**Game-Hub**

**documentation**

**Contents**

[1. Implementing the dark mode 4](#_Toc131842128)

[2. Building the color mode switch 5](#_Toc131842129)

[3. Fetching the Games 6](#_Toc131842130)

[4. Creating a Custom Hook for Fetching Games 7](#_Toc131842131)

[5. Building Game Cards 9](#_Toc131842132)

[6. Displaying Platform Icons 11](#_Toc131842133)

[7. Displaying Critic Score 14](#_Toc131842134)

[8. Getting Optimized Images 16](#_Toc131842135)

[9. Improving User Experience with Loading Skeletons 17](#_Toc131842136)

[10. Refactor- Removing Duplicated Styles 21](#_Toc131842137)

[11. Fetching the Genres 22](#_Toc131842138)

[12. Creating a Generic Data Fetching Hook 24](#_Toc131842139)

[13. Displaying the Genres 28](#_Toc131842140)

[14. Showing a Spinner 31](#_Toc131842141)

[15. Filtering Games by Genre 32](#_Toc131842142)

[16. Highlighting the Selected Genre 38](#_Toc131842143)

[17. Building Platform Selector 40](#_Toc131842144)

[18. Filtering Games by Platform 42](#_Toc131842145)

[19. Refactoring- Extracting a Query Object 47](#_Toc131842146)

[20. Building Sort Selector 51](#_Toc131842147)

[21. Sorting Games 53](#_Toc131842148)

[22. Handling Games without an Image 58](#_Toc131842149)

[23. Fixing the Issue with Chakra Menus 58](#_Toc131842150)

[24. Building Search Input 60](#_Toc131842151)

[25. Searching Games 62](#_Toc131842152)

[26. Adding a Dynamic Heading 66](#_Toc131842153)

[27. Cleaning Up the Genres 69](#_Toc131842154)

[28. Cleaning Up the Game Cards 70](#_Toc131842155)

[29. Adding Emojis 72](#_Toc131842156)

[30. Shipping Static Data 75](#_Toc131842157)

[31. Customizing the Chakra Theme 75](#_Toc131842158)

[32. Refactoring Game Grid 76](#_Toc131842159)

[33. Building for Production 77](#_Toc131842160)

[34. Deploying to Vercel 77](#_Toc131842161)

# 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

In hooks folder, useGenres.ts

import { useEffect, useState } from "react";

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

import { CanceledError } from "axios";

interface Genre {

  id: number;

  name: string;

}

interface FetchGenresResponse {

  count: number;

  results: Genre[];

}

const useGenres = () => {

  const [genres, setGenres] = useState<Genre[]>([]);

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

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

  useEffect(() => {

    const controller = new AbortController();

    setLoading(true);

    apiClient

      .get<FetchGenresResponse>("/genres", { signal: controller.signal })

      .then((res) => {

        setGenres(res.data.results);

        setLoading(false);

      })

      .catch((err) => {

        if (err instanceof CanceledError) return;

        setError(err.message);

        setLoading(false);

      });

    return () => controller.abort();

  }, []);

  return { genres, error, isLoading };

};

export default useGenres;

In components folder, GenreList.tsx

import useGenres from "../hooks/useGenres";

const GenreList = () => {

  const { genres } = useGenres();

  return (

    <ul>

      {genres.map((genre) => (

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

      ))}

    </ul>

  );

};

export default GenreList;

In App.tsx

import { Grid, GridItem, Show } from "@chakra-ui/react";

import NavBar from "./components/NavBar";

import GameGrid from "./components/GameGrid";

import GenreList from "./components/GenreList";

function App() {

  return (

    <Grid

      templateAreas={{

*/\* an object is passed here \*/*

        base: `"nav" "main"` */\* mobile phone වලට \*/*,

        lg: `"nav nav" "aside main"` */\* 1024px laptop වගේ ලොකු screen තියෙන ඒවට. \*/*,

      }}

    >

      {*/\* navigation bar එක \*/*}

      <GridItem area="nav">

        <NavBar />

      </GridItem>

      {*/\* show එකෙන් large screen device වලට විතරක් aside grid item එක පේන්න හැදුවෑකි. \*/*}

      <Show above="lg">

        <GridItem area="aside">

          <GenreList />

        </GridItem>

      </Show>

      {*/\* main area එක \*/*}

      <GridItem area="main">

        <GameGrid />

      </GridItem>

    </Grid>

  );

}

export default App;

# Creating a Generic Data Fetching Hook

In hooks folder, useData.ts

import { useEffect, useState } from "react";

*//\*apiClient මේකට ඕන නමක් දැම්මෑකි*

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

import { CanceledError } from "axios";

interface FetchResponse<T> {

  count: number;

  results: T[];

}

const useData = <T>(endpoint: string) => {

  const [data, setData] = useState<T[]>([]);

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

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

  useEffect(() => {

    const controller = new AbortController();

    setLoading(true);

    apiClient

      .get<FetchResponse<T>>(endpoint, { signal: controller.signal })

      .then((res) => {

        setData(res.data.results);

        setLoading(false);

      })

      .catch((err) => {

        if (err instanceof CanceledError) return;

        setError(err.message);

        setLoading(false);

      });

    return () => controller.abort();

  }, []);

  return { data, error, isLoading };

};

export default useData;

In hooks folder, useGenres.ts

import useData from "./useData";

export interface Genre {

  id: number;

  name: string;

}

const useGenres = () => useData<Genre>("/genres");

export default useGenres;

In hooks folder, useGames.ts

import useData from "./useData";

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;

}

const useGames = () => useData<Game>("/games");

export default useGames;

In components folder, GenreList.tsx

import useGenres from "../hooks/useGenres";

const GenreList = () => {

  const { data } = useGenres();

  return (

    <ul>

      {data.map((genre) => (

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

      ))}

    </ul>

  );

};

export default GenreList;

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 { data, 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>

          ))}

        {data.map((game) => (

          <GameCardContainer>

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

          </GameCardContainer>

        ))}

      </SimpleGrid>

    </>

  );

};

export default GameGrid;

# Displaying the Genres

In hooks folder, useGenres.ts

import useData from "./useData";

export interface Genre {

  id: number;

  name: string;

  image\_background: string;

}

const useGenres = () => useData<Genre>("/genres");

export default useGenres;

In components folder, GenreList.tsx

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

import useGenres from "../hooks/useGenres";

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

const GenreList = () => {

  const { data } = useGenres();

  return (

    <List>

      {data.map((genre) => (

        <ListItem key={genre.id} paddingY="5px">

          <HStack>

            <Image

              boxSize="32px"

              borderRadius={8}

              src={getCroppedImageUrl(genre.image\_background)}

            />

            <Text fontSize="lg">{genre.name}</Text>

          </HStack>

        </ListItem>

      ))}

    </List>

  );

};

export default GenreList;

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 borderRadius={10} overflow="hidden">

      {children}

    </Box>

  );

};

export default GameCardContainer;

In App.tsx

import { Grid, GridItem, Show } from "@chakra-ui/react";

import NavBar from "./components/NavBar";

import GameGrid from "./components/GameGrid";

import GenreList from "./components/GenreList";

function App() {

  return (

    <Grid

      templateAreas={{

*/\* an object is passed here \*/*

        base: `"nav" "main"` */\* mobile phone වලට \*/*,

        lg: `"nav nav" "aside main"` */\* 1024px laptop වගේ ලොකු screen තියෙන ඒවට. \*/*,

      }}

      templateColumns={{

        base: "1fr",

        lg: "200px 1fr",

      }}

    >

      {*/\* navigation bar එක \*/*}

      <GridItem area="nav">

        <NavBar />

      </GridItem>

      {*/\* show එකෙන් large screen device වලට විතරක් aside grid item එක පේන්න හැදුවෑකි. \*/*}

      <Show above="lg">

        <GridItem area="aside" paddingX={5}>

          <GenreList />

        </GridItem>

      </Show>

      {*/\* main area එක \*/*}

      <GridItem area="main">

        <GameGrid />

      </GridItem>

    </Grid>

  );

}

export default App;

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 { data, error, isLoading } = useGames();

  const skeletons = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];

  return (

    <>

      {error && <Text>{error}</Text>}

      <SimpleGrid

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

        spacing={3}

        padding="10px"

      >

        {isLoading &&

          skeletons.map((skeleton) => (

            <GameCardContainer key={skeleton}>

              <GameCardSkeleton key={skeleton} />

            </GameCardContainer>

          ))}

        {data.map((game) => (

          <GameCardContainer key={game.id}>

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

          </GameCardContainer>

        ))}

      </SimpleGrid>

    </>

  );

};

export default GameGrid;

# Showing a Spinner

In components folder, GenreList.tsx

import { HStack, Image, List, ListItem, Spinner, Text } from "@chakra-ui/react";

import useGenres from "../hooks/useGenres";

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

const GenreList = () => {

  const { data, isLoading, error } = useGenres();

  if (error) return null;

  if (isLoading) return <Spinner />;

  return (

    <List>

      {data.map((genre) => (

        <ListItem key={genre.id} paddingY="5px">

          <HStack>

            <Image

              boxSize="32px"

              borderRadius={8}

              src={getCroppedImageUrl(genre.image\_background)}

            />

            <Text fontSize="lg">{genre.name}</Text>

          </HStack>

        </ListItem>

      ))}

    </List>

  );

};

export default GenreList;

# Filtering Games by Genre

In components folder, GenreList.tsx

import {

  Button,

  HStack,

  Image,

  List,

  ListItem,

  Spinner,

  Text,

} from "@chakra-ui/react";

import useGenres, { Genre } from "../hooks/useGenres";

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

interface Props {

  onSelectGenre: (genre: Genre) => void;

}

const GenreList = ({ onSelectGenre }: Props) => {

  const { data, isLoading, error } = useGenres();

  if (error) return null;

  if (isLoading) return <Spinner />;

  return (

    <List>

      {data.map((genre) => (

        <ListItem key={genre.id} paddingY="5px">

          <HStack>

            <Image

              boxSize="32px"

              borderRadius={8}

              src={getCroppedImageUrl(genre.image\_background)}

            />

            <Button

              onClick={() => {

                onSelectGenre(genre);

                console.log(genre);

              }}

              fontSize="lg"

              variant="link"

            >

              {genre.name}

            </Button>

          </HStack>

        </ListItem>

      ))}

    </List>

  );

};

export default GenreList;

In APP component,

import { Grid, GridItem, Show } from "@chakra-ui/react";

import NavBar from "./components/NavBar";

import GameGrid from "./components/GameGrid";

import GenreList from "./components/GenreList";

import { useState } from "react";

import { Genre } from "./hooks/useGenres";

function App() {

  const [selectedGenre, setSelectedGenre] = useState<Genre | null>(null);

  return (

    <Grid

      templateAreas={{

*/\* an object is passed here \*/*

        base: `"nav" "main"` */\* mobile phone වලට \*/*,

        lg: `"nav nav" "aside main"` */\* 1024px laptop වගේ ලොකු screen තියෙන ඒවට. \*/*,

      }}

      templateColumns={{

        base: "1fr",

        lg: "200px 1fr",

      }}

    >

      {*/\* navigation bar එක \*/*}

      <GridItem area="nav">

        <NavBar />

      </GridItem>

      {*/\* show එකෙන් large screen device වලට විතරක් aside grid item එක පේන්න හැදුවෑකි. \*/*}

      <Show above="lg">

        <GridItem area="aside" paddingX={5}>

          <GenreList onSelectGenre={(genre) => setSelectedGenre(genre)} />

        </GridItem>

      </Show>

      {*/\* main area එක \*/*}

      <GridItem area="main">

        <GameGrid selectedGenre={selectedGenre} />

      </GridItem>

    </Grid>

  );

}

export default App;

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";

import { Genre } from "../hooks/useGenres";

interface Props {

  selectedGenre: Genre | null;

}

const GameGrid = ({ selectedGenre }: Props) => {

  const { data, error, isLoading } = useGames(selectedGenre);

  const skeletons = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];

  return (

    <>

      {error && <Text>{error}</Text>}

      <SimpleGrid

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

        spacing={3}

        padding="10px"

      >

        {isLoading &&

          skeletons.map((skeleton) => (

            <GameCardContainer key={skeleton}>

              <GameCardSkeleton key={skeleton} />

            </GameCardContainer>

          ))}

        {data.map((game) => (

          <GameCardContainer key={game.id}>

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

          </GameCardContainer>

        ))}

      </SimpleGrid>

    </>

  );

};

export default GameGrid;

In hooks folder, useGames.ts

import useData from "./useData";

import { Genre } from "./useGenres";

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;

}

*//? second arguments is one of the properties of the AXIOS request config object. Sent*

*//? as a Query String. set params to an object. In this games hook, we are passing the selected*

*//? genre, as a query string parameter to the useData hook. As part of this lesson,k we had to*

*//? open up our hook and make it more flexible. Now we can pass query string parameters or*

*//? request data to our request objects. We also added an array of dependencies, if any of*

*//? these dependencies changes, our effect hook in the useData.ts will re-run and refresh the*

*//? data from the server.*

const useGames = (selectedGenre: Genre | null) =>

  useData<Game>("/games", { params: { genres: selectedGenre?.id } }, [

    selectedGenre?.id,

  ]);

export default useGames;

In hooks folder, useData.ts

import { useEffect, useState } from "react";

*//\*apiClient මේකට ඕන නමක් දැම්මෑකි*

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

import { AxiosRequestConfig, CanceledError } from "axios";

interface FetchResponse<T> {

  count: number;

  results: T[];

}

const useData = <T>(

  endpoint: string,

  requestConfig?: AxiosRequestConfig,

  deps?: any[]

) => {

  const [data, setData] = useState<T[]>([]);

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

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

  useEffect(

    () => {

      const controller = new AbortController();

      setLoading(true);

      apiClient

*//, second argument is an AXIOS request config object. In this object, we pass data in the*

*//, request body. we can set query string parameters, headers, etc.*

        .get<FetchResponse<T>>(endpoint, {

          signal: controller.signal,

          ...requestConfig,

        })

        .then((res) => {

          setData(res.data.results);

          setLoading(false);

        })

        .catch((err) => {

          if (err instanceof CanceledError) return;

          setError(err.message);

          setLoading(false);

        });

      return () => controller.abort();

    },

    deps ? [...deps] : []

  );

  return { data, error, isLoading };

};

export default useData;

# Highlighting the Selected Genre

In components folder, GenreList.tsx

import {

  Button,

  HStack,

  Image,

  List,

  ListItem,

  Spinner,

  Text,

} from "@chakra-ui/react";

import useGenres, { Genre } from "../hooks/useGenres";

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

interface Props {

  onSelectGenre: (genre: Genre) => void;

  selectedGenre: Genre | null;

}

const GenreList = ({ onSelectGenre, selectedGenre }: Props) => {

  const { data, isLoading, error } = useGenres();

  if (error) return null;

  if (isLoading) return <Spinner />;

  return (

    <List>

      {data.map((genre) => (

        <ListItem key={genre.id} paddingY="5px">

          <HStack>

            <Image

              boxSize="32px"

              borderRadius={8}

              src={getCroppedImageUrl(genre.image\_background)}

            />

            <Button

              fontWeight={genre.id === selectedGenre?.id ? "bold" : "normal"}

              onClick={() => {

                onSelectGenre(genre);

                console.log(genre);

              }}

              fontSize="lg"

              variant="link"

            >

              {genre.name}

            </Button>

          </HStack>

        </ListItem>

      ))}

    </List>

  );

};

export default GenreList;

In App component

import { Grid, GridItem, Show } from "@chakra-ui/react";

import NavBar from "./components/NavBar";

import GameGrid from "./components/GameGrid";

import GenreList from "./components/GenreList";

import { useState } from "react";

import { Genre } from "./hooks/useGenres";

function App() {

  const [selectedGenre, setSelectedGenre] = useState<Genre | null>(null);

  return (

    <Grid

      templateAreas={{

*/\* an object is passed here \*/*

        base: `"nav" "main"` */\* mobile phone වලට \*/*,

        lg: `"nav nav" "aside main"` */\* 1024px laptop වගේ ලොකු screen තියෙන ඒවට. \*/*,

      }}

      templateColumns={{

        base: "1fr",

        lg: "250px 1fr",

      }}

    >

      {*/\* navigation bar එක \*/*}

      <GridItem area="nav">

        <NavBar />

      </GridItem>

      {*/\* show එකෙන් large screen device වලට විතරක් aside grid item එක පේන්න හැදුවෑකි. \*/*}

      <Show above="lg">

        <GridItem area="aside" paddingX={5}>

          <GenreList

            selectedGenre={selectedGenre}

            onSelectGenre={(genre) => setSelectedGenre(genre)}

          />

        </GridItem>

      </Show>

      {*/\* main area එක \*/*}

      <GridItem area="main">

        <GameGrid selectedGenre={selectedGenre} />

      </GridItem>

    </Grid>

  );

}

export default App;

# Building Platform Selector

In components folder, PlatformSelector.tsx

import { Button, Menu, MenuButton, MenuItem, MenuList } from "@chakra-ui/react";

import React from "react";

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

import usePlatforms from "../hooks/usePlatforms";

const PlatformSelector = () => {

  const { data, error } = usePlatforms();

  if (error) return null;

  return (

    <Menu>

      <MenuButton as={Button} rightIcon={<BsChevronDown />}>

        Platforms

      </MenuButton>

      <MenuList>

        {data.map((platform) => (

          <MenuItem key={platform.id}>{platform.name}</MenuItem>

        ))}

      </MenuList>

    </Menu>

  );

};

export default PlatformSelector;

In hooks folder, usePlatforms.ts

import useData from "./useData";

interface Platform {

  id: number;

  name: string;

  slug: string;

}

const usePlatforms = () => useData<Platform>("/platforms/lists/parents");

export default usePlatforms;

In App component,

import { Grid, GridItem, Show } from "@chakra-ui/react";

import NavBar from "./components/NavBar";

import GameGrid from "./components/GameGrid";

import GenreList from "./components/GenreList";

import { useState } from "react";

import { Genre } from "./hooks/useGenres";

import PlatformSelector from "./components/PlatformSelector";

function App() {

  const [selectedGenre, setSelectedGenre] = useState<Genre | null>(null);

  return (

    <Grid

      templateAreas={{

*/\* an object is passed here \*/*

        base: `"nav" "main"` */\* mobile phone වලට \*/*,

        lg: `"nav nav" "aside main"` */\* 1024px laptop වගේ ලොකු screen තියෙන ඒවට. \*/*,

      }}

      templateColumns={{

        base: "1fr",

        lg: "250px 1fr",

      }}

    >

      {*/\* navigation bar එක \*/*}

      <GridItem area="nav">

        <NavBar />

      </GridItem>

      {*/\* show එකෙන් large screen device වලට විතරක් aside grid item එක පේන්න හැදුවෑකි. \*/*}

      <Show above="lg">

        <GridItem area="aside" paddingX={5}>

          <GenreList

            selectedGenre={selectedGenre}

            onSelectGenre={(genre) => setSelectedGenre(genre)}

          />

        </GridItem>

      </Show>

      {*/\* main area එක \*/*}

      <GridItem area="main">

        <PlatformSelector />

        <GameGrid selectedGenre={selectedGenre} />

      </GridItem>

    </Grid>

  );

}

export default App;

# Filtering Games by Platform

In components folder, PlatformSelector.tsx

import { Button, Menu, MenuButton, MenuItem, MenuList } from "@chakra-ui/react";

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

import usePlatforms from "../hooks/usePlatforms";

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

interface Props {

  onSelectPlatform: (platform: Platform) => void;

  selectedPlatform: Platform | null;

}

const PlatformSelector = ({ onSelectPlatform, selectedPlatform }: Props) => {

  const { data, error } = usePlatforms();

  if (error) return null;

  return (

    <Menu>

      <MenuButton as={Button} rightIcon={<BsChevronDown />}>

        {selectedPlatform?.name || "Platforms"}

      </MenuButton>

      <MenuList>

        {data.map((platform) => (

          <MenuItem

            onClick={() => onSelectPlatform(platform)}

            key={platform.id}

          >

            {platform.name}

          </MenuItem>

        ))}

      </MenuList>

    </Menu>

  );

};

export default PlatformSelector;

In App component

import { Grid, GridItem, Show } from "@chakra-ui/react";

import NavBar from "./components/NavBar";

import GameGrid from "./components/GameGrid";

import GenreList from "./components/GenreList";

import { useState } from "react";

import { Genre } from "./hooks/useGenres";

import PlatformSelector from "./components/PlatformSelector";

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

function App() {

  const [selectedGenre, setSelectedGenre] = useState<Genre | null>(null);

  const [selectedPlatform, setSelectedPlatform] = useState<Platform | null>(

    null

  );

  return (

    <Grid

      templateAreas={{

*/\* an object is passed here \*/*

        base: `"nav" "main"` */\* mobile phone වලට \*/*,

        lg: `"nav nav" "aside main"` */\* 1024px laptop වගේ ලොකු screen තියෙන ඒවට. \*/*,

      }}

      templateColumns={{

        base: "1fr",

        lg: "250px 1fr",

      }}

    >

      {*/\* navigation bar එක \*/*}

      <GridItem area="nav">

        <NavBar />

      </GridItem>

      {*/\* show එකෙන් large screen device වලට විතරක් aside grid item එක පේන්න හැදුවෑකි. \*/*}

      <Show above="lg">

        <GridItem area="aside" paddingX={5}>

          <GenreList

            selectedGenre={selectedGenre}

            onSelectGenre={(genre) => setSelectedGenre(genre)}

          />

        </GridItem>

      </Show>

      {*/\* main area එක \*/*}

      <GridItem area="main">

        <PlatformSelector

          selectedPlatform={selectedPlatform}

          onSelectPlatform={(platform) => setSelectedPlatform(platform)}

        />

        <GameGrid

          selectedPlatform={selectedPlatform}

          selectedGenre={selectedGenre}

        />

      </GridItem>

    </Grid>

  );

}

export default App;

In components folder, GameGrid.tsx

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

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

import GameCard from "./GameCard";

import GameCardSkeleton from "./GameCardSkeleton";

import GameCardContainer from "./GameCardContainer";

import { Genre } from "../hooks/useGenres";

interface Props {

  selectedGenre: Genre | null;

  selectedPlatform: Platform | null;

}

const GameGrid = ({ selectedGenre, selectedPlatform }: Props) => {

  const { data, error, isLoading } = useGames(selectedGenre, selectedPlatform);

  const skeletons = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];

  return (

    <>

      {error && <Text>{error}</Text>}

      <SimpleGrid

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

        spacing={3}

        padding="10px"

      >

        {isLoading &&

          skeletons.map((skeleton) => (

            <GameCardContainer key={skeleton}>

              <GameCardSkeleton key={skeleton} />

            </GameCardContainer>

          ))}

        {data.map((game) => (

          <GameCardContainer key={game.id}>

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

          </GameCardContainer>

        ))}

      </SimpleGrid>

    </>

  );

};

export default GameGrid;

In hooks folder, useGames.tsx

import useData from "./useData";

import { Genre } from "./useGenres";

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;

}

*//? second arguments is one of the properties of the AXIOS request config object. Sent*

*//? as a Query String. set params to an object. In this games hook, we are passing the selected*

*//? genre, as a query string parameter to the useData hook. As part of this lesson,k we had to*

*//? open up our hook and make it more flexible. Now we can pass query string parameters or*

*//? request data to our request objects. We also added an array of dependencies, if any of*

*//? these dependencies changes, our effect hook in the useData.ts will re-run and refresh the*

*//? data from the server.*

const useGames = (

  selectedGenre: Genre | null,

  selectedPlatform: Platform | null

) =>

  useData<Game>(

    "/games",

    {

      params: {

        genres: selectedGenre?.id,

        parent\_platforms: selectedPlatform?.id,

      },

    },

    [selectedGenre?.id, selectedPlatform?.id]

  );

export default useGames;

# Refactoring- Extracting a Query Object

In APP component

import { Grid, GridItem, Show } from "@chakra-ui/react";

import NavBar from "./components/NavBar";

import GameGrid from "./components/GameGrid";

import GenreList from "./components/GenreList";

import { useState } from "react";

import { Genre } from "./hooks/useGenres";

import PlatformSelector from "./components/PlatformSelector";

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

export interface GameQuery {

  genre: Genre | null;

  platform: Platform | null;

}

function App() {

  const [gameQuery, setGameQuery] = useState<GameQuery>({} as GameQuery);

  return (

    <Grid

      templateAreas={{

*/\* an object is passed here \*/*

        base: `"nav" "main"` */\* mobile phone වලට \*/*,

        lg: `"nav nav" "aside main"` */\* 1024px laptop වගේ ලොකු screen තියෙන ඒවට. \*/*,

      }}

      templateColumns={{

        base: "1fr",

        lg: "250px 1fr",

      }}

    >

      {*/\* navigation bar එක \*/*}

      <GridItem area="nav">

        <NavBar />

      </GridItem>

      {*/\* show එකෙන් large screen device වලට විතරක් aside grid item එක පේන්න හැදුවෑකි. \*/*}

      <Show above="lg">

        <GridItem area="aside" paddingX={5}>

          <GenreList

            selectedGenre={gameQuery.genre}

            onSelectGenre={(genre) => setGameQuery({ ...gameQuery, genre })}

          />

        </GridItem>

      </Show>

      {*/\* main area එක \*/*}

      <GridItem area="main">

        <PlatformSelector

          selectedPlatform={gameQuery.platform}

          onSelectPlatform={(platform) =>

            setGameQuery({ ...gameQuery, platform })

          }

        />

        <GameGrid gameQuery={gameQuery} />

      </GridItem>

    </Grid>

  );

}

export default App;

In components folder, GameGrid.tsx

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

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

import GameCard from "./GameCard";

import GameCardSkeleton from "./GameCardSkeleton";

import GameCardContainer from "./GameCardContainer";

import { Genre } from "../hooks/useGenres";

import { GameQuery } from "../App";

interface Props {

  gameQuery: GameQuery;

}

const GameGrid = ({ gameQuery }: Props) => {

  const { data, error, isLoading } = useGames(gameQuery);

  const skeletons = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];

  return (

    <>

      {error && <Text>{error}</Text>}

      <SimpleGrid

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

        spacing={3}

        padding="10px"

      >

        {isLoading &&

          skeletons.map((skeleton) => (

            <GameCardContainer key={skeleton}>

              <GameCardSkeleton key={skeleton} />

            </GameCardContainer>

          ))}

        {data.map((game) => (

          <GameCardContainer key={game.id}>

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

          </GameCardContainer>

        ))}

      </SimpleGrid>

    </>

  );

};

export default GameGrid;

In hooks folder, useGames.ts

import { GameQuery } from "../App";

import useData from "./useData";

import { Genre } from "./useGenres";

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;

}

*//? second arguments is one of the properties of the AXIOS request config object. Sent*

*//? as a Query String. set params to an object. In this games hook, we are passing the selected*

*//? genre, as a query string parameter to the useData hook. As part of this lesson,k we had to*

*//? open up our hook and make it more flexible. Now we can pass query string parameters or*

*//? request data to our request objects. We also added an array of dependencies, if any of*

*//? these dependencies changes, our effect hook in the useData.ts will re-run and refresh the*

*//? data from the server.*

const useGames = (gameQuery: GameQuery) =>

  useData<Game>(

    "/games",

    {

      params: {

        genres: gameQuery.genre?.id,

        parent\_platforms: gameQuery.platform?.id,

      },

    },

    [gameQuery]

  );

export default useGames;

# Building Sort Selector

In components folder, SortSelector.tsx

import { Button, Menu, MenuButton, MenuItem, MenuList } from "@chakra-ui/react";

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

const SortSelector = () => {

  return (

    <Menu>

      <MenuButton as={Button} rightIcon={<BsChevronDown />}>

        Order by: Relevance

      </MenuButton>

      <MenuList>

        <MenuItem>Relevance</MenuItem>

        <MenuItem>Date added</MenuItem>

        <MenuItem>Name</MenuItem>

        <MenuItem>Release date</MenuItem>

        <MenuItem>Popularity</MenuItem>

        <MenuItem>Average rating</MenuItem>

      </MenuList>

    </Menu>

  );

};

export default SortSelector;

In App component,

import { Grid, GridItem, HStack, Show } from "@chakra-ui/react";

import NavBar from "./components/NavBar";

import GameGrid from "./components/GameGrid";

import GenreList from "./components/GenreList";

import { useState } from "react";

import { Genre } from "./hooks/useGenres";

import PlatformSelector from "./components/PlatformSelector";

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

import SortSelector from "./components/SortSelector";

export interface GameQuery {

  genre: Genre | null;

  platform: Platform | null;

}

function App() {

  const [gameQuery, setGameQuery] = useState<GameQuery>({} as GameQuery);

  return (

    <Grid

      templateAreas={{

*/\* an object is passed here \*/*

        base: `"nav" "main"` */\* mobile phone වලට \*/*,

        lg: `"nav nav" "aside main"` */\* 1024px laptop වගේ ලොකු screen තියෙන ඒවට. \*/*,

      }}

      templateColumns={{

        base: "1fr",

        lg: "250px 1fr",

      }}

    >

      {*/\* navigation bar එක \*/*}

      <GridItem area="nav">

        <NavBar />

      </GridItem>

      {*/\* show එකෙන් large screen device වලට විතරක් aside grid item එක පේන්න හැදුවෑකි. \*/*}

      <Show above="lg">

        <GridItem area="aside" paddingX={5}>

          <GenreList

            selectedGenre={gameQuery.genre}

            onSelectGenre={(genre) => setGameQuery({ ...gameQuery, genre })}

          />

        </GridItem>

      </Show>

      {*/\* main area එක \*/*}

      <GridItem area="main">

        <HStack spacing={5} paddingLeft={2} marginBottom={5}>

          <PlatformSelector

            selectedPlatform={gameQuery.platform}

            onSelectPlatform={(platform) =>

              setGameQuery({ ...gameQuery, platform })

            }

          />

          <SortSelector />

        </HStack>

        <GameGrid gameQuery={gameQuery} />

      </GridItem>

    </Grid>

  );

}

export default App;

# Sorting Games

In components folder, SortSelector.tsx

import { Button, Menu, MenuButton, MenuItem, MenuList } from "@chakra-ui/react";

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

interface Props {

  onSelectSortOrder: (sortOrder: string) => void;

  selectedSortOrder: string;

}

const SortSelector = ({ onSelectSortOrder, selectedSortOrder }: Props) => {

  const sortOrders = [

    { value: "", label: "Relevance" },

    { value: "-added", label: "Date added" },

    { value: "name", label: "Name" },

    { value: "-released", label: "Release Date" },

    { value: "-metacritic", label: "Popularity" },

    { value: "-rating", label: "Average rating" },

  ];

  const currentSortOrder = sortOrders.find(

    (order) => order.value === selectedSortOrder

  );

  return (

    <Menu>

      <MenuButton as={Button} rightIcon={<BsChevronDown />}>

        Order by: {currentSortOrder?.label || "Relevance"}

      </MenuButton>

      <MenuList>

        {sortOrders.map((order) => (

          <MenuItem

            onClick={() => onSelectSortOrder(order.value)}

            key={order.value}

            value={order.value}

          >

            {order.label}

          </MenuItem>

        ))}

      </MenuList>

    </Menu>

  );

};

export default SortSelector;

In App component,

import { Grid, GridItem, HStack, Show } from "@chakra-ui/react";

import NavBar from "./components/NavBar";

import GameGrid from "./components/GameGrid";

import GenreList from "./components/GenreList";

import { useState } from "react";

import { Genre } from "./hooks/useGenres";

import PlatformSelector from "./components/PlatformSelector";

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

import SortSelector from "./components/SortSelector";

export interface GameQuery {

  genre: Genre | null;

  platform: Platform | null;

  sortOrder: string;

}

function App() {

  const [gameQuery, setGameQuery] = useState<GameQuery>({} as GameQuery);

  return (

    <Grid

      templateAreas={{

*/\* an object is passed here \*/*

        base: `"nav" "main"` */\* mobile phone වලට \*/*,

        lg: `"nav nav" "aside main"` */\* 1024px laptop වගේ ලොකු screen තියෙන ඒවට. \*/*,

      }}

      templateColumns={{

        base: "1fr",

        lg: "250px 1fr",

      }}

    >

      {*/\* navigation bar එක \*/*}

      <GridItem area="nav">

        <NavBar />

      </GridItem>

      {*/\* show එකෙන් large screen device වලට විතරක් aside grid item එක පේන්න හැදුවෑකි. \*/*}

      <Show above="lg">

        <GridItem area="aside" paddingX={5}>

          <GenreList

            selectedGenre={gameQuery.genre}

            onSelectGenre={(genre) => setGameQuery({ ...gameQuery, genre })}

          />

        </GridItem>

      </Show>

      {*/\* main area එක \*/*}

      <GridItem area="main">

        <HStack spacing={5} paddingLeft={2} marginBottom={5}>

          <PlatformSelector

            selectedPlatform={gameQuery.platform}

            onSelectPlatform={(platform) =>

              setGameQuery({ ...gameQuery, platform })

            }

          />

          <SortSelector

            selectedSortOrder={gameQuery.sortOrder}

            onSelectSortOrder={(sortOrder) =>

              setGameQuery({ ...gameQuery, sortOrder })

            }

          />

        </HStack>

        <GameGrid gameQuery={gameQuery} />

      </GridItem>

    </Grid>

  );

}

export default App;

In components folder, GameGrid.tsx

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

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

import GameCard from "./GameCard";

import GameCardSkeleton from "./GameCardSkeleton";

import GameCardContainer from "./GameCardContainer";

import { Genre } from "../hooks/useGenres";

import { GameQuery } from "../App";

interface Props {

  gameQuery: GameQuery;

}

const GameGrid = ({ gameQuery }: Props) => {

  const { data, error, isLoading } = useGames(gameQuery);

  const skeletons = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];

  return (

    <>

      {error && <Text>{error}</Text>}

      <SimpleGrid

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

        spacing={3}

        padding="10px"

      >

        {isLoading &&

          skeletons.map((skeleton) => (

            <GameCardContainer key={skeleton}>

              <GameCardSkeleton key={skeleton} />

            </GameCardContainer>

          ))}

        {data.map((game) => (

          <GameCardContainer key={game.id}>

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

          </GameCardContainer>

        ))}

      </SimpleGrid>

    </>

  );

};

export default GameGrid;

In hooks folder, useGames.ts

import { GameQuery } from "../App";

import useData from "./useData";

import { Genre } from "./useGenres";

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;

}

*//? second arguments is one of the properties of the AXIOS request config object. Sent*

*//? as a Query String. set params to an object. In this games hook, we are passing the selected*

*//? genre, as a query string parameter to the useData hook. As part of this lesson,k we had to*

*//? open up our hook and make it more flexible. Now we can pass query string parameters or*

*//? request data to our request objects. We also added an array of dependencies, if any of*

*//? these dependencies changes, our effect hook in the useData.ts will re-run and refresh the*

*//? data from the server.*

const useGames = (gameQuery: GameQuery) =>

  useData<Game>(

    "/games",

    {

      params: {

        genres: gameQuery.genre?.id,

        parent\_platforms: gameQuery.platform?.id,

        ordering: gameQuery.sortOrder,

      },

    },

    [gameQuery]

  );

export default useGames;

# Handling Games without an Image

In services folder, image-url.ts

import noImage from "../assets/GameHub Resources/Image Placeholder/no-image-placeholder-6f3882e0.webp";

const getCroppedImageUrl = (url: string) => {

  if (!url) return noImage;

  const target = "media/";

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

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

};

export default getCroppedImageUrl;

# Fixing the Issue with Chakra Menus

In App component,

import { Box, Flex, Grid, GridItem, HStack, Show } from "@chakra-ui/react";

import NavBar from "./components/NavBar";

import GameGrid from "./components/GameGrid";

import GenreList from "./components/GenreList";

import { useState } from "react";

import { Genre } from "./hooks/useGenres";

import PlatformSelector from "./components/PlatformSelector";

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

import SortSelector from "./components/SortSelector";

export interface GameQuery {

  genre: Genre | null;

  platform: Platform | null;

  sortOrder: string;

}

function App() {

  const [gameQuery, setGameQuery] = useState<GameQuery>({} as GameQuery);

  return (

    <Grid

      templateAreas={{

*/\* an object is passed here \*/*

        base: `"nav" "main"` */\* mobile phone වලට \*/*,

        lg: `"nav nav" "aside main"` */\* 1024px laptop වගේ ලොකු screen තියෙන ඒවට. \*/*,

      }}

      templateColumns={{

        base: "1fr",

        lg: "250px 1fr",

      }}

    >

      {*/\* navigation bar එක \*/*}

      <GridItem area="nav">

        <NavBar />

      </GridItem>

      {*/\* show එකෙන් large screen device වලට විතරක් aside grid item එක පේන්න හැදුවෑකි. \*/*}

      <Show above="lg">

        <GridItem area="aside" paddingX={5}>

          <GenreList

            selectedGenre={gameQuery.genre}

            onSelectGenre={(genre) => setGameQuery({ ...gameQuery, genre })}

          />

        </GridItem>

      </Show>

      {*/\* main area එක \*/*}

      <GridItem area="main">

        <Flex paddingLeft={2} marginBottom={5}>

          <Box marginRight={5}>

            <PlatformSelector

              selectedPlatform={gameQuery.platform}

              onSelectPlatform={(platform) =>

                setGameQuery({ ...gameQuery, platform })

              }

            />

          </Box>

          <SortSelector

            selectedSortOrder={gameQuery.sortOrder}

            onSelectSortOrder={(sortOrder) =>

              setGameQuery({ ...gameQuery, sortOrder })

            }

          />

        </Flex>

        <GameGrid gameQuery={gameQuery} />

      </GridItem>

    </Grid>

  );

}

export default App;

# Building Search Input

In components, SearchInput.tsx

import { Input, InputGroup, InputLeftElement } from "@chakra-ui/react";

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

const SearchInput = () => {

  return (

    <InputGroup>

      <InputLeftElement children={<BsSearch />} />

      <Input

        borderRadius={20}

        placeholder="Search games...."

        variant="filled"

      />

    </InputGroup>

  );

};

export default SearchInput;

In components, NavBar.tsx

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

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

import ColorModeSwitch from "./ColorModeSwitch";

import SearchInput from "./SearchInput";

const NavBar = () => {

  return (

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

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

    <HStack padding="10px">

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

      <SearchInput />

      <ColorModeSwitch />

    </HStack>

  );

};

export default NavBar;

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 whiteSpace="nowrap">Dark Mode</Text>

    </HStack>

  );

};

export default ColorModeSwitch;

# Searching Games

In components folder, SearchInput.tsx

import { Input, InputGroup, InputLeftElement } from "@chakra-ui/react";

import { useRef } from "react";

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

interface Props {

  onSearch: (searchText: string) => void;

}

const SearchInput = ({ onSearch }: Props) => {

  const ref = useRef<HTMLInputElement>(null);

  return (

    <form

      onSubmit={(event) => {

        event.preventDefault();

        if (ref.current) onSearch(ref.current.value);

      }}

    >

      <InputGroup>

        <InputLeftElement children={<BsSearch />} />

        <Input

          ref={ref}

          borderRadius={20}

          placeholder="Search games...."

          variant="filled"

        />

      </InputGroup>

    </form>

  );

};

export default SearchInput;

In comoponents folder, NavBar.tsx

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

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

import ColorModeSwitch from "./ColorModeSwitch";

import SearchInput from "./SearchInput";

interface Props {

  onSearch: (searchText: string) => void;

}

const NavBar = ({ onSearch }: Props) => {

  return (

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

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

    <HStack padding="10px">

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

      <SearchInput onSearch={onSearch} />

      <ColorModeSwitch />

    </HStack>

  );

};

export default NavBar;

In App component,

import { Box, Flex, Grid, GridItem, HStack, Show } from "@chakra-ui/react";

import NavBar from "./components/NavBar";

import GameGrid from "./components/GameGrid";

import GenreList from "./components/GenreList";

import { useState } from "react";

import { Genre } from "./hooks/useGenres";

import PlatformSelector from "./components/PlatformSelector";

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

import SortSelector from "./components/SortSelector";

export interface GameQuery {

  genre: Genre | null;

  platform: Platform | null;

  sortOrder: string;

  searchText: string;

}

function App() {

  const [gameQuery, setGameQuery] = useState<GameQuery>({} as GameQuery);

  return (

    <Grid

      templateAreas={{

*/\* an object is passed here \*/*

        base: `"nav" "main"` */\* mobile phone වලට \*/*,

        lg: `"nav nav" "aside main"` */\* 1024px laptop වගේ ලොකු screen තියෙන ඒවට. \*/*,

      }}

      templateColumns={{

        base: "1fr",

        lg: "250px 1fr",

      }}

    >

      {*/\* navigation bar එක \*/*}

      <GridItem area="nav">

        <NavBar

          onSearch={(searchText) => setGameQuery({ ...gameQuery, searchText })}

        />

      </GridItem>

      {*/\* show එකෙන් large screen device වලට විතරක් aside grid item එක පේන්න හැදුවෑකි. \*/*}

      <Show above="lg">

        <GridItem area="aside" paddingX={5}>

          <GenreList

            selectedGenre={gameQuery.genre}

            onSelectGenre={(genre) => setGameQuery({ ...gameQuery, genre })}

          />

        </GridItem>

      </Show>

      {*/\* main area එක \*/*}

      <GridItem area="main">

        <Flex paddingLeft={2} marginBottom={5}>

          <Box marginRight={5}>

            <PlatformSelector

              selectedPlatform={gameQuery.platform}

              onSelectPlatform={(platform) =>

                setGameQuery({ ...gameQuery, platform })

              }

            />

          </Box>

          <SortSelector

            selectedSortOrder={gameQuery.sortOrder}

            onSelectSortOrder={(sortOrder) =>

              setGameQuery({ ...gameQuery, sortOrder })

            }

          />

        </Flex>

        <GameGrid gameQuery={gameQuery} />

      </GridItem>

    </Grid>

  );

}

export default App;

In hooks folder, useGames.ts

import { GameQuery } from "../App";

import useData from "./useData";

import { Genre } from "./useGenres";

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;

}

*//? second arguments is one of the properties of the AXIOS request config object. Sent*

*//? as a Query String. set params to an object. In this games hook, we are passing the selected*

*//? genre, as a query string parameter to the useData hook. As part of this lesson,k we had to*

*//? open up our hook and make it more flexible. Now we can pass query string parameters or*

*//? request data to our request objects. We also added an array of dependencies, if any of*

*//? these dependencies changes, our effect hook in the useData.ts will re-run and refresh the*

*//? data from the server.*

const useGames = (gameQuery: GameQuery) =>

  useData<Game>(

    "/games",

    {

      params: {

        genres: gameQuery.genre?.id,

        parent\_platforms: gameQuery.platform?.id,

        ordering: gameQuery.sortOrder,

        search: gameQuery.searchText,

      },

    },

    [gameQuery]

  );

export default useGames;

In index.css

form {

  width: 100%;

}

# Adding a Dynamic Heading

In components folder, GameHeading.tsx

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

import { GameQuery } from "../App";

interface Props {

  gameQuery: GameQuery;

}

const GameHeading = ({ gameQuery }: Props) => {

  const heading = `${gameQuery.platform?.name || ""} ${

    gameQuery.genre?.name || ""

  } Games`;

  return (

    <Heading as="h1" marginY={5} fontSize="5xl">

      {heading}

    </Heading>

  );

};

export default GameHeading;

In App component,

import { Box, Flex, Grid, GridItem, HStack, Show } from "@chakra-ui/react";

import NavBar from "./components/NavBar";

import GameGrid from "./components/GameGrid";

import GenreList from "./components/GenreList";

import { useState } from "react";

import { Genre } from "./hooks/useGenres";

import PlatformSelector from "./components/PlatformSelector";

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

import SortSelector from "./components/SortSelector";

import GameHeading from "./components/GameHeading";

export interface GameQuery {

  genre: Genre | null;

  platform: Platform | null;

  sortOrder: string;

  searchText: string;

}

function App() {

  const [gameQuery, setGameQuery] = useState<GameQuery>({} as GameQuery);

  return (

    <Grid

      templateAreas={{

*/\* an object is passed here \*/*

        base: `"nav" "main"` */\* mobile phone වලට \*/*,

        lg: `"nav nav" "aside main"` */\* 1024px laptop වගේ ලොකු screen තියෙන ඒවට. \*/*,

      }}

      templateColumns={{

        base: "1fr",

        lg: "250px 1fr",

      }}

    >

      {*/\* navigation bar එක \*/*}

      <GridItem area="nav">

        <NavBar

          onSearch={(searchText) => setGameQuery({ ...gameQuery, searchText })}

        />

      </GridItem>

      {*/\* show එකෙන් large screen device වලට විතරක් aside grid item එක පේන්න හැදුවෑකි. \*/*}

      <Show above="lg">

        <GridItem area="aside" paddingX={5}>

          <GenreList

            selectedGenre={gameQuery.genre}

            onSelectGenre={(genre) => setGameQuery({ ...gameQuery, genre })}

          />

        </GridItem>

      </Show>

      {*/\* main area එක \*/*}

      <GridItem area="main">

        <Box paddingLeft={2}>

          <GameHeading gameQuery={gameQuery} />

          <Flex marginBottom={5}>

            <Box marginRight={5}>

              <PlatformSelector

                selectedPlatform={gameQuery.platform}

                onSelectPlatform={(platform) =>

                  setGameQuery({ ...gameQuery, platform })

                }

              />

            </Box>

            <SortSelector

              selectedSortOrder={gameQuery.sortOrder}

              onSelectSortOrder={(sortOrder) =>

                setGameQuery({ ...gameQuery, sortOrder })

              }

            />

          </Flex>

        </Box>

        <GameGrid gameQuery={gameQuery} />

      </GridItem>

    </Grid>

  );

}

export default App;

# Cleaning Up the Genres

In components folder, GenreList.tsx

import {

  Button,

  HStack,

  Heading,

  Image,

  List,

  ListItem,

  Spinner,

  Text,

} from "@chakra-ui/react";

import useGenres, { Genre } from "../hooks/useGenres";

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

interface Props {

  onSelectGenre: (genre: Genre) => void;

  selectedGenre: Genre | null;

}

const GenreList = ({ onSelectGenre, selectedGenre }: Props) => {

  const { data, isLoading, error } = useGenres();

  if (error) return null;

  if (isLoading) return <Spinner />;

  return (

    <>

      <Heading fontSize="2xl" marginBottom={3}>

        Genres

      </Heading>

      <List>

        {data.map((genre) => (

          <ListItem key={genre.id} paddingY="5px">

            <HStack>

              <Image

                objectFit="cover"

                boxSize="32px"

                borderRadius={8}

                src={getCroppedImageUrl(genre.image\_background)}

              />

              <Button

                whiteSpace="normal"

                textAlign="left"

                fontWeight={genre.id === selectedGenre?.id ? "bold" : "normal"}

                onClick={() => {

                  onSelectGenre(genre);

                  console.log(genre);

                }}

                fontSize="lg"

                variant="link"

              >

                {genre.name}

              </Button>

            </HStack>

          </ListItem>

        ))}

      </List>

    </>

  );

};

export default GenreList;

# Cleaning Up the Game Cards

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>

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

      <CardBody>

        <HStack justifyContent="space-between" marginBottom={3}>

          <PlatformIconList

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

          />

          <CriticScore score={game.metacritic} />

        </HStack>

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

      </CardBody>

    </Card>

  );

};

export default GameCard;

GameGrid.tsx

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

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

import GameCard from "./GameCard";

import GameCardSkeleton from "./GameCardSkeleton";

import GameCardContainer from "./GameCardContainer";

import { Genre } from "../hooks/useGenres";

import { GameQuery } from "../App";

interface Props {

  gameQuery: GameQuery;

}

const GameGrid = ({ gameQuery }: Props) => {

  const { data, error, isLoading } = useGames(gameQuery);

  const skeletons = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];

  return (

    <>

      {error && <Text>{error}</Text>}

      <SimpleGrid

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

        spacing={6}

        padding="10px"

      >

        {isLoading &&

          skeletons.map((skeleton) => (

            <GameCardContainer key={skeleton}>

              <GameCardSkeleton key={skeleton} />

            </GameCardContainer>

          ))}

        {data.map((game) => (

          <GameCardContainer key={game.id}>

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

          </GameCardContainer>

        ))}

      </SimpleGrid>

    </>

  );

};

export default GameGrid;

# Adding Emojis

In useGames/tsx

import { GameQuery } from "../App";

import useData from "./useData";

import { Genre } from "./useGenres";

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;

  rating\_top: number;

}

*//? second arguments is one of the properties of the AXIOS request config object. Sent*

*//? as a Query String. set params to an object. In this games hook, we are passing the selected*

*//? genre, as a query string parameter to the useData hook. As part of this lesson,k we had to*

*//? open up our hook and make it more flexible. Now we can pass query string parameters or*

*//? request data to our request objects. We also added an array of dependencies, if any of*

*//? these dependencies changes, our effect hook in the useData.ts will re-run and refresh the*

*//? data from the server.*

const useGames = (gameQuery: GameQuery) =>

  useData<Game>(

    "/games",

    {

      params: {

        genres: gameQuery.genre?.id,

        parent\_platforms: gameQuery.platform?.id,

        ordering: gameQuery.sortOrder,

        search: gameQuery.searchText,

      },

    },

    [gameQuery]

  );

export default useGames;

In Emoji.tsx

import bullsEye from "../assets/GameHub Resources/Emojis/bulls-eye.webp";

import thumbsUp from "../assets/GameHub Resources/Emojis/thumbs-up.webp";

import meh from "../assets/GameHub Resources/Emojis/meh.webp";

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

interface Props {

  rating: number;

}

const Emoji = ({ rating }: Props) => {

  if (rating < 3) return null;

  const emojiMap: { [key: number]: ImageProps } = {

    3: { src: meh, alt: "meh", boxSize: "25px" },

    4: { src: thumbsUp, alt: "recommended", boxSize: "25px" },

    5: { src: bullsEye, alt: "exceptional", boxSize: "35px" },

  };

  return <Image {...emojiMap[rating]} marginTop={1} />;

};

export default Emoji;

In 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";

import Emoji from "./Emoji";

interface Props {

  game: Game;

}

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

  return (

    <Card>

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

      <CardBody>

        <HStack justifyContent="space-between" marginBottom={3}>

          <PlatformIconList

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

          />

          <CriticScore score={game.metacritic} />

        </HStack>

        <Heading fontSize="2xl">

          {game.name}

          <Emoji rating={game.rating\_top} />

        </Heading>

      </CardBody>

    </Card>

  );

};

export default GameCard;

# Shipping Static Data

Make directory data and add genres.ts file and result from the server and paste it here.

In useGames.ts hook

import genres from "../data/genres";

export interface Genre {

  id: number;

  name: string;

  image\_background: string;

}

const useGenres = () => ({ data: genres, isLoading: false, error: null });

export default useGenres;

# Customizing the Chakra Theme

In theme.ts

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

const config: ThemeConfig = {

  initialColorMode: "dark",

};

const theme = extendTheme({

  config,

  colors: {

    gray: {

      50: "#f9f9f9",

      100: "#ededed",

      200: "#d3d3d3",

      300: "#b3b3b3",

      400: "#a0a0a0",

      500: "#898989",

      600: "#6c6c6c",

      700: "#202020",

      800: "#121212",

      900: "#111",

    },

  },

});

export default theme;

# Refactoring Game Grid

In GameGrid.tsx

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

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

import GameCard from "./GameCard";

import GameCardSkeleton from "./GameCardSkeleton";

import GameCardContainer from "./GameCardContainer";

import { Genre } from "../hooks/useGenres";

import { GameQuery } from "../App";

interface Props {

  gameQuery: GameQuery;

}

const GameGrid = ({ gameQuery }: Props) => {

  const { data, error, isLoading } = useGames(gameQuery);

  const skeletons = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15];

  if (error) return <Text>{error}</Text>;

  return (

    <SimpleGrid

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

      spacing={6}

      padding="10px"

    >

      {isLoading &&

        skeletons.map((skeleton) => (

          <GameCardContainer key={skeleton}>

            <GameCardSkeleton key={skeleton} />

          </GameCardContainer>

        ))}

      {data.map((game) => (

        <GameCardContainer key={game.id}>

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

        </GameCardContainer>

      ))}

    </SimpleGrid>

  );

};

export default GameGrid;

# Building for Production

To build locally: npm run build

Shortcut => Ctrl + Shift + B

# Deploying to Vercel

Make a GIT repository.

Create a Vercel account.

Install vercel CLI => npm I -g vercel

To deploy, just run vercel in CMD and answer the questions.

Link the project to GITHUB repository for real-time updates. Once you commit vercel gets the update.