Ø<ß® SkillSprout

Code Documentation

Educational Puzzle Game for Kids

Ages 3-6

Technology Stack:

Next.js 16
React 19
TypeScript 5
Tailwind CSS 4

Generated: 12/16/2025

Ø=ÜË Project Overview

About SkillSprout

SkillSprout is an interactive educational puzzle game designed for young children aged 3-6 years. The application provides a fun and engaging way for kids to develop cognitive skills through pattern recognition, shape matching, and logical thinking exercises. Built with modern web technologies, it offers a responsive, child-friendly interface with role-based authentication for administrators, teachers, and parents.

Key Features

The application includes three main puzzle categories: Patterns, Shape Match, and Logical Thinking. Each category contains age-appropriate puzzles with difficulty levels ranging from 3-4 years to 5-6 years. The system includes real-time scoring, helpful hints, progress tracking, and a beautiful gradient-based UI designed to appeal to young learners.

Architecture Highlights

Built on Next.js 16 with the App Router, the application uses React Server Components and Client Components strategically. Authentication is managed through React Context with localStorage persistence. The puzzle data is stored in JSON files and filtered dynamically based on user-selected age ranges. Protected routes ensure that only authenticated users can access the game content.

Ø=Ý Authentication System

Overview

The authentication system is built using React Context API, providing a centralized state management solution for user authentication across the application. It supports three user roles: Administrator, Teacher, and Parent, each with demo credentials for easy testing.

AuthContext Implementation

User types and demo credentials stored in the AuthContext

```
interface User {
  username: string;
  role: string;
}

const validUsers = [
  { username: 'admin', password: 'admin123', role: 'Administrator' },
  { username: 'teacher', password: 'teacher123', role: 'Teacher' },
  { username: 'parent', password: 'parent123', role: 'Parent' },
];
```

Login Function

Simple but effective authentication with localStorage persistence

```
const login = (username: string, password: string): boolean => {
  const foundUser = validUsers.find(
    (u) => u.username === username && u.password === password
  );

  if (foundUser) {
    const userData = { username: foundUser.username, role: foundUser.role };
    setUser(userData);
    localStorage.setItem('user', JSON.stringify(userData));
    return true;
  }
  return false;
};
```

Key Features

The auth system includes automatic redirect after login, localStorage persistence for session management, and a loading state to prevent flash of unauthenticated content.

Ø<ß¨ Login Page Design

Ø=Üñ Login Page (/login)

A vibrant, child-friendly login interface with a beautiful gradient background transitioning from yellow through pink to purple. The page features the SkillSprout branding with a game controller emoji and welcoming text.

Key Features:

- **Gradient background: from-yellow-100 via-pink-100 to-purple-100**
- **Centered white card with rounded corners (rounded-3xl) and shadow-2xl**
- **Username and password input fields with purple focus states**
- **Colorful gradient button (purple-500 to pink-500) with hover effects**
- **Toggleable demo credentials section showing all test accounts**
- **Responsive design with mobile-first approach (sm:, md:, lg: breakpoints)**
- **Error messaging with red background for invalid credentials**
- **Auto-redirect to home page when already authenticated**

Gradient Button Styling

Modern Tailwind CSS styling with hover animations

```
className="w-full bg-gradient-to-r from-purple-500 to-pink-500
  text-white font-bold text-xl py-4 rounded-xl
  hover:shadow-lg transform hover:scale-105
  active:scale-95 transition-all"
```

Form Validation

The login form includes client-side validation to ensure both username and password are provided before submission. Error messages are displayed in a prominent red box above the login button, providing clear feedback to users.

Ø=Þáþ Protected Route System

Route Protection

The ProtectedRoute component ensures that only authenticated users can access puzzle content. It wraps protected pages and automatically redirects unauthenticated users to the login page.

ProtectedRoute Component

File: src/components/ProtectedRoute.tsx

```
export default function ProtectedRoute({ children }: { children: React.ReactNode }) {
  const { isAuthenticated } = useAuth();
  const router = useRouter();

  useEffect(() => {
    if (!isAuthenticated) {
      router.push('/login');
    }
  }, [isAuthenticated, router]);

  if (!isAuthenticated) {
    return (
      <div className="min-h-screen flex items-center justify-center">
        <div className="text-2xl font-bold text-purple-600">
          Redirecting to login...
        </div>
      </div>
    );
  }

  return <>{children}</>;
}
```

Usage

Every protected page wraps its content with the ProtectedRoute component. This includes the home page and all puzzle category pages, ensuring a consistent authentication barrier across the application.

Ø<ßà Home Page & Categories

Ø=Üñ Home Page (Main Dashboard)

The main dashboard welcomes authenticated users and allows them to select their age range and choose from three puzzle categories. It features a clean, colorful design with large, touch-friendly buttons perfect for young children.

Key Features:

- **User welcome banner showing username and role**
- **Logout button in the top-right corner**
- **Age selection buttons (3-4, 4-5, 5-6 years) with active state highlighting**
- **Three category cards: Patterns (Ø=Ý), Shape Match (+P), and Logical Thinking (Ø>Ýà)**
- **Each category has a unique gradient color scheme**
- **Grid layout that adapts from single column on mobile to 3 columns on large screens**
- **Hover effects with scale transform for interactive feedback**
- **Dynamic routing with age parameter passed to puzzle pages**

Category Configuration

File: src/app/page.tsx

```
const categories = [
  {
    id: 'patterns',
    name: 'Patterns',
    emoji: 'Ø=Ý ',
    description: 'Complete the pattern!',
    color: 'from-purple-400 to-pink-400'
  },
  {
    id: 'shapes_match',
    name: 'Shape Match',
    emoji: '+P',
    description: 'Find the right shape!',
    color: 'from-blue-400 to-cyan-400'
  },
  {
    id: 'logical',
    name: 'Logical Thinking',
    emoji: 'Ø>Ýà',
    description: 'Use your brain!',
    color: 'from-green-400 to-emerald-400'
  }
];
```

Ø=ÜÊ Puzzle Data Structure

Data Organization

Puzzles are organized in three JSON files (patterns.json, shapes_match.json, logical.json), each containing an array of puzzle objects. Each puzzle includes metadata for age-appropriate filtering and gameplay.

Puzzle Interface

File: src/types/puzzle.ts

```
export interface Puzzle {
  id: string;
  ageRange: AgeRange;
  title: string;
  question: string;
  options: string[];
  correctAnswer: number;
  hint?: string;
}

export type AgeRange = "3-4" | "4-5" | "5-6" | "3-5" | "4-6";
```

Sample Puzzle Data

Example from src/data/patterns.json - Uses emojis for visual appeal

```
{
  "id": "pattern_1",
  "ageRange": "3-4",
  "title": "Color Pattern",
  "question": "Ø=Ý4 Ø=ßâ Ø=Ý4 Ø=ßâ ... ?",
  "options": ["Ø=Ý4", "Ø=ßâ", "Ø=Ý5"],
  "correctAnswer": 0,
  "hint": "It repeats red, green, red, green..."
}
```

Age-Based Filtering

The application dynamically filters puzzles based on the selected age range. The filtering algorithm compares the selected age range with each puzzle's age range to ensure children only see age-appropriate content.

Ø<ß® Puzzle Game Engine

Game State Management

The puzzle page manages multiple state variables to track the current puzzle, user selections, score, and UI states. This state-driven approach ensures a smooth, interactive gaming experience.

State Variables

File: src/app/puzzle/[category]/page.tsx - Core game state

```
const [currentIndex, setCurrentIndex] = useState(0);
const [selectedAnswer, setSelectedAnswer] = useState<number | null>(null);
const [showResult, setShowResult] = useState(false);
const [score, setScore] = useState(0);
const [showHint, setShowHint] = useState(false);
const [filteredPuzzles, setFilteredPuzzles] = useState<Puzzle[]>([]);
```

Answer Handler

Validates answer and updates score immediately

```
const handleAnswer = (index: number) => {
  if (showResult) return;
  setSelectedAnswer(index);
  setShowResult(true);
  if (index === currentPuzzle.correctAnswer) {
    setScore(score + 1);
  }
};
```

Navigation Logic

The game includes smart navigation with Next and Restart functions. When reaching the last puzzle, users are presented with their final score and options to play again or choose a new category.

Next Puzzle Handler

Resets state and advances to next puzzle

```
const handleNext = () => {
  if (currentIndex < filteredPuzzles.length - 1) {
    setCurrentIndex(currentIndex + 1);
    setSelectedAnswer(null);
    setShowResult(false);
    setShowHint(false);
  }
};
```

Ø<ß¨ UI Components & Styling

Ø=Üñ Puzzle Page Interface

The puzzle page presents an engaging, colorful interface where children can interact with puzzles through large, touch-friendly buttons and receive immediate visual feedback.

Key Features:

- **Header with Home button, Score display, and Logout button**
- **Category banner with gradient background matching category theme**
- **White card containing the puzzle question and options**
- **Large option buttons (min-height: 120-140px) for easy selection**
- **Visual feedback: Green for correct, Red for incorrect answers**
- **Animated emoji (Ø<ß‰ or Ø=Üª) shown after answer selection**
- **Hint button with toggleable yellow box for assistance**
- **Progress bar at bottom showing completion percentage**
- **Responsive grid layout for options (1 column mobile, 3 columns desktop)**

Design Philosophy

The entire application follows a consistent design language with rounded corners, soft shadows, gradient backgrounds, and playful emoji icons. The color scheme uses purple, pink, blue, and cyan gradients to create a cheerful, engaging atmosphere suitable for young children.

Responsive Option Buttons

Fully responsive with mobile-first approach

```
className="p-5 sm:p-6 md:p-7 lg:p-8
  rounded-xl sm:rounded-2xl
  text-2xl sm:text-3xl md:text-4xl lg:text-xl
  font-bold transition-all transform
  active:scale-95 sm:hover:scale-105
  min-h-[100px] sm:min-h-[120px]
  md:min-h-[130px] lg:min-h-[140px]"
```

&¡ Key Technical Highlights

1. Next.js App Router

Utilizes Next.js 16 App Router with dynamic routes for category pages ([category]/page.tsx). The use() hook unwraps params promises for async route parameters.

2. Client-Side State Management

React Context API provides global auth state without external dependencies. LocalStorage ensures session persistence across browser refreshes.

3. TypeScript Integration

Full TypeScript implementation with custom interfaces for User, Puzzle, Category, and AgeRange types, providing type safety throughout the application.

4. Data-Driven Architecture

JSON-based puzzle storage allows easy content updates without code changes. The filtering system dynamically adjusts content based on user selections.

5. Responsive Design

Tailwind CSS with comprehensive breakpoints (sm, md, lg, xl) ensures perfect rendering on all devices from mobile phones to desktop monitors.

6. Progressive Enhancement

Graceful degradation with fallback states, loading indicators, and error boundaries. The UI remains functional even if JavaScript features are limited.

- File Structure:
  - src/app/ - Next.js pages and layouts
  - src/components/ - Reusable React components
  - src/contexts/ - React Context providers
  - src/data/ - JSON puzzle data files
  - src/types/ - TypeScript type definitions

Deployment

The application is deployed on Vercel at puzzle-game-vibe-coding.vercel.app, leveraging Vercel's edge network for fast global performance and automatic HTTPS.

% %

End of Documentation

Ø<ß® SkillSprout - Making Learning Fun!