes(key) || key.includes(normalizedTopic)) {
    matchedTopic = key
    break
  }
}

// If no match found, default to monuments
const cards = matchedTopic ? topicData[matchedTopic] : topicData.monuments

// Sort cards by difficulty for the learning ladder
const sortedCards = [...cards].sort((a, b) => {
  const difficultyOrder = { easy: 1, medium: 2, hard: 3 }
  return difficultyOrder[a.difficulty] - difficultyOrder[b.difficulty]
})

setGeneratedTopic(matchedTopic || "monuments")
setFlashcards(sortedCards)
setQuiz(generateQuiz(sortedCards))
setCurrentCardIndex(0)
setCurrentQuizIndex(0)
setIsFlipped(false)
setScore(0)
setQuizCompleted(false)
setActiveTab("flashcards")

// Reset gamification state
setMasteredCards([])
setCurrentLevel(0)
setXp(0)
```

```
  setShowFeedback(false)
  setCheckpointReached(false)
  setCurrentCheckpoint(checkpoints[0])
}

const nextCard = () => {
  if (currentCardIndex < flashcards.length - 1) {
    setCurrentCardIndex(currentCardIndex + 1)
    setIsFlipped(false)
  }
}

const prevCard = () => {
  if (currentCardIndex > 0) {
    setCurrentCardIndex(currentCardIndex - 1)
    setIsFlipped(false)
  }
}

const flipCard = () => {
  setIsFlipped(!isFlipped)
}

const markCardMastered = () => {
  if (!masteredCards.includes(currentCardIndex)) {
    const newMasteredCards = [...masteredCards, currentCardIndex]
    setMasteredCards(newMasteredCards)

    // Add XP based on card difficulty
```

```
    const cardDifficulty = flashcards[currentCardIndex].difficulty
    const xpGain = cardDifficulty === "easy" ? 10 : cardDifficulty === "medium" ? 20 : 30
    setXp(xp + xpGain)

    // Check if we've reached a checkpoint
    const nextCheckpointIndex = checkpoints.findIndex((cp) => cp.cards === newMasteredCards.length)
    if (nextCheckpointIndex !== -1) {
      setCheckpointReached(true)
      setCurrentCheckpoint(checkpoints[nextCheckpointIndex])
      setCurrentLevel(nextCheckpointIndex + 1)
      setShowConfetti(true)
      setTimeout(() => setShowConfetti(false), 3000)
    }

    // Show positive feedback
    setIsCorrect(true)
    setShowFeedback(true)
    setTimeout(() => {
      setShowFeedback(false)
      if (checkpointReached) {
        setCheckpointReached(false)
      }
    }, 2000)
  }
 }

 const markCardNotMastered = () => {
  // Remove from mastered if it was there
  if (masteredCards.includes(currentCardIndex)) {
```

```
    setMasteredCards(masteredCards.filter((index) => index !== currentCardIndex))
  }

  // Show negative feedback
  setIsCorrect(false)
  setShowFeedback(true)
  setTimeout(() => {
    setShowFeedback(false)
  }, 2000)
}

const handleAnswerSelect = (answer) => {
  setSelectedAnswer(answer)
}

const submitAnswer = () => {
  const isAnswerCorrect = selectedAnswer === quiz[currentQuizIndex].correctAnswer

  if (isAnswerCorrect) {
    setScore(score + 1)
  }

  // Show feedback
  setIsCorrect(isAnswerCorrect)
  setShowFeedback(true)

  setTimeout(() => {
    setShowFeedback(false)
```

```
    if (currentQuizIndex < quiz.length - 1) {
      setCurrentQuizIndex(currentQuizIndex + 1)
      setSelectedAnswer("")
    } else {
      setQuizCompleted(true)
      setShowConfetti(true)
      setTimeout(() => setShowConfetti(false), 3000)
      // Save quiz results to localStorage for dashboard
      saveQuizResults()
    }
  }, 1500)
}

// Add this new function
const saveQuizResults = () => {
  // In a real app, this would save to a database
  // For now, we'll just simulate saving to localStorage
  try {
    const quizResult = {
      id: Date.now(),
      topic: generatedTopic,
      score: score + (selectedAnswer === quiz[currentQuizIndex].correctAnswer ? 1 : 0),
      total: quiz.length,
      date: new Date().toISOString().split("T")[0],
      xpEarned: xp,
      level: currentLevel,
    }

    // Get existing results or initialize empty array
```

```
    const existingResults = JSON.parse(localStorage.getItem("quizResults") || "[]")

    // Add new result
    const updatedResults = [quizResult, ...existingResults]

    // Save back to localStorage
    localStorage.setItem("quizResults", JSON.stringify(updatedResults))

    console.log("Quiz results saved:", quizResult)
  } catch (error) {
    console.error("Error saving quiz results:", error)
  }
}

const restartQuiz = () => {
  setCurrentQuizIndex(0)
  setSelectedAnswer("")
  setScore(0)
  setQuizCompleted(false)
}

const startQuiz = () => {
  setActiveTab("quiz")
  setCurrentQuizIndex(0)
  setSelectedAnswer("")
  setScore(0)
  setQuizCompleted(false)
}
```

```
  // Calculate progress percentage for the learning ladder
  const progressPercentage = flashcards.length > 0 ? (masteredCards.length / flashcards.length) * 100 : 0

  // Get current topic icon
  const TopicIcon = generatedTopic ? topicIcons[generatedTopic] || Sparkles : Sparkles

  // Study tips
  const studyTips = [
    "Try explaining the concept in your own words to improve retention!",
    "Taking short breaks between study sessions can improve focus.",
    "Connecting new information to things you already know helps memory.",
    "Testing yourself is more effective than just re-reading material.",
    "Studying in different locations can improve recall.",
    "Teaching someone else is one of the best ways to master a topic!",
  ]

  const randomTip = studyTips[Math.floor(Math.random() * studyTips.length)]

  return (
    <div className="flex min-h-screen flex-col">
      {showConfetti && <ConfettiExplosion />}

      <header className="sticky top-0 z-50 w-full border-b bg-background/95 backdrop-blur supports-[backdrop-filter]:bg-background/60">
        <div className="container flex h-14 items-center">
```

```
    <div className="mr-4 flex">
      <Link href="/" className="flex items-center space-x-2">
        <Sparkles className="h-6 w-6 text-primary animate-pulse" />
        <span className="font-bold text-xl">EduTrack</span>
      </Link>
    </div>
    <nav className="flex flex-1 items-center justify-end space-x-4">
      <Button variant="ghost" asChild>
        <Link href="/">Home</Link>
      </Button>
      <Button variant="ghost" asChild>
        <Link href="/create">Create</Link>
      </Button>
      <Button variant="ghost" asChild>
        <Link href="/dashboard">Dashboard</Link>
      </Button>
    </nav>
  </div>
</header>

<main className="flex-1 container py-8">
  {/* Study Tip */}
  <AnimatePresence>
    {tipVisible && (
      <motion.div
        initial={{ opacity: 0, y: -20 }}
```

```
          animate={{ opacity: 1, y: 0 }}
          exit={{ opacity: 0, y: -20 }}
          className="mb-6 bg-yellow-50 border border-yellow-200 rounded-lg p-4 flex items-start gap-3"
        >
          <Lightbulb className="h-5 w-5 text-yellow-500 flex-shrink-0 mt-0.5" />
          <div>
            <p className="font-medium text-yellow-800">Study Tip:</p>
            <p className="text-yellow-700">{randomTip}</p>
          </div>
        </motion.div>
      )}
    </AnimatePresence>

    <Tabs value={activeTab} onValueChange={setActiveTab} className="w-full max-w-3xl mx-auto">
      <TabsList className="grid w-full grid-cols-3">
        <TabsTrigger
          value="input"
          className="data-[state=active]:bg-primary data-[state=active]:text-primary-foreground"
        >
          Topic Input
        </TabsTrigger>
        <TabsTrigger
          value="flashcards"
          disabled={flashcards.length === 0}
```

```
          className="data-[state=active]:bg-primary data-[state=active]:text-primary-foreground"
          >
            Flashcards
          </TabsTrigger>
          <TabsTrigger
            value="quiz"
            disabled={flashcards.length === 0}
            className="data-[state=active]:bg-primary data-[state=active]:text-primary-foreground"
          >
            Quiz
          </TabsTrigger>
        </TabsList>

        <TabsContent value="input" className="mt-6">
          <motion.div initial={{ opacity: 0, y: 20 }} animate={{ opacity: 1, y: 0 }} transition={{ duration: 0.5 }}>
            <Card>
              <CardContent className="pt-6">
                <div className="space-y-4">
                  <div className="space-y-2">
                    <Label htmlFor="topic">Enter a topic to study</Label>
                    <Input
                      id="topic"
                      placeholder="e.g., monuments, space, animals, math, geography..."
```

```
            value={topic}
            onChange={(e) => setTopic(e.target.value)}
            className="transition-all focus:ring-2 focus:ring-primary"
          />
        </div>
        <Button
          onClick={() => handleGenerate()}
          disabled={!topic.trim()}
          className="w-full transition-all hover:scale-105"
        >
          Generate Flashcards
          <Sparkles className="ml-2 h-4 w-4 animate-pulse" />
        </Button>

        <div className="pt-6">
          <h3 className="font-medium mb-3">Or choose a topic:</h3>
          <div className="grid grid-cols-3 sm:grid-cols-4 md:grid-cols-5 gap-2">
            {Object.entries(topicIcons).map(([topicName, Icon]) => (
              <Button
                key={topicName}
                variant="outline"
                className="flex flex-col h-auto py-3 gap-2 capitalize hover:bg-primary/10 hover:text-primary hover:border-primary/30"
                onClick={() => {
                  setTopic(topicName)
                  handleGenerate(topicName)
```

```
              }}
            >
              <Icon className="h-5 w-5" />
              <span className="text-xs">{topicName}</span>
            </Button>
          ))}
        </div>
      </div>
    </div>
  </CardContent>
 </Card>
</motion.div>
</TabsContent>

<TabsContent value="flashcards" className="mt-6">
 {flashcards.length > 0 && (
  <div className="space-y-6">
   <motion.div
    initial={{ opacity: 0, y: 20 }}
    animate={{ opacity: 1, y: 0 }}
    transition={{ duration: 0.5 }}
    className="flex justify-between items-center"
   >
    <div className="flex items-center gap-2">
     <div className="p-2 bg-primary/10 rounded-full">
      <TopicIcon className="h-5 w-5 text-primary" />
```

```
            </div>
            <h2 className="text-2xl font-bold capitalize">{generatedTopic}</h2>
          </div>
          <div className="flex items-center gap-2">
            <Badge variant="outline" className="bg-primary/10 text-primary animate-pulse">
              XP: {xp}
            </Badge>
            <Button onClick={startQuiz} variant="outline" className="group">
              Take Quiz
              <ArrowRight className="ml-2 h-4 w-4 group-hover:translate-x-1 transition-transform" />
            </Button>
          </div>
        </motion.div>

        {/* Learning Ladder Progress */}
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ delay: 0.1, duration: 0.5 }}
          className="bg-muted p-4 rounded-lg"
        >
          <div className="flex justify-between items-center mb-2">
            <h3 className="font-medium">Learning Ladder</h3>
            <span className="text-sm text-muted-foreground">
```

```
            {masteredCards.length}/{flashcards.length} Cards Mastered
          </span>
        </div>

        <Progress value={progressPercentage} className="h-2 mb-4" />

        <div className="relative">
          {/* Checkpoint line */}
          <div className="absolute top-5 left-0 right-0 h-0.5 bg-muted-foreground/20"></div>

          {/* Checkpoints */}
          <div className="flex justify-between relative">
            {checkpoints.slice(0, flashcards.length).map((checkpoint, index) => {
              const CheckpointIcon = checkpoint.icon
              const isActive = masteredCards.length >= checkpoint.cards
              const isCurrent = currentLevel === index + 1

              return (
                <TooltipProvider key={checkpoint.id}>
                  <Tooltip>
                    <TooltipTrigger asChild>
                      <motion.div
                        className={`flex flex-col items-center ${
                          isActive ? checkpoint.color : "text-muted-foreground/40"
                        } ${isCurrent ? "scale-125 transition-transform" : ""}`}
```

```
                whileHover={{ scale: 1.1 }}
                animate={
                  isCurrent
                    ? {
                        scale: [1.25, 1.15, 1.25],
                        transition: {
                          repeat: Number.POSITIVE_INFINITY,
                          duration: 1.5,
                        },
                      }
                    : {}
                }
              >
                <CheckpointIcon
                  className={size-6 ${isActive ? "text-current" : "text-muted-foreground/40"}}
                />
                <span className="text-xs mt-1 font-medium">{checkpoint.name}</span>
              </motion.div>
            </TooltipTrigger>
            <TooltipContent>
              <p>Master {checkpoint.cards} cards to reach this level</p>
            </TooltipContent>
          </Tooltip>
        </TooltipProvider>
      )
```

```
          })}
        </div>
      </div>
    </motion.div>

    {/* Checkpoint Reached Notification */}
    <AnimatePresence>
      {checkpointReached && currentCheckpoint && (
        <motion.div
          initial={{ opacity: 0, scale: 0.8 }}
          animate={{ opacity: 1, scale: 1 }}
          exit={{ opacity: 0, scale: 0.8 }}
          className="bg-primary/10 border border-primary/20 text-primary p-4 rounded-lg flex items-center gap-3"
        >
          <Trophy className="h-6 w-6 animate-bounce" />
          <div>
            <h3 className="font-bold">Checkpoint Reached!</h3>
            <p>You've reached the {currentCheckpoint.name} level!</p>
          </div>
        </motion.div>
      )}
    </AnimatePresence>

    {/* Flashcard */}
    <div className="relative">
      <motion.div
```

```
            initial={{ opacity: 0, y: 20 }}
            animate={{ opacity: 1, y: 0 }}
            transition={{ delay: 0.2, duration: 0.5 }}
          >
            <Card className="relative h-[400px] transition-all duration-500 transform-gpu">
              <div
                className={`absolute inset-0 w-full h-full transition-all duration-500 transform-gpu ${
                  isFlipped ? "rotate-y-180 invisible" : ""
                } ${getCardBackground(generatedTopic, flashcards[currentCardIndex].difficulty)} ${getTopicDecoration(generatedTopic)} overflow-hidden border-2`}
                onClick={flipCard}
              >
                <CardContent className="flex flex-col items-center justify-center h-full p-6 text-center relative z-10">
                  <div className="max-w-full">
                    <Badge
                      className={`mb-4 ${
                        flashcards[currentCardIndex].difficulty === "easy"
                          ? "bg-green-100 text-green-800 hover:bg-green-100"
                          : flashcards[currentCardIndex].difficulty === "medium"
                            ? "bg-yellow-100 text-yellow-800 hover:bg-yellow-100"
                            : "bg-red-100 text-red-800 hover:bg-red-100"
                      }`}
                    >
```

```
{flashcards[currentCardIndex].difficulty.charAt(0).toUpperCase() +
                    flashcards[currentCardIndex].difficulty.slice(1)}
                  </Badge>
                  <h3 className="text-xl font-medium mb-4">{flashcards[currentCardIndex].question}</h3>

                  <p className="text-sm text-muted-foreground mt-4">Tap card to see answer</p>
                 </div>
                </CardContent>
               </div>
               <div
                 className={`absolute inset-0 w-full h-full transition-all duration-500 transform-gpu ${
                  isFlipped ? "" : "rotate-y-180 invisible"
                 } bg-gradient-to-br from-slate-100 to-slate-200 border-2 border-slate-300 overflow-hidden`}
                 onClick={flipCard}
               >
                 <div className="absolute inset-0 bg-white/70"></div>
                 <CardContent className="flex flex-col items-center justify-center h-full p-6 text-center relative z-10">
                  <p className="mb-6 text-lg font-medium">{flashcards[currentCardIndex].answer}</p>

                  <div className="flex gap-2 mt-4">
                   <motion.div whileHover={{ scale: 1.05 }} whileTap={{ scale: 0.95 }}>
```

```
                    <Button
                      variant="outline"
                      size="sm"
                      className="bg-red-100 text-red-800 hover:bg-red-200 border-red-200"
                      onClick={markCardNotMastered}
                    >
                      <XCircle className="mr-1 h-4 w-4" />
                      Still Learning
                    </Button>
                  </motion.div>
                  <motion.div whileHover={{ scale: 1.05 }} whileTap={{ scale: 0.95 }}>
                    <Button
                      variant="outline"
                      size="sm"
                      className="bg-green-100 text-green-800 hover:bg-green-200 border-green-200"
                      onClick={markCardMastered}
                    >
                      <CheckCircle className="mr-1 h-4 w-4" />
                      Got It!
                    </Button>
                  </motion.div>
                </div>
              </CardContent>
            </div>
```

```
            {/* Mastery indicator */}
            {masteredCards.includes(currentCardIndex) && (
              <motion.div
                initial={{ opacity: 0, scale: 0.5 }}
                animate={{ opacity: 1, scale: 1 }}
                className="absolute top-2 right-2 z-10"
              >
                <Badge className="bg-green-500 hover:bg-green-500">
                  <CheckCircle className="mr-1 h-3 w-3" />
                  Mastered
                </Badge>
              </motion.div>
            )}
          </Card>

          {/* Feedback overlay */}
          <AnimatePresence>
            {showFeedback && (
              <motion.div
                initial={{ opacity: 0 }}
                animate={{ opacity: 1 }}
                exit={{ opacity: 0 }}
                className={`absolute inset-0 flex items-center justify-center rounded-lg bg-opacity-80 ${
                  isCorrect ? "bg-green-500/20" : "bg-red-500/20"
                }`}
```

```
                >
                  <motion.div
                    initial={{ scale: 0.5 }}
                    animate={{ scale: 1.2 }}
                    transition={{
                      type: "spring",
                      stiffness: 300,
                      damping: 10,
                    }}
                    className={text-4xl ${isCorrect ? "text-green-500" : "text-red-
500"}}
                  >
                    {isCorrect ? (
                      <div className="flex flex-col items-center">
                        <CheckCircle className="h-16 w-16" />
                        <span className="text-lg font-bold mt-2">
                          {isCorrect
                            ? +${flashcards[currentCardIndex].difficulty === "easy" ? 10
: flashcards[currentCardIndex].difficulty === "medium" ? 20 : 30} XP
                            : ""}
                        </span>
                      </div>
                    ) : (
                      <XCircle className="h-16 w-16" />
                    )}
                  </motion.div>
                </motion.div>
```

```
          )}
        </AnimatePresence>
      </motion.div>
    </div>

    <motion.div
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ delay: 0.3, duration: 0.5 }}
      className="flex justify-between items-center"
    >
      <motion.div whileHover={{ scale: 1.05 }} whileTap={{ scale: 0.95 }}>
        <Button onClick={prevCard} disabled={currentCardIndex === 0} variant="outline">
          <ArrowLeft className="mr-2 h-4 w-4" />
          Previous
        </Button>
      </motion.div>
      <div className="text-sm text-muted-foreground">
        Card {currentCardIndex + 1} of {flashcards.length}
      </div>
      <motion.div whileHover={{ scale: 1.05 }} whileTap={{ scale: 0.95 }}>
        <Button onClick={nextCard} disabled={currentCardIndex === flashcards.length - 1} variant="outline">
          Next
          <ArrowRight className="ml-2 h-4 w-4" />
        </Button>
```

```
          </motion.div>
        </motion.div>

        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ delay: 0.4, duration: 0.5 }}
          className="text-center"
        >
          <Button onClick={flipCard} variant="ghost" className="group">
            {isFlipped ? (
              <>
                <motion.span
                  animate={{ rotate: [0, -10, 0, 10, 0] }}
                  transition={{ repeat: Number.POSITIVE_INFINITY, repeatDelay: 3
}}
                >
                  👁
                </motion.span>
                <span className="ml-2">Show Question</span>
              </>
            ) : (
              <>
                <motion.span
                  animate={{ rotate: [0, -10, 0, 10, 0] }}
                  transition={{ repeat: Number.POSITIVE_INFINITY, repeatDelay: 3
}}
```

```
            >
              🔍
            </motion.span>
            <span className="ml-2">Show Answer</span>
          </>
        )}
      </Button>
    </motion.div>
  </div>
 )}
</TabsContent>

<TabsContent value="quiz" className="mt-6">
 {quiz.length > 0 && !quizCompleted && (
  <div className="space-y-6">
   <motion.div
    initial={{ opacity: 0, y: 20 }}
    animate={{ opacity: 1, y: 0 }}
    transition={{ duration: 0.5 }}
    className="flex justify-between items-center"
   >
    <div className="flex items-center gap-2">
     <div className="p-2 bg-primary/10 rounded-full">
      <TopicIcon className="h-5 w-5 text-primary" />
     </div>
```

```
            <h2 className="text-2xl font-bold capitalize">Quiz: {generatedTopic}</h2>
          </div>
          <div className="flex items-center gap-2">
            <Badge variant="outline" className="bg-primary/10 text-primary">
              XP: {xp}
            </Badge>
            <div className="text-sm text-muted-foreground">
              Question {currentQuizIndex + 1} of {quiz.length}
            </div>
          </div>
        </motion.div>

        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ delay: 0.1, duration: 0.5 }}
        >
          <Card
            className={overflow-hidden border-2 ${getCardBackground(generatedTopic, quiz[currentQuizIndex].difficulty)} ${getTopicDecoration(generatedTopic)}}
          >
            <CardContent className="pt-6 relative z-10">
              <div className="space-y-4">
                <div className="flex items-center gap-2">
                  <Badge
```

```
                    className={`${
                      quiz[currentQuizIndex].difficulty === "easy"
                        ? "bg-green-100 text-green-800 hover:bg-green-100"
                        : quiz[currentQuizIndex].difficulty === "medium"
                          ? "bg-yellow-100 text-yellow-800 hover:bg-yellow-100"
                          : "bg-red-100 text-red-800 hover:bg-red-100"
                    }`}
                  >
                    {quiz[currentQuizIndex].difficulty.charAt(0).toUpperCase() +
                      quiz[currentQuizIndex].difficulty.slice(1)}
                  </Badge>
                  <h3 className="text-xl font-medium">{quiz[currentQuizIndex].question}</h3>
                </div>

                <div className="space-y-2">
                  {quiz[currentQuizIndex].options.map((option, index) => (
                    <motion.div key={index} whileHover={{ scale: 1.02 }} whileTap={{ scale: 0.98 }}>
                      <div
                        className={`p-4 border rounded-md cursor-pointer transition-colors backdrop-blur-sm ${
                          selectedAnswer === option
                            ? "border-primary bg-white/80 shadow-md"
                            : "hover:bg-white/60 bg-white/40"
                        }`}
                        onClick={() => handleAnswerSelect(option)}
```

```
                >
                  {option}
                </div>
              </motion.div>
            ))}
          </div>
        </div>
      </CardContent>
    </Card>
  </motion.div>

  <motion.div
    initial={{ opacity: 0, y: 20 }}
    animate={{ opacity: 1, y: 0 }}
    transition={{ delay: 0.2, duration: 0.5 }}
    className="flex justify-end"
  >
    <motion.div whileHover={{ scale: 1.05 }} whileTap={{ scale: 0.95 }}>
      <Button onClick={submitAnswer} disabled={!selectedAnswer} className="group">
        {currentQuizIndex === quiz.length - 1 ? "Finish Quiz" : "Next Question"}
        <ArrowRight className="ml-2 h-4 w-4 group-hover:translate-x-1 transition-transform" />
      </Button>
    </motion.div>
  </motion.div>
```

```
        {/* Feedback overlay */}
        <AnimatePresence>
          {showFeedback && (
            <motion.div
              initial={{ opacity: 0 }}
              animate={{ opacity: 1 }}
              exit={{ opacity: 0 }}
              className="fixed inset-0 flex items-center justify-center z-50 bg-black/30"
            >
              <motion.div
                initial={{ scale: 0.5 }}
                animate={{ scale: 1.2 }}
                transition={{
                  type: "spring",
                  stiffness: 300,
                  damping: 10,
                }}
                className={text-4xl ${isCorrect ? "text-green-500" : "text-red-500"}}
              >
                {isCorrect ? (
                  <div className="flex flex-col items-center bg-white p-8 rounded-xl shadow-lg">
                    <CheckCircle className="h-16 w-16" />
                    <span className="text-lg font-bold mt-2">Correct!</span>
```

```
                  </div>
                ) : (
                  <div className="flex flex-col items-center bg-white p-8 rounded-xl shadow-lg">
                    <XCircle className="h-16 w-16" />
                    <span className="text-lg font-bold mt-2">Try Again!</span>
                  </div>
                )}
              </motion.div>
            </motion.div>
          )}
        </AnimatePresence>
      </div>
    )}

    {quizCompleted && (
      <motion.div
        initial={{ opacity: 0, y: 20 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ duration: 0.5 }}
        className="space-y-6"
      >
        <Card className="overflow-hidden">
          <CardContent className="pt-6 text-center relative">
            <div className="space-y-4">
              <motion.div
                initial={{ scale: 0.8 }}
```

```
          animate={{ scale: 1 }}
          transition={{
            type: "spring",
            stiffness: 300,
            damping: 10,
            delay: 0.2,
          }}
        >
          <h3 className="text-2xl font-bold">Quiz Completed!</h3>
          <div className="text-5xl font-bold mt-4 mb-2">
            {score} / {quiz.length}
          </div>
        </motion.div>

        <motion.div
          initial={{ opacity: 0 }}
          animate={{ opacity: 1 }}
          transition={{ delay: 0.5 }}
          className="flex justify-center"
        >
          <Badge variant="outline" className="bg-primary/10 text-primary text-lg px-3 py-1">
            <Trophy className="mr-2 h-5 w-5" />
            Total XP: {xp}
          </Badge>
        </motion.div>
```

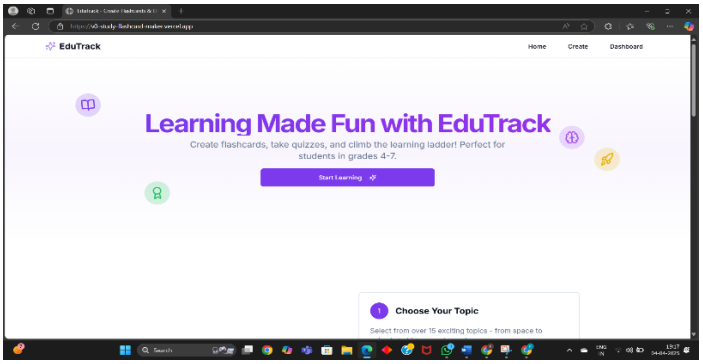
```
<motion.p
  initial={{ opacity: 0 }}
  animate={{ opacity: 1 }}
  transition={{ delay: 0.7 }}
  className="text-muted-foreground"
>
  {score === quiz.length
    ? "Perfect score! Excellent work! 🎉 "
    : score >= quiz.length / 2
      ? "Good job! Keep studying to improve. 👍 "
      : "Keep practicing to improve your score. 💪 "}
</motion.p>

<motion.div
  initial={{ opacity: 0, y: 20 }}
  animate={{ opacity: 1, y: 0 }}
  transition={{ delay: 0.9 }}
  className="pt-4 flex justify-center gap-4"
>
  <motion.div whileHover={{ scale: 1.05 }} whileTap={{ scale: 0.95 }}>
    <Button onClick={restartQuiz} variant="outline">
      Restart Quiz
    </Button>
  </motion.div>
  <motion.div whileHover={{ scale: 1.05 }} whileTap={{ scale: 0.95 }}>
```

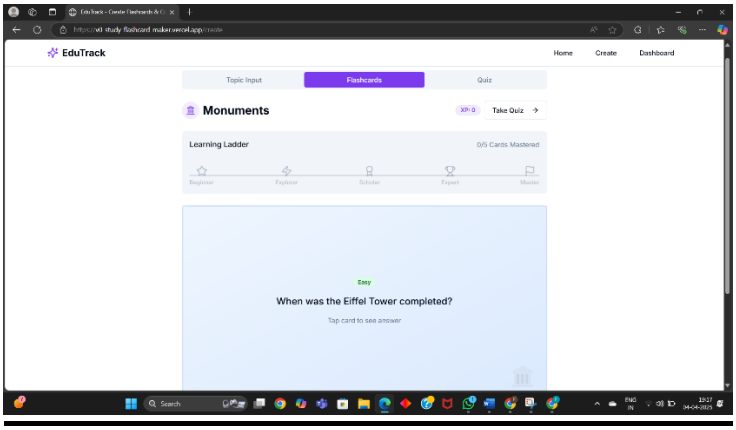
```
            <Button onClick={() => setActiveTab("flashcards")}>Back to Flashcards</Button>
          </motion.div>
          <motion.div whileHover={{ scale: 1.05 }} whileTap={{ scale: 0.95 }}>
            <Button asChild>
              <Link href="/dashboard">View Dashboard</Link>
            </Button>
          </motion.div>
        </motion.div>
      </div>

      {/* Animated stars */}
      {score === quiz.length && (
        <>
          <motion.div
            className="absolute top-10 left-10"
            animate={{
              rotate: 360,
              scale: [1, 1.2, 1],
            }}
            transition={{
              rotate: { repeat: Number.POSITIVE_INFINITY, duration: 10 },
              scale: { repeat: Number.POSITIVE_INFINITY, duration: 2 },
            }}
          >
            <Star className="h-8 w-8 text-yellow-400 fill-yellow-400" />
          </motion.div>
```

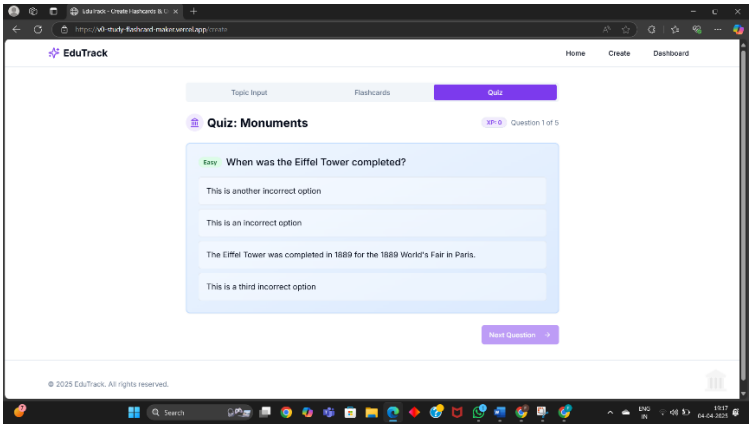
```
                <motion.div
                  className="absolute top-20 right-10"
                  animate={{
                    rotate: 360,
                    scale: [1, 1.2, 1],
                  }}
                  transition={{
                    rotate: { repeat: Number.POSITIVE_INFINITY, duration: 8, delay:
0.5 },
                    scale: { repeat: Number.POSITIVE_INFINITY, duration: 2, delay:
0.5 },
                  }}
                >
                  <Star className="h-6 w-6 text-yellow-400 fill-yellow-400" />
                </motion.div>
                <motion.div
                  className="absolute bottom-10 left-20"
                  animate={{
                    rotate: 360,
                    scale: [1, 1.2, 1],
                  }}
                  transition={{
                    rotate: { repeat: Number.POSITIVE_INFINITY, duration: 12, delay:
1 },
                    scale: { repeat: Number.POSITIVE_INFINITY, duration: 2, delay: 1
},
                  }}
```

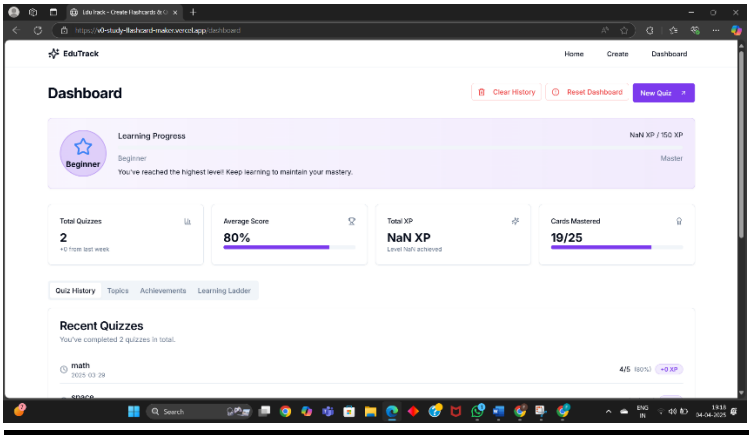
```
                >
                  <Star className="h-10 w-10 text-yellow-400 fill-yellow-400" />
                </motion.div>
              </>
            )}
          </CardContent>
        </Card>
      </motion.div>
    )}
  </TabsContent>
 </Tabs>
</main>
<footer className="border-t py-6 md:py-0">
 <div className="container flex flex-col items-center justify-between gap-4 md:h-24 md:flex-row">
  <p className="text-center text-sm leading-loose text-muted-foreground md:text-left">
    © 2025 EduTrack. All rights reserved.
  </p>
 </div>
</footer>
</div>
)
}
```

Result


EduTrack
Home
Create
Dashboard
Learning Made Fun with EduTrack
Create flashcards, take quizzes, and climb the learning ladder! Perfect for students in grades 4-7.
Start Learning
Choose Your Topic


EduTrack
Home
Create
Dashboard
Topic Input
Flashcards
Quiz
Monuments
Take Quiz
Learning Ladder
When was the Eiffel Tower completed?
Tap card to see answer


EduTrack
Home
Create
Dashboard
Topic Input
Flashcards
Quiz
Quiz: Monuments
When was the Eiffel Tower completed?
This is another incorrect option
This is an incorrect option
The Eiffel Tower was completed in 1889 for the 1889 World's Fair in Paris.
This is a third incorrect option
Next Question
© 2025 EduTrack. All rights reserved.


EduTrack
Home
Create
Dashboard
Dashboard
Clear History
Reset Dashboard
New Quiz
Learning Progress
Beginner
You've reached the highest level! Keep learning to maintain your mastery.
Total Quizzes
2
Average Score
80%
Total XP
NaN XP
Cards Mastered
19/25
Quiz History
Topics
Achievements
Learning Ladder
Recent Quizzes
You've completed 2 quizzes in total.
math

Team Contributions

Fiza Babariya

- Developed the flashcard generation module using Generative AI to create dynamic learning content.
- Designed and implemented the quiz system, ensuring AI-generated questions align with user progress.
- Worked on the backend logic for handling flashcards, quizzes, and data storage.
- Assisted in integrating real-time progress tracking and analytics for the dashboard.
- Conducted testing and debugging to enhance the overall accuracy and functionality of the platform.

Karan Soni

- Developed the dashboard interface, incorporating graphs and progress-tracking features.
- Implemented data visualization to display user performance trends effectively.
- Designed and integrated the history deletion functionality, allowing users to reset their learning progress.
- Focused on UI/UX design to ensure a smooth and interactive experience for users.
- Assisted in **frontend development** and optimized the platform for responsiveness and usability.

Conclusion

EduTrack successfully bridges the gap between traditional study tools and modern digital education platforms. It makes learning more effective, engaging, and personalized. With features like dynamic flashcard generation, gamified quiz mechanics, and real-time performance tracking, EduTrack motivates users to stay consistent and improve continuously.

As a modular, responsive, and scalable platform, EduTrack is well-positioned for further growth. Future iterations will include AI-generated content, collaborative learning modes, and advanced analytics dashboards. With its intuitive design and robust architecture, EduTrack is more than a tool – it's a companion in every student's learning journey.

Future Scope:

- Role-Based Access – Introduce student, parent, teacher, and admin roles, allowing teachers to assign quizzes, parents to track progress, and admins to manage users.
- AI-Driven Personalization – Implement AI tutors, adaptive learning paths, and smart recommendations to enhance individualized learning.
- Voice & Chatbot Interaction – Develop an AI chatbot and voice-based assistant for real-time learning support.
- Advanced Analytics & Reports – Provide detailed performance insights for students, teachers, and parents using AI-driven analytics.
- Integration & Scalability – Expand to multiple subjects, multilingual support, and LMS integration for broader accessibility.