PokeDex:Red App

Jacob Westerback

Application Definition

A Pokedex that will provide information on the various pokemon in the game “Pokemon Red” to players. This is a game companion app. I plan to use the Pokeapi, https://pokeapi.co/

About this app

Features marked in red go beyond the standard requirements

- Appealing and Simple UI
  - respects the iOS Mobile HIG
  - Clear titles, labeling and icons
  - Content focused
  - Vibrant and Immersive
- Pokemon 1-151
  - Pokemon Info: Name, Number, Type, Abilities, Gender Ratio, Catch Rate, Egg Group, Hatch Time, Height, Weight, Pokedex number region, Experience yield, leveling rate, Base Friendship, type effectiveness, evolution(s), sprite, description of pokemon
  - Uses segmented control and stacks to display data
  - Search the dex by name
  - Pokemon details can be viewed from all pokemon lists
  - Evolutions are calculated to be able to handle complex trees as is the case with #133 Evee
- The user can favorite Pokemon
  - And view their favorites
  - Favorites are saved to the device
  - If the pokemon is already favorited then this becomes an Unfavorite button. When tapped it removes the pokemon from the favorites list
  - Pokemon can only be favorited once.
- PokeData singleton handles the apps data
  - Handles Teams
  - Handles the Dex
  - PokeData handles calculations for Damage Taken (weaknesses, strengths, imunity)
- Pokemon are always sorted by number for user convenience

- The user can save up to 6 “teams” of pokemon
  - Pokemon can be added to multiple teams and more than once to a single team
    - Uses picker to select pokemon
  - Pokemon can easily be rearranged on teams
  - Pokemon can be deleted from teams
  - Teams are saved to the device
- API - https://pokeapi.co/
  - Information about the pokemon is received from the API
  - Calls are made when a pokemon is selected
  - This pokemon is now “downloaded” and the information is saved
    - This means that if you have already looked at a pokemon it can now be available offline
    - The data also does not change, so saving this data provides a better, faster, user experience
  - Due to the nature of the app 3 calls to the API are required for each pokemon
    - Huge amount of data to parse through and display
    - Data is displayed in an organized manner so the sheer amount is not so overwhelming to the user
- The Pokemon and Evolution class conform to the NSCoding protocol. The contents of the pokemon are serialized and saved to disk, and then loaded in whenever the app starts up if they are present
- The About screen displays my name and the name of the app. As well as other resources used.
- The app has a custom launch screen that displays a loading screen.
- The app has custom app icons in all of the required sizes.
  - The app also has custom icons for the tab bar
- The app functions on both iPhone and iPad, in portrait orientation.
- The code follows all code conventions

Struggles

Evolution Trees

This was a particular challenge for me. Due to how the information was stored in the API, figuring how to parse it into what I wanted to display was quite confusing. The API provides information overload at every step. Ultimately I solved the issue by creating a class to use to specifically store the child-parent relationships on top of a lot of trial and error.

Damage Taken

This was also something that posed a problem to be solved. Given the types of the pokemon I had to create functions that would return the resistance lists. This was a challenge as navigating the matrix and adding up the rows was not as straightforward as it might seem. For one I had to enter by hand the matrix from the chart on bulbapedia. Then I had to figure out what math needed to be done to cancel our resistance on multi-typed pokemon.

API Calls

In order to get the initial list of pokemon, and their types and their sprites, I would need to do 3 API calls per pokemon when the app stars. This works out to about 3000 API calls with the full pokemon list. The API blocks you if you try to do this. To work around this I hand coded a dex JSON list to start with and only called on the API once a pokemon was selected. Because of this I stuck to the original 151 pokemon present in Pokemon Red.

Resources

Icons

I used icons created by Icons8. I used his pokeball icon to create a team icon in photoshop.

Code

All code was written by me unless otherwize stated. I referenced various resources online as well as in-class ICEs. I learned how to use the UIPicker here. SwiftyJSON was pulled from an ICE. Image downloading solution found on Stack Overflow. I also used the Pokeapi. The two display warnings were discussed in class and no solution was found, they were deemed not important enough to worry about and were excused.

Grade

I feel that I deserve at least a 95 on this project. I completed all the requirements and went above and beyond. I worked really hard on this and put a ton of time into this project. I am proud of the result. I create an app that cleanly displays useful information to a specific audience.