**Game-Hub**

**documentation**

**Contents**

[1. Implementing the dark mode 3](#_Toc131586372)

[2. Building the color mode switch 4](#_Toc131586373)

[3. Fetching the Games 5](#_Toc131586374)

[4. Creating a Custom Hook for Fetching Games 6](#_Toc131586375)

[5. Building Game Cards 8](#_Toc131586376)

[6. Displaying Platform Icons 10](#_Toc131586377)

# Implementing the dark mode

theme.ts

import { extendTheme, type ThemeConfig } from "@chakra-ui/react";

const config: ThemeConfig = {

  initialColorMode: "dark",

};

const theme = extendTheme({ config });

export default theme;

main.tsx

import React from "react";

import ReactDOM from "react-dom/client";

import { ChakraProvider, ColorModeScript } from "@chakra-ui/react";

import App from "./App";

import theme from "./theme";

import "./index.css";

ReactDOM.createRoot(document.getElementById("root") as HTMLElement).render(

  <React.StrictMode>

    <ChakraProvider theme={theme}>

      <ColorModeScript initialColorMode={theme.config.initialColorMode} />

      <App />

    </ChakraProvider>

  </React.StrictMode>

Clear Local Storage => cahkra-ui-color-mode => memorize the theme color mode

# Building the color mode switch

In components folder, ColorModeSwitch.tsx

import { HStack, Switch, Text, useColorMode } from "@chakra-ui/react";

const ColorModeSwitch = () => {

  const { toggleColorMode, colorMode } = useColorMode();

  return (

    <HStack>

      <Switch

        colorScheme="green"

        isChecked={colorMode === "dark"}

        onChange={toggleColorMode}

      />

      <Text>Dark Mode</Text>

    </HStack>

  );

};

export default ColorModeSwitch;

In components folder, NavBar.tsx

import { HStack, Image } from "@chakra-ui/react";

import logo from "../assets/GameHub Resources/Logo/logo.webp";

import ColorModeSwitch from "./ColorModeSwitch";

const NavBar = () => {

  return (

*/\* navbar එක  horizontal stack එකක් විදියට.. \*/*

*/\* works just like a flex-container \*/*

    <HStack justifyContent="space-between" padding="10px">

      <Image src={logo} boxSize="60px" />

      <ColorModeSwitch />

    </HStack>

  );

};

export default NavBar;

# Fetching the Games

Install AXIOS => npm i axios in the terminal

In services folder;

import axios from "axios";

*/// මේ service module එකෙන් default export කරන්නෙ පහල එක. මේ module එකට කොහෙදි*

*/// හරි කෝල් කලොත්(ඕන නමක් දීල) එන්නෙ පහල එක.*

export default axios.create({

  baseURL: "https://api.rawg.io/api",

  params: {

    key: "15a75c9eddda41cab85f7ea0887a3bd2",

  },

});

In components folder, GameGrid.tsx

import { Text } from "@chakra-ui/react";

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

*//\*apiClient මේකට ඕන නමක් දැම්මෑකි*

import apiClient from "../services/api-client";

interface Game {

  id: number;

  name: string;

}

interface FetchGamesResponse {

  count: number;

  results: Game[];

}

const GameGrid = () => {

  const [games, setGames] = useState<Game[]>([]);

  const [error, setError] = useState("");

  useEffect(() => {

    apiClient

      .get<FetchGamesResponse>("/games")

      .then((res) => setGames(res.data.results))

      .catch((err) => setError(err.message));

  },[]);

  return (

    <>

      {error && <Text>{error}</Text>}

      <ul>

        {games.map((game) => (

          <li key={game.id}>{game.name}</li>

        ))}

      </ul>

    </>

  );

};

export default GameGrid;

# Creating a Custom Hook for Fetching Games

In hooks folder, useGames.ts custom hook

import { useEffect, useState } from "react";

*//\*apiClient මේකට ඕන නමක් දැම්මෑකි*

import apiClient from "../services/api-client";

import { CanceledError } from "axios";

interface Game {

  id: number;

  name: string;

}

interface FetchGamesResponse {

  count: number;

  results: Game[];

}

const useGames = () => {

  const [games, setGames] = useState<Game[]>([]);

  const [error, setError] = useState("");

  useEffect(() => {

    const controller = new AbortController();

    apiClient

      .get<FetchGamesResponse>("/games", { signal: controller.signal })

      .then((res) => setGames(res.data.results))

      .catch((err) => {

        if (err instanceof CanceledError) return;

        setError(err.message);

      });

    return () => controller.abort();

  }, []);

  return { games, error };

};

export default useGames;

In components folder, GameGrid.tsx

import { Text } from "@chakra-ui/react";

import useGames from "../hooks/useGames";

const GameGrid = () => {

  const { games, error } = useGames();

  return (

    <>

      {error && <Text>{error}</Text>}

      <ul>

        {games.map((game) => (

          <li key={game.id}>{game.name}</li>

        ))}

      </ul>

    </>

  );

};

export default GameGrid;

# Building Game Cards

In components folder, GameGrid.tsx

import { SimpleGrid, Text } from "@chakra-ui/react";

import useGames from "../hooks/useGames";

import GameCard from "./GameCard";

const GameGrid = () => {

  const { games, error } = useGames();

  return (

    <>

      {error && <Text>{error}</Text>}

      <SimpleGrid

        columns={{ sm: 1, md: 2, lg: 3, xl: 5 }}

        spacing={10}

        padding="10px"

      >

        {games.map((game) => (

          <GameCard key={game.id} game={game} />

        ))}

      </SimpleGrid>

    </>

  );

};

export default GameGrid;

In components folder, GameCard.tsx

import React from "react";

import { Game } from "../hooks/useGames";

import { Card, CardBody, Heading, Image } from "@chakra-ui/react";

interface Props {

  game: Game;

}

const GameCard = ({ game }: Props) => {

  return (

    <Card borderRadius={10} overflow="hidden">

      <Image src={game.background\_image} />

      <CardBody>

        <Heading fontSize="2xl">{game.name}</Heading>

      </CardBody>

    </Card>

  );

};

export default GameCard;

In hooks folder, useGames.ts

import { useEffect, useState } from "react";

*//\*apiClient මේකට ඕන නමක් දැම්මෑකි*

import apiClient from "../services/api-client";

import { CanceledError } from "axios";

export interface Game {

  id: number;

  name: string;

  background\_image: string;

}

interface FetchGamesResponse {

  count: number;

  results: Game[];

}

const useGames = () => {

  const [games, setGames] = useState<Game[]>([]);

  const [error, setError] = useState("");

  useEffect(() => {

    const controller = new AbortController();

    apiClient

      .get<FetchGamesResponse>("/games", { signal: controller.signal })

      .then((res) => setGames(res.data.results))

      .catch((err) => {

        if (err instanceof CanceledError) return;

        setError(err.message);

      });

    return () => controller.abort();

  }, []);

  return { games, error };

};

export default useGames;

# Displaying Platform Icons

Install REACT ICONS => npm i [react-icons@4.7.1](mailto:react-icons@4.7.1)

In components, GameCard.tsx

import React from "react";

import { Game } from "../hooks/useGames";

import { Card, CardBody, Heading, Image, Text } from "@chakra-ui/react";

import PlatformIconList from "./PlatformIconList";

interface Props {

  game: Game;

}

const GameCard = ({ game }: Props) => {

  return (

    <Card borderRadius={10} overflow="hidden">

      <Image src={game.background\_image} />

      <CardBody>

        <Heading fontSize="2xl">{game.name}</Heading>

        <PlatformIconList

          platforms={game.parent\_platforms.map((p) => p.platform)}

        />

      </CardBody>

    </Card>

  );

};

export default GameCard;

In components, PlatformIconList.tsx

import { HStack, Icon, Text } from "@chakra-ui/react";

import {

  FaWindows,

  FaPlaystation,

  FaXbox,

  FaApple,

  FaLinux,

  FaAndroid,

} from "react-icons/fa";

import { MdPhoneIphone } from "react-icons/md";

import { SiNintendo } from "react-icons/si";

import { BsGlobe } from "react-icons/bs";

import { Platform } from "../hooks/useGames";

import { IconType } from "react-icons";

interface Props {

  platforms: Platform[];

}

const PlatformIconList = ({ platforms }: Props) => {

  const iconMap: { [key: string]: IconType } = {

*//name: Play Station,*

*//slug: playstation => this may not change i the future. So use this.*

    pc: FaWindows,

    playstation: FaPlaystation,

    xbox: FaXbox,

    nintendo: SiNintendo,

    mac: FaApple,

    linux: FaLinux,

    android: FaAndroid,

    ios: MdPhoneIphone,

    web: BsGlobe,

  };

  return (

    <HStack marginY={1}>

      {platforms.map((platform) => (

        <Icon as={iconMap[platform.slug]} color="gray.500"></Icon>

      ))}

    </HStack>

  );

};

export default PlatformIconList;

In hooks, useGames.ts

import { useEffect, useState } from "react";

*//\*apiClient මේකට ඕන නමක් දැම්මෑකි*

import apiClient from "../services/api-client";

import { CanceledError } from "axios";

export interface Platform {

  id: number;

  name: string;

  slug: string;

}

export interface Game {

  id: number;

  name: string;

  background\_image: string;

*//? මේක මෙහෙම උනේ RAWG.IO එක design කරල තියෙන විදියෙ අවුලක් හින්ද,*

*//\* this is an array of objects, where each object has a property called "platform" of*

*//\* type Platform.*

  parent\_platforms: { platform: Platform }[];

}

interface FetchGamesResponse {

  count: number;

  results: Game[];

}

const useGames = () => {

  const [games, setGames] = useState<Game[]>([]);

  const [error, setError] = useState("");

  useEffect(() => {

    const controller = new AbortController();

    apiClient

      .get<FetchGamesResponse>("/games", { signal: controller.signal })

      .then((res) => setGames(res.data.results))

      .catch((err) => {

        if (err instanceof CanceledError) return;

        setError(err.message);

      });

    return () => controller.abort();

  }, []);

  return { games, error };

};

export default useGames;

# Displaying Critic Score

In hooks folder, useGames.ts

import { useEffect, useState } from "react";

*//\*apiClient මේකට ඕන නමක් දැම්මෑකි*

import apiClient from "../services/api-client";

import { CanceledError } from "axios";

export interface Platform {

  id: number;

  name: string;

  slug: string;

}

export interface Game {

  id: number;

  name: string;

  background\_image: string;

*//? මේක මෙහෙම උනේ RAWG.IO එක design කරල තියෙන විදියෙ අවුලක් හින්ද,*

*//\* this is an array of objects, where each object has a property called "platform" of*

*//\* type Platform.*

  parent\_platforms: { platform: Platform }[];

  metacritic: number;

}

interface FetchGamesResponse {

  count: number;

  results: Game[];

}

const useGames = () => {

  const [games, setGames] = useState<Game[]>([]);

  const [error, setError] = useState("");

  useEffect(() => {

    const controller = new AbortController();

    apiClient

      .get<FetchGamesResponse>("/games", { signal: controller.signal })

      .then((res) => setGames(res.data.results))

      .catch((err) => {

        if (err instanceof CanceledError) return;

        setError(err.message);

      });

    return () => controller.abort();

  }, []);

  return { games, error };

};

export default useGames;

In components folder, GameCard.tsx

import React from "react";

import { Game } from "../hooks/useGames";

import { Card, CardBody, HStack, Heading, Image, Text } from "@chakra-ui/react";

import PlatformIconList from "./PlatformIconList";

import CriticScore from "./CriticScore";

interface Props {

  game: Game;

}

const GameCard = ({ game }: Props) => {

  return (

    <Card borderRadius={10} overflow="hidden">

      <Image src={game.background\_image} />

      <CardBody>

        <Heading fontSize="2xl">{game.name}</Heading>

        <HStack justifyContent="space-between">

          <PlatformIconList

            platforms={game.parent\_platforms.map((p) => p.platform)}

          />

          <CriticScore score={game.metacritic} />

        </HStack>

      </CardBody>

    </Card>

  );

};

export default GameCard;

In components,CriticScore.tsx

import { Badge } from "@chakra-ui/react";

interface Props {

  score: number;

}

const CriticScore = ({ score }: Props) => {

  let color = score > 75 ? "green" : score > 60 ? "yellow" : "";

  return (

    <Badge colorScheme={color} fontSize="14px" paddingX={2} borderRadius="4px">

      {score}

    </Badge>

  );

};

export default CriticScore;

# Getting Optimized Images

In services folder, image-url.ts

const getCroppedImageUrl = (url: string) => {

  const target = "media/";

  const index = url.indexOf(target) + target.length;

  return url.slice(0, index) + "crop/600/400/" + url.slice(index);

};

export default getCroppedImageUrl;

In components folder, GameCard.tsx

import React from "react";

import { Game } from "../hooks/useGames";

import { Card, CardBody, HStack, Heading, Image, Text } from "@chakra-ui/react";

import PlatformIconList from "./PlatformIconList";

import CriticScore from "./CriticScore";

import getCroppedImageUrl from "../services/image-url";

interface Props {

  game: Game;

}

const GameCard = ({ game }: Props) => {

  return (

    <Card borderRadius={10} overflow="hidden">

      <Image src={getCroppedImageUrl(game.background\_image)} />

      <CardBody>

        <Heading fontSize="2xl">{game.name}</Heading>

        <HStack justifyContent="space-between">

          <PlatformIconList

            platforms={game.parent\_platforms.map((p) => p.platform)}

          />

          <CriticScore score={game.metacritic} />

        </HStack>

      </CardBody>

    </Card>

  );

};

export default GameCard;

# Improving User Experience with Loading Skeletons

In hooks folder, useGames.ts

import { useEffect, useState } from "react";

*//\*apiClient මේකට ඕන නමක් දැම්මෑකි*

import apiClient from "../services/api-client";

import { CanceledError } from "axios";

export interface Platform {

  id: number;

  name: string;

  slug: string;

}

export interface Game {

  id: number;

  name: string;

  background\_image: string;

*//? මේක මෙහෙම උනේ RAWG.IO එක design කරල තියෙන විදියෙ අවුලක් හින්ද,*

*//\* this is an array of objects, where each object has a property called "platform" of*

*//\* type Platform.*

  parent\_platforms: { platform: Platform }[];

  metacritic: number;

}

interface FetchGamesResponse {

  count: number;

  results: Game[];

}

const useGames = () => {

  const [games, setGames] = useState<Game[]>([]);

  const [error, setError] = useState("");

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

  useEffect(() => {

    const controller = new AbortController();

    setLoading(true);

    apiClient

      .get<FetchGamesResponse>("/games", { signal: controller.signal })

      .then((res) => {

        setGames(res.data.results);

        setLoading(false);

      })

      .catch((err) => {

        if (err instanceof CanceledError) return;

        setError(err.message);

        setLoading(false);

      });

    return () => controller.abort();

  }, []);

  return { games, error, isLoading };

};

export default useGames;

In components folder, GameCardSkeleton.tsx

import {

  Card,

  CardBody,

  Heading,

  Skeleton,

  SkeletonText,

} from "@chakra-ui/react";

const GameCardSkeleton = () => {

  return (

    <Card width="300px" borderRadius={10} overflow="hidden">

      <Skeleton height="200px" />

      <CardBody>

        <SkeletonText />

      </CardBody>

    </Card>

  );

};

export default GameCardSkeleton;

In components folder, GameCard.tsx

import React from "react";

import { Game } from "../hooks/useGames";

import { Card, CardBody, HStack, Heading, Image, Text } from "@chakra-ui/react";

import PlatformIconList from "./PlatformIconList";

import CriticScore from "./CriticScore";

import getCroppedImageUrl from "../services/image-url";

interface Props {

  game: Game;

}

const GameCard = ({ game }: Props) => {

  return (

    <Card width="300px" borderRadius={10} overflow="hidden">

      <Image src={getCroppedImageUrl(game.background\_image)} />

      <CardBody>

        <Heading fontSize="2xl">{game.name}</Heading>

        <HStack justifyContent="space-between">

          <PlatformIconList

            platforms={game.parent\_platforms.map((p) => p.platform)}

          />

          <CriticScore score={game.metacritic} />

        </HStack>

      </CardBody>

    </Card>

  );

};

export default GameCard;

In components folder, GameGrid.tsx

import { SimpleGrid, Skeleton, Text } from "@chakra-ui/react";

import useGames from "../hooks/useGames";

import GameCard from "./GameCard";

import GameCardSkeleton from "./GameCardSkeleton";

const GameGrid = () => {

  const { games, error, isLoading } = useGames();

  const skeletons = [1, 2, 3, 4, 5, 6, 7, 8];

  return (

    <>

      {error && <Text>{error}</Text>}

      <SimpleGrid

        columns={{ sm: 1, md: 2, lg: 3, xl: 4 }}

        spacing={10}

        padding="10px"

      >

        {isLoading &&

          skeletons.map((skeleton) => <GameCardSkeleton key={skeleton} />)}

        {games.map((game) => (

          <GameCard key={game.id} game={game} />

        ))}

      </SimpleGrid>

    </>

  );

};

export default GameGrid;

# Refactor- Removing Duplicated Styles

In components folder, GameCardContainer.tsx

import { Box } from "@chakra-ui/react";

import { ReactNode } from "react";

interface Props {

  children: ReactNode;

}

const GameCardContainer = ({ children }: Props) => {

*//, this retrns a div*

  return (

    <Box width="300px" borderRadius={10} overflow="hidden">

      {children}

    </Box>

  );

};

export default GameCardContainer;

In components folder, GameGrid.tsx

import { SimpleGrid, Skeleton, Text } from "@chakra-ui/react";

import useGames from "../hooks/useGames";

import GameCard from "./GameCard";

import GameCardSkeleton from "./GameCardSkeleton";

import GameCardContainer from "./GameCardContainer";

const GameGrid = () => {

  const { games, error, isLoading } = useGames();

  const skeletons = [1, 2, 3, 4, 5, 6, 7, 8];

  return (

    <>

      {error && <Text>{error}</Text>}

      <SimpleGrid

        columns={{ sm: 1, md: 2, lg: 3, xl: 4 }}

        spacing={10}

        padding="10px"

      >

        {isLoading &&

          skeletons.map((skeleton) => (

            <GameCardContainer>

              <GameCardSkeleton key={skeleton} />

            </GameCardContainer>

          ))}

        {games.map((game) => (

          <GameCardContainer>

            <GameCard key={game.id} game={game} />

          </GameCardContainer>

        ))}

      </SimpleGrid>

    </>

  );

};

export default GameGrid;

# Fetching the Genres