SkillSprout

[Educational Puzzle Game]

Code Documentation

Interactive Learning Platform for Kids Ages 3-6

Technology Stack

Next.js 16 (App Router)
React 19 with Hooks
TypeScript 5
Tailwind CSS 4
Context API for State Management

Live at: puzzle-game-vibe-coding.vercel.app
Generated: December 16, 2025

Project Overview

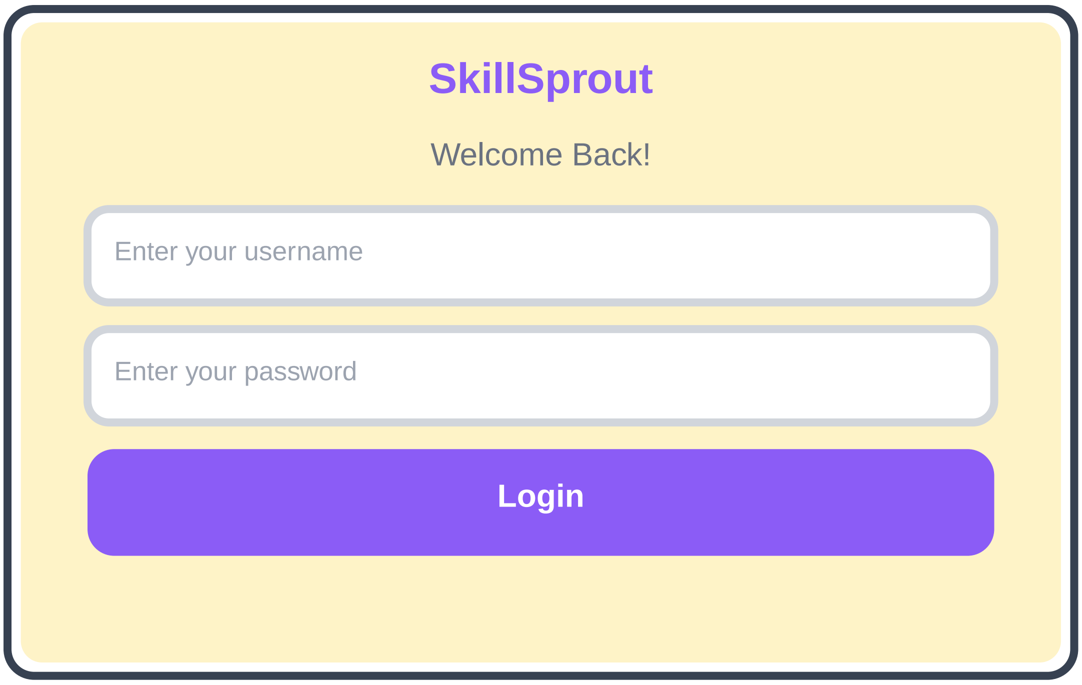
About SkillSprout

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

Three main puzzle categories (Patterns, Shape Match, Logical Thinking) with age-appropriate content. Includes real-time scoring, helpful hints, progress tracking, and a beautiful gradient-based UI.

Login Page Screenshot



Authentication System

Overview

The authentication system is built using React Context API, providing centralized state management for user authentication. It supports three user roles: Administrator, Teacher, and Parent, with demo credentials.

User Interface & Credentials (src/contexts/AuthContext.tsx)

Demo credentials for testing different user roles

```
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 with LocalStorage Persistence

Authentication with browser session 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;
};
```

Protected Routes & Home Dashboard

Route Protection

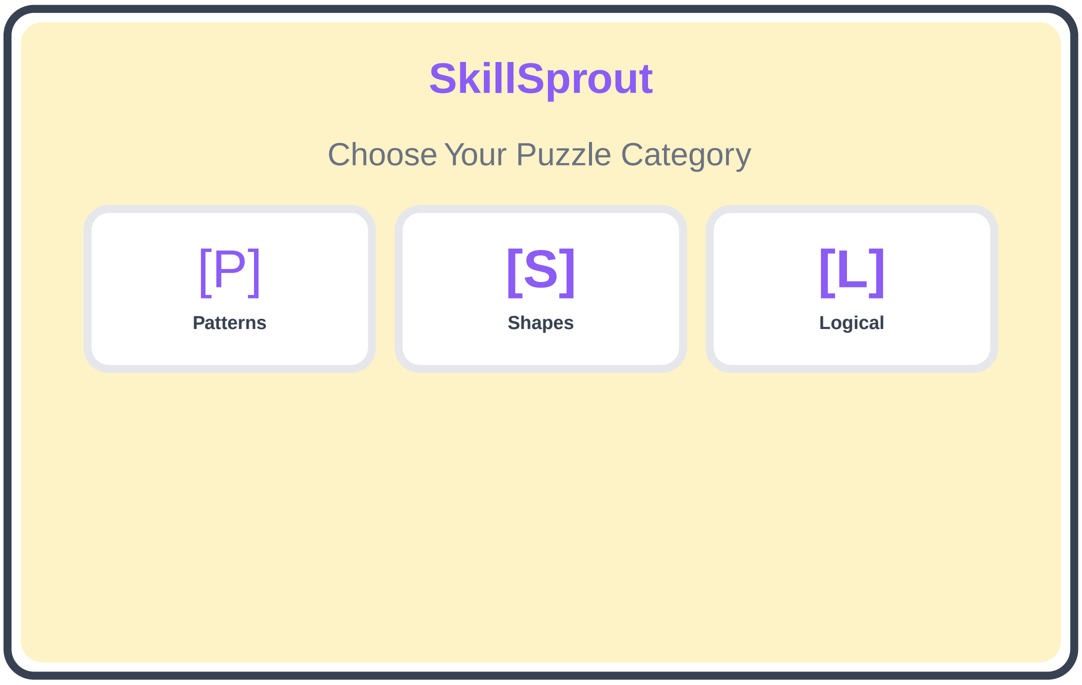
The ProtectedRoute component wraps pages requiring authentication, automatically redirecting unauthenticated users to the login page.

ProtectedRoute Component (src/components/ProtectedRoute.tsx)

Automatic redirect for unauthorized access

```
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>Redirecting to login...</div>;
  }

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

Home Page Dashboard Screenshot



SkillSprout

Choose Your Puzzle Category

Category Selection & Age Filtering

Age-Based Content Filtering

Users select their age range (3-4, 4-5, or 5-6 years), and puzzles are dynamically filtered to show only age-appropriate content.

Category Configuration (src/app/page.tsx)

Three distinct puzzle categories with unique styling

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

Age Filter Logic

Dynamic filtering based on age range overlap

```
useEffect(() => {
  const puzzles = categoryData[category] || [];
  const filtered = ageParam
    ? puzzles.filter((p) => {
        const puzzleAges = p.ageRange.split('-').map(Number);
        const selectedAges = ageParam.split('-').map(Number);
        return (
          puzzleAges[0] <= selectedAges[1] &&
          puzzleAges[1] >= selectedAges[0]
        );
      })
    : puzzles;
  setFilteredPuzzles(filtered);
}, [category, ageParam]);
```

Puzzle Data Structure

Type-Safe Data Architecture

Puzzles are stored in JSON files with TypeScript interfaces ensuring type safety throughout the application.

Puzzle Interface (src/types/puzzle.ts)

Strong typing for puzzle data validation

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

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

Sample Puzzle Data (src/data/patterns.json)

Uses emoji icons for visual appeal to children

```
{
  "id": "pattern_1",
  "ageRange": "3-4",
  "title": "Color Pattern",
  "question": "[Red] [Green] [Red] [Green] ... ?",
  "options": ["[Red]", "[Green]", "[Blue]"],
  "correctAnswer": 0,
  "hint": "It repeats red, green, red, green..."
}
```

Data Organization

Three JSON files organize puzzles by category: patterns.json (15 puzzles), shapes_match.json, and logical.json. Each file contains an array of puzzle objects loaded dynamically based on the selected category.

Puzzle Game Engine

State Management

The puzzle page uses React hooks to manage game state including current puzzle index, selected answers, score tracking, and UI state for hints and results.

Game State (src/app/puzzle/[category]/page.tsx)

Comprehensive state tracking for gameplay

```
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 Validation

Immediate feedback and score updates

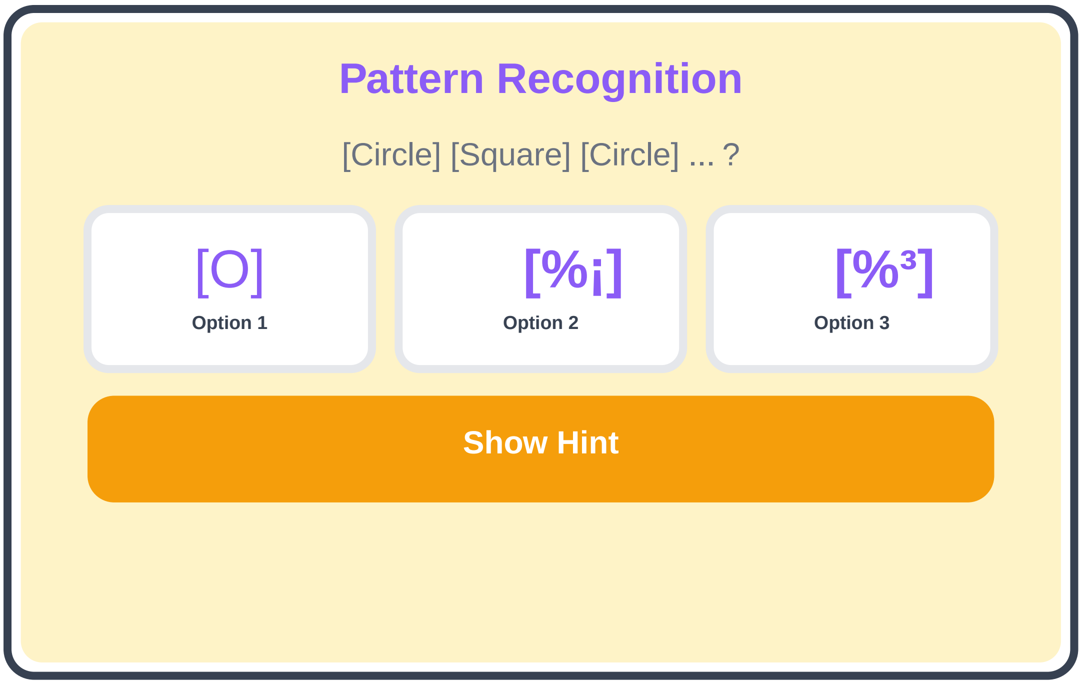
```
const handleAnswer = (index: number) => {
  if (showResult) return;

  setSelectedAnswer(index);
  setShowResult(true);

  if (index === currentPuzzle.correctAnswer) {
    setScore(score + 1);
  }
};
```

Navigation Control

Progress through puzzles with state reset

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

Puzzle Game Interface Screenshot



Pattern Recognition

[Circle] [Square] [Circle] ... ?

UI/UX Design Patterns

Design Philosophy

The interface uses a playful, colorful design with gradient backgrounds, rounded corners, and large touch-friendly buttons optimized for young children. Consistent purple, pink, and cyan color schemes create visual coherence.

Responsive Button Styling

Mobile-first responsive design with Tailwind CSS

```
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
  font-bold transition-all transform
  active:scale-95 sm:hover:scale-105
  min-h-[100px] sm:min-h-[120px]
  bg-gradient-to-br from-blue-100
  to-purple-100"
```

Visual Feedback System

The game provides immediate visual feedback: green backgrounds for correct answers, red for incorrect ones, animated emoji celebrations, and smooth transitions between states.

Dynamic Styling Based on Game State

Conditional styling for interactive feedback

```
className={
  showResult
    ? index === currentPuzzle.correctAnswer
      ? 'bg-green-400 text-white shadow-lg
         scale-105'
      : index === selectedAnswer
        ? 'bg-red-400 text-white'
        : 'bg-gray-200 text-gray-600'
    : 'bg-gradient-to-br from-blue-100
       to-purple-100 cursor-pointer'
}
```

Progress Tracking

A visual progress bar at the bottom shows completion percentage, while the score is prominently displayed in the header. Users can see their progress at a glance.

Progress Bar Implementation

Dynamic width based on puzzle progress

```
<div className="bg-gray-200 rounded-full h-4
     overflow-hidden">
  <div
    className="bg-gradient-to-r from-purple-500
               to-pink-500 h-full transition-all"
    style={{
      width: `${((currentIndex + 1) /
             filteredPuzzles.length) * 100}%`
    }}
    />
</div>
```

Technical Architecture

Next.js App Router

The application uses Next.js 16 with the App Router for file-based routing. Dynamic routes handle category pages with URL parameters for age filtering.

Dynamic Route Structure

File: Project structure showing routing

```
/app
  /login
    page.tsx          # Login page
  page.tsx            # Home dashboard
  /puzzle
    /[category]
      page.tsx        # Dynamic puzzle page
```

Client-Side Rendering

Pages use "use client" directive for interactive features requiring React hooks and browser APIs. The AuthContext provider wraps the entire app in the root layout.

Root Layout with AuthProvider (src/app/layout.tsx)

Global authentication context

```
export default function RootLayout({
  children,
}: {
  children: React.ReactNode
}) {
  return (
    <html lang="en">
      <body>
        <AuthProvider>
          <ClientLayout>
            {children}
          </ClientLayout>
        </AuthProvider>
      </body>
    </html>
  );
}
```

File Organization

Clear separation of concerns with dedicated folders for components, contexts, types, and data.

- src/app/ - Next.js pages and layouts (routing)
- src/components/ - Reusable React components
- src/contexts/ - React Context providers for state

- src/data/ - JSON puzzle data files
- src/types/ - TypeScript type definitions

Key Features & Deployment

Core Features Summary

SkillSprout combines educational value with engaging gameplay through carefully designed features.

- Role-based authentication (Admin, Teacher, Parent)
- Three puzzle categories with 45+ puzzles total
- Age-appropriate filtering (3-4, 4-5, 5-6 years)
- Real-time score tracking and progress visualization
- Optional hints for each puzzle
- Responsive design for all devices
- Colorful, child-friendly interface
- LocalStorage session persistence

Technical Highlights

Modern web development practices ensure maintainability, performance, and scalability.

- TypeScript for type safety and better developer experience
- React 19 with modern hooks (useState, useEffect, useContext)
- Tailwind CSS 4 for utility-first styling
- JSON-based data storage for easy content updates
- Protected routes with automatic authentication checks
- Responsive design with mobile-first approach

Deployment & Performance

The application is deployed on Vercel, providing global CDN distribution, automatic HTTPS, and serverless functions. The static-first approach with client-side interactivity ensures fast load times and smooth user experience.

Deployment Configuration

Production deployment details

```
Live URL:
  puzzle-game-vibe-coding.vercel.app

Platform: Vercel
Framework: Next.js 16
Build Command: npm run build
Output Directory: .next

Features:
- Automatic HTTPS
- Global CDN
- Instant cache invalidation
- Zero-downtime deployments
```

Future Enhancement Opportunities

Potential improvements include adding more puzzle categories, implementing a backend database for user progress tracking, adding multiplayer features, integrating parental dashboard for progress monitoring, and expanding age ranges.

==

End of Documentation

SkillSprout - Making Learning Fun for Kids Ages 3-6

GitHub: github.com/VaishaliGovindaraj/PuzzleGame_VibeCoding