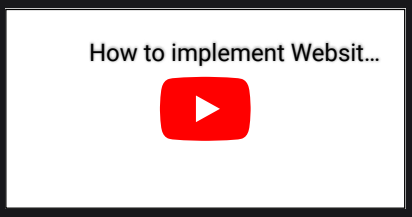
Embracing Faith Through Fun: The Impact of Christian Games and Puzzles edited

engaging activities are more than just pastimes; they are powerful tools designed to deepen one's understanding of scripture, foster community, and nurture spiritual development.

This is an heading

In a world where entertainment often seems detached from spiritual growth, Christian games and puzzles offer a refreshing way to combine fun with faith. These engaging activities are more than just pastimes; they are powerful tools designed to deepen one's understanding of scripture, foster community, and nurture spiritual development.



Christian games and puzzles come in various forms, each tailored to convey biblical lessons in an interactive and enjoyable manner. From board games that bring Bible stories to life to intricate puzzles that feature scriptural themes, these resources are crafted to make learning about faith both accessible and enjoyable.

This is another heading

One of the most significant benefits of Christian games and puzzles is their ability to make scripture memorable. By engaging with Bible verses through interactive play, players are more likely to internalize and reflect on spiritual teachings. For instance, a game that challenges players to answer questions about Bible characters or events not only tests their knowledge but also reinforces their understanding of the material.

```
"use client";


import React, { useState, useEffect, useRef } from "react";
import { Share2, Eye, Menu, Search, ArrowUpRight } from "lucide-react";
import Image from "next/image";
import { Button } from "@/components/ui/button";
import {
  Sheet,
  SheetContent,
  SheetDescription,
  SheetHeader,
  SheetTitle,
  SheetTrigger
} from "@/components/ui/sheet";
```

```tsx
import { usePathname } from "next/navigation";
import { useQuery } from "@tanstack/react-query";
import axios from "axios";
import { cn } from "@/lib/utils";
import "highlight.js/styles/atom-one-dark.css";
import "react-quill/dist/quill.snow.css";


import { createRoot } from "react-dom/client";
import { Copy } from "lucide-react";
import hljs from "highlight.js";


const CodeBlockNav = () => (
  <div className="flex items-center justify-between bg-zinc-700 px-4 py-2 text-sm text-gray-200 rounded-t-md">
    {/* <span>{language}</span> */}
    <button className="hover:text-white">
      <Copy size={16} />
    </button>
  </div>
);


interface HeadingObject {
  id: string;
  text: string;
  level: number;
  items: HeadingObject[];
}


interface Article {
  id: string;
  title: string;
  description: string;
  blurImage: string;
  imageUrl: string;
  shortSummary: string;
  pdfUrl: string;
}


interface ApiResponse {
  article: Article;
  relatedArticles: Article[];
}


const ArticleLayout: React.FC = () => {
  const [activeSection, setActiveSection] = useState<string>("");
  const [isDrawerOpen, setIsDrawerOpen] = useState<boolean>(false);
  const [articleStructure, setArticleStructure] = useState<HeadingObject[]>([]);
  const contentRef = useRef<HTMLDivElement>(null);
  const articleContentRef = useRef<HTMLDivElement>(null);
  const [articleHtml, setArticleHtml] = useState<string>("");


  const pathname = usePathname();
  const id = pathname.split("/").pop() || "";


  const { isPending, error, data } = useQuery<ApiResponse, Error>({
    queryKey: [id],
    queryFn: () =>
      axios(`/api/routes/fetchSingleArticle?articleId=${id}`).then(
        (res) => res.data
      )
  });
```

```
const article = data?.article;
const handleDownload = async (pdfUrl: any) => {
  console.log(pdfUrl, "pdfurl.....................");
  try {
    const response = await fetch(pdfUrl);
    const blob = await response.blob();
    const url = window.URL.createObjectURL(blob);
    const link = document.createElement("a");
    link.href = url;
    link.download = "Christian-Games-and-Puzzles.pdf";
    document.body.appendChild(link);
    link.click();
    document.body.removeChild(link);
    window.URL.revokeObjectURL(url);
  } catch (error) {
    console.error("Error downloading PDF:", error);
    // You might want to show an error message to the user here
  }
};
// Modify code blocks and other elements


const processArticleContent = (htmlContent: string) => {
  const parser = new DOMParser();
  const doc = parser.parseFromString(htmlContent, "text/html");


  const codeBlocks = doc.querySelectorAll("pre");


  codeBlocks.forEach((codeBlock) => {
    // Create wrapper
    const wrapper = document.createElement("div");
    wrapper.className = "my-4 rounded-lg overflow-hidden";


    // Navbar for the code block
    const navbar = document.createElement("div");
    navbar.className =
      "bg-zinc-800 px-4 py-2 flex justify-between items-center";


    const languageSpan = document.createElement("span");
    languageSpan.className = "text-zinc-400";


    // Convert codeBlock content to string for highlighting
    const codeContent = codeBlock.textContent || "";
    const result = hljs.highlightAuto(codeContent);
    const language = result.language || "plaintext";
    languageSpan.textContent = language;
    navbar.appendChild(languageSpan);


    const copyButton = document.createElement("button");
    copyButton.className =
      "copy-button text-zinc-400 hover:text-white flex items-center";
    copyButton.innerHTML = `
      <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" viewBox="0 0 24 24" fill="none" stroke=
        <rect x="9" y="9" width="13" height="13" rx="2" ry="2"></rect>
        <path d="M5 15H4a2 2 0 0 1-2-2V4a2 2 0 0 1 2-2h9a2 2 0 0 1 2 2v1"></path>
      </svg>
      <span className="text-sm">Copy</span>
    `;


    navbar.appendChild(copyButton);


    // Insert the navbar before the code block
```

```
    codeBlock.parentNode?.insertBefore(wrapper, codeBlock);

    // Move the code block into the wrapper
    wrapper.appendChild(navbar);
    wrapper.appendChild(codeBlock);
  });

  // Add event listener to the document body for event delegation
  document.body.addEventListener("click", function (event) {
    const target = event.target as Element;
    if (target.closest(".copy-button")) {
      const copyButton = target.closest(".copy-button") as HTMLElement;
      const codeBlock = copyButton.closest(".my-4")?.querySelector("pre");
      if (codeBlock) {
        const codeText = codeBlock.textContent || "";
        navigator.clipboard
          .writeText(codeText)
          .then(() => {
            console.log("Text copied to clipboard");
            copyButton.innerHTML = `
             <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" viewBox="0 0 24 24" fill="none"
               <polyline points="20 6 9 17 4 12"></polyline>
              </svg>
              <span className="text-sm">Copied!</span>
            `;
            setTimeout(() => {
              copyButton.innerHTML = `
                <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" viewBox="0 0 24 24" fill="non
                  <rect x="9" y="9" width="13" height="13" rx="2" ry="2"></rect>
                  <path d="M5 15H4a2 2 0 0 1-2-2V4a2 2 0 0 1 2-2h9a2 2 0 0 1 2 2v1"></path>
                </svg>
                <span className="text-sm" >Copy</span>
              `;
            }, 2000);
          })
          .catch((err) => {
            console.error("Failed to copy text: ", err);
          });
      }
    }
  });

  return doc.body.innerHTML; // Return the processed HTML
};

useEffect(() => {
  if (article?.description && articleContentRef.current) {
    articleContentRef.current.innerHTML = articleHtml;
    const headings = articleContentRef.current.querySelectorAll("h1, h2, h3");

    const structure: HeadingObject[] = [
      { id: "article-title", text: article.title, level: 1, items: [] }
    ];
    let currentH1: HeadingObject | null = null;
    let currentH2: HeadingObject | null = null;

    headings.forEach((heading, index) => {
      const headingId = heading.id || `heading-${index}`;
      heading.id = headingId;

      const headingObject: HeadingObject = {
        id: headingId,
        text: heading.textContent || "",
```

```
        level: parseInt(heading.tagName[1]),
        items: []
      };


      switch (heading.tagName.toLowerCase()) {
        case "h1":
          currentH1 = headingObject;
          currentH2 = null;
          structure.push(currentH1);
          break;
        case "h2":
          currentH2 = headingObject;
          if (currentH1) {
            currentH1.items.push(currentH2);
          } else {
            structure.push(currentH2);
          }
          break;
        case "h3":
          if (currentH2) {
            currentH2.items.push(headingObject);
          } else if (currentH1) {
            currentH1.items.push(headingObject);
          } else {
            structure.push(headingObject);
          }
          break;
      }
    });
    const processedContent = processArticleContent(article.description);
    setArticleHtml(processedContent);
    setArticleStructure(structure);
  }
}, [article, articleHtml]);


useEffect(() => {
  const handleScroll = () => {
    const headings =
      articleContentRef.current?.querySelectorAll("h1, h2, h3");
    if (headings) {
      for (let i = headings.length - 1; i >= 0; i--) {
        const heading = headings[i];
        const rect = heading.getBoundingClientRect();
        if (rect.top <= 100) {
          setActiveSection(heading.id);
          break;
        }
      }
    }
  };


  window.addEventListener("scroll", handleScroll);
  return () => window.removeEventListener("scroll", handleScroll);
}, []);


const scrollToSection = (id: string) => {
  const element = document.getElementById(id);
  if (element) {
    const navbarHeight = 60; // Height of the main navbar
    const secondNavbarHeight = 50; // Height of the second navbar
    const padding = 20; // Additional padding
    const windowWidth = window.innerWidth;


    let offsetPosition;
    if (windowWidth < 1024) {
```

```
      // For screens smaller than 1024px, account for both navbars
      offsetPosition =
        element.getBoundingClientRect().top +
        window.pageYOffset -
        navbarHeight -
        secondNavbarHeight -
        padding;
    } else {
      // For larger screens, only account for the main navbar
      offsetPosition =
        element.getBoundingClientRect().top +
        window.pageYOffset -
        navbarHeight -
        padding;
    }


    window.scrollTo({
      top: offsetPosition,
      behavior: "smooth"
    });


    setActiveSection(id);
  }
  setIsDrawerOpen(false);
};


const LeftSidebar: React.FC = () => {
  const [hoveredSection, setHoveredSection] = useState<string | null>(null);
  const renderSidebarItems = (items: HeadingObject[]) => {
    return items.map((item, index) => (
      <div key={index} className={`mb-2 ${item.level === 2 ? "mt-4" : ""}`}>
        <a
          href={`#${item.id}`}
          className={`block py-1 text-sm transition-colors duration-150 ease-in-out ${
            activeSection === item.id
              ? "text-[#4b9bff] font-semibold"
              : hoveredSection === item.id
              ? "text-white"
              : "text-gray-400"
          }`}
          onClick={(e) => {
            e.preventDefault();
            scrollToSection(item.id);
          }}
          onMouseEnter={() => setHoveredSection(item.id)}
          onMouseLeave={() => setHoveredSection(null)}
        >
          {item.text}
        </a>
        {item.items && item.items.length > 0 && (
          <div className={`relative ${item.level === 2 ? "ml-0h mt-2" : ""}`}>
            {item.level === 2 && (
              <div className="absolute left-0 top-0 bottom-0 w-px bg-zinc-700" />
            )}
            {item.items.map((subItem, subIndex) => (
              <div key={subIndex} className="relative">
                {item.level === 2 && (
                  <div
                    className={cn(
                      activeSection === subItem.id
                        ? "absolute left-0 top-0 bottom-0 w-px bg-[#4b9bff] "
                        : hoveredSection === subItem.id
                        ? "absolute left-0 top-0 bottom-0 w-px bg-white"
                        : ""
                    )}
                  />
                )}
```

```
                <a
                  href={`#${subItem.id}`}
                  className={`block py-1 pl-4 text-sm transition-colors duration-150 ease-in-out ${
                    activeSection === subItem.id
                      ? "text-[#4b9bff] font-semibold"
                      : hoveredSection === subItem.id
                      ? "text-white"
                      : "text-gray-400"
                  }`}
                  onClick={(e) => {
                    e.preventDefault();
                    scrollToSection(subItem.id);
                  }}
                  onMouseEnter={() => setHoveredSection(subItem.id)}
                  onMouseLeave={() => setHoveredSection(null)}
                >
                  {subItem.text}
                </a>
              </div>
            ))}
          </div>
        )}
      </div>
    ));
  };


  return (
    <div className="bg-zinc-800/50 backdrop-blur-sm text-gray-400 rounded-lg">
      <div className="p-4">
        <div className="relative">
          <input
            type="text"
            placeholder="Quick search..."
            className="w-full py-2 pl-8 pr-3 text-sm bg-[rgba(32,33,39,0.8)] text-white rounded-lg border bo
          />
          <Search
            className="absolute left-2 top-1/2 transform -translate-y-1/2 text-[#a1a1aa]"
            size={16}
          />
          <span className="absolute right-2 top-1/2 transform -translate-y-1/2 text-[#a1a1aa] text-xs">
            Ctrl K
          </span>
        </div>
      </div>
      <nav className="px-4 mt-4 h-full overflow-y-auto">
        {renderSidebarItems(articleStructure)}
      </nav>
    </div>
  );
};


// ... (rest of the component remains the same)


return (
  <div className="min-h-screen bg-zinc-900 text-white">
    {/* Mobile Navigation */}
    <div className="mobile lg:hidden fixed top-0 left-0 right-0 pt-20 z-50 bg-zinc-900/95 backdrop-blur-sm p
      <Sheet open={isDrawerOpen} onOpenChange={setIsDrawerOpen}>
        <SheetTrigger asChild>
          <Button variant="outline" size="icon" className="bg-zinc-800">
            <Menu className="h-4 w-4" />
          </Button>
        </SheetTrigger>
        <SheetContent
          side="left"
          className="bg-zinc-900 z-[1000] border-zinc-800 overflow-y-auto"
        >
```

```
        <SheetHeader>
          <SheetTitle className="text-white">Table of Contents</SheetTitle>
          <SheetDescription className="text-gray-400">
            Navigate through the article sections
          </SheetDescription>
        </SheetHeader>
        <div className="mt-4 h-[calc(100vh-120px)]">
          <LeftSidebar />
        </div>
      </SheetContent>
    </Sheet>
    <div className="relative flex-1 max-w-[400px] ml-4">
      <div className="relative">
        <input
          type="text"
          placeholder="Quick search..."
          className="w-full py-2 pl-8 pr-3 text-sm bg-zinc-800 text-white rounded-lg border border-zinc-70
        />
        <Search
          className="absolute left-2 top-1/2 transform -translate-y-1/2 text-gray-400"
          size={16}
        />
      </div>
    </div>
  </div>


<div className="mx-auto px-4 lg:px-0 max-w-screen-2xl">
  <div className="flex flex-col lg:flex-row gap-8 left">
    {/* Left Sidebar */}
    <div className="hidden lg:block lg:w-1/4 sticky top-24 h-[calc(100vh-6rem)] overflow-y-auto">
      <LeftSidebar />
    </div>


    {/* Main Content */}
    <div className="lg:w-1/2 lg:mx-8 main " ref={contentRef}>
      {/* Hero Section */}
      <section className="pt-48 lg:pt-32 pb-12 px-4">
        <div className="mx-auto max-w-4xl">
          <h1 className="text-3xl md:text-4xl font-bold mb-6">
            {article?.title}
          </h1>
          <p className="text-gray-400 text-lg mb-8 max-w-2xl">
            {article?.shortSummary}
          </p>
          <div className="flex flex-wrap gap-4">
            <Button className="inline-flex items-center justify-center px-6 py-3 bg-blue-400 text-black
              Share <Share2 className="ml-2" size={20} />
            </Button>
            <div>
              <Button
                onClick={() => handleDownload(article?.pdfUrl)}
                className="inline-flex items-center justify-center px-6 py-3 border border-blue-400 text
              >
                <Eye className="mr-2" size={20} /> Read Later
              </Button>
            </div>
          </div>
        </div>
      </section>


      <div className="mb-16 relative">
        <div className="relative aspect-[16/9] rounded-xl overflow-hidden bg-zinc-800">
          <Image
            alt={article?.title || "Article image"}
            className="max-w-[90vw] max-h-[90vh] mx-auto my-auto"
            src={article?.imageUrl || "/"}
            blurDataURL={`${
```

```
                  article && article.blurImage
                }`}
                placeholder="blur"
                layout="fill"
                objectFit="cover"
              />
            </div>
            {/* Decorative elements */}
            <div className="absolute -inset-x-4 -inset-y-4 z-0 bg-gradient-to-r from-blue-400/20 to-purple-4
            <div className="absolute -inset-x-10 -inset-y-10 z-0 bg-blue-400/10 opacity-30 blur-2xl" />
          </div>


          {/* Article Content */}
          <div
            ref={articleContentRef}
            className="prose prose-invert max-w-none"
          >
            {/* The article content will be inserted here by the useEffect */}
          </div>
        </div>


        {/* Right Sidebar */}
        <div className="lg:w-1/4 sticky top-24 h-[calc(100vh-6rem)] overflow-y-auto right">
          <div className="space-y-8 p-4">
            {/* Social Media Section */}
            <div className="bg-zinc-800/50 backdrop-blur-sm text-gray-400 rounded-lg p-6">
              <h2 className="text-white text-lg font-semibold mb-4">
                Share This Article
              </h2>
              <div className="flex flex-wrap justify-start items-start gap-4">
                <Button className="bg-blue-400 text-black hover:bg-blue-300 transition-colors">
                  <Share2 className="mr-2" size={18} /> Share
                </Button>
                <Button
                  variant="outline"
                  className="border-blue-400 text-white hover:bg-blue-400 hover:text-black transition-all"
                >
                  <Eye className="mr-2" size={18} /> Read Later
                </Button>
              </div>
            </div>
            <div className="bg-zinc-800/50 backdrop-blur-sm text-gray-400 rounded-lg p-6">
              <h3 className="text-xl font-bold mb-2">Related Articles</h3>
              <ul>
                {data?.relatedArticles
                  ?.slice(0, 5)
                  .map((relatedArticle, index) => (
                    <li key={index} className="mb-2">
                      <a
                        href="#"
                        className="text-blue-400 hover:text-blue-300 transition-colors"
                      >
                        {relatedArticle.title}
                      </a>
                    </li>
                  ))}
              </ul>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};


export default ArticleLayout;
```

In addition to educational value, Christian games and puzzles serve as excellent tools for building community. Whether used in church groups, family gatherings, or youth ministries, these activities create opportunities for fellowship and discussion. They provide a relaxed setting where individuals can bond over shared experiences and deepen their relationships with one another while exploring their faith.

Here's a bullet list of the key points from the updated approach:

- **Removed `onSuccess`**:
  React Query v5 no longer supports `onSuccess` in the `useQuery` options.
- **Moved success handling to `useEffect`**:
  Used a `useEffect` to watch the `isSuccess` state returned by `useQuery`.
  After a successful fetch, the `shouldFetch` state is set to `false`, preventing further queries.
- **Used `enabled` option**:
  Controlled query triggering using the `enabled` option based on `shouldFetch`.
- **Fetch only on initial render**:
  Set `shouldFetch` to `true` after the component mounts via `useEffect`.

This is an Ordered Bullet List

1. Moreover, Christian puzzles, such as those featuring scripture-based crosswords or word searches, offer a solitary yet contemplative experience. These puzzles allow individuals to engage in personal reflection and meditation, helping them to connect with their faith on a deeper level.
2. Christian games and puzzles also address various age groups and learning styles. For children, they offer a playful way to learn about Bible stories and values, encouraging early spiritual development.
3. For adults, these activities provide a means to revisit and reflect on their faith in a meaningful way, often serving as conversation starters or teaching tools in group settings.

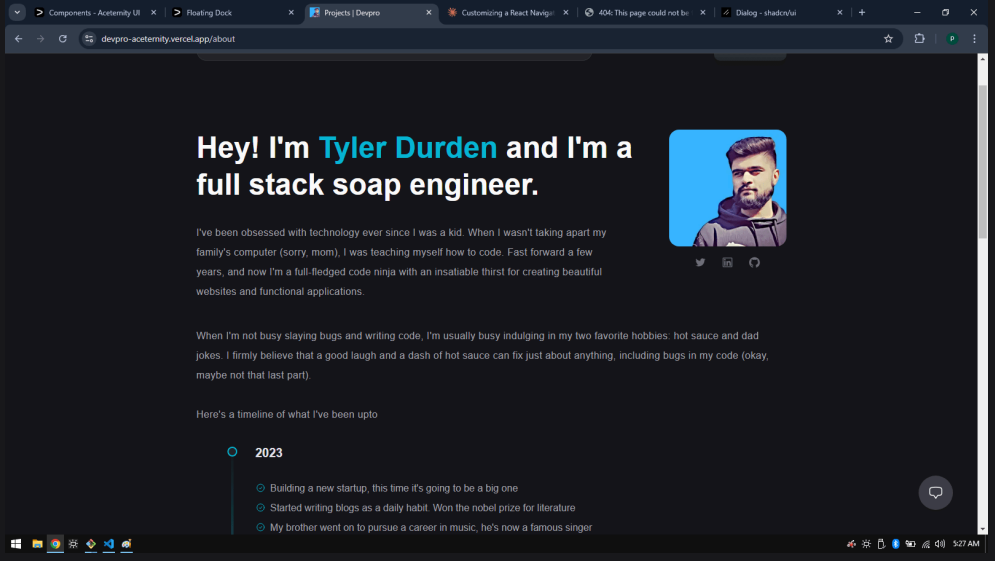
```
@tailwind base;
@tailwind components;
@tailwind utilities;


@layer base {
  :root {
    --background: 0 0% 100%;
    --foreground: 240 10% 3.9%;
    --buttonBackground: 99 64% 85%;


    --card: 0 0% 100%;
    --card-foreground: 240 10% 3.9%;


    --popover: 0 0% 100%;
    --popover-foreground: 240 10% 3.9%;


    --primary: 240 5.9% 10%;
    --primary-foreground: 0 0% 98%;


    --secondary: 240 4.8% 95.9%;
    --secondary-foreground: 240 5.9% 10%;


    --muted: 240 4.8% 95.9%;
    --muted-foreground: 240 3.8% 46.1%;


    --accent: 240 4.8% 95.9%;
    --accent-foreground: 240 5.9% 10%;


    --destructive: 0 84.2% 60.2%;
    --destructive-foreground: 0 0% 98%;


    --border: 240 5.9% 90%;
    --input: 240 5.9% 90%;
```

```
    --ring: 240 10% 3.9%;


    --radius: 0.5rem;
  }


  .dark {
    --background: 240 10% 3.9%;
    --foreground: 0 0% 98%;
    --buttonBackground: 99 64% 85%;
    --card: 240 10% 3.9%;
    --card-foreground: 0 0% 98%;


    --popover: 240 10% 3.9%;
    --popover-foreground: 0 0% 98%;


    --primary: 0 0% 98%;
    --primary-foreground: 240 5.9% 10%;


    --secondary: 240 3.7% 15.9%;
    --secondary-foreground: 0 0% 98%;


    --muted: 240 3.7% 15.9%;
    --muted-foreground: 240 5% 64.9%;


    --accent: 240 3.7% 15.9%;
    --accent-foreground: 0 0% 98%;


    --destructive: 0 62.8% 30.6%;
    --destructive-foreground: 0 0% 98%;


    --border: 240 3.7% 15.9%;
    --input: 240 3.7% 15.9%;
    --ring: 240 4.9% 83.9%;
  }
}


.ql-syntax {
  background-color: #282c34 !important;
}


ol {
  list-style-type: decimal; /* Use 'decimal' for normal numbers, 'lower-alpha' for lowercase letters, etc. */
  margin-left: 20px;
}


/* Set unordered lists to have dots */
ul {
  list-style-type: disc; /* Use 'disc' for filled dots, 'circle' for hollow circles, etc. */
  margin-left: 14px;
}
ul li {
  padding-left: 8px;
}
ol li {
  padding-left: 15px;
}
@media (hover: hover) {
  .ql-syntax::-webkit-scrollbar {
```

```
    width: 7px;
    height: 7px;
    cursor: grab;
  }


  .ql-syntax::-webkit-scrollbar-thumb {
    background-color: #c8c8c8;
    border-radius: 7px;
    height: 7px;
  }


  .ql-syntax::-webkit-scrollbar-track {
    background-color: transparent;
    border-radius: 6px;
  }
}


/* ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/* custom scroll bar  */
@media (hover: hover) {
  .custom-scrollbar::-webkit-scrollbar {
    scroll-behavior: smooth;
    width: 7px;
    height: 7px;
    cursor: grab;
  }


  .custom-scrollbar::-webkit-scrollbar-thumb {
    background-color: #565658;
    border-radius: 7px;
    height: 7px;
    cursor: grab;
  }


  .custom-scrollbar::-webkit-scrollbar-track {
    background-color: transparent;
    border-radius: 10px;
  }
}
```

The impact of Christian games and puzzles extends beyond personal enrichment. They can be used as effective outreach tools, helping to introduce individuals to faith in a non-threatening and engaging manner. By offering a blend of entertainment and education, these resources have the potential to reach a wider audience and facilitate discussions about faith in diverse settings.

> In summary, Christian games and puzzles are invaluable resources that blend fun with faith, providing opportunities for spiritual growth, community building, and personal reflection. As they continue to evolve and adapt to new formats, they remain a testament to the creative ways in which faith can be woven into everyday life. Embracing these activities not only enhances one's spiritual journey but



also fosters a deeper connection with both scripture and fellow believers.