GSDEX - Pokemon TeamBuilder

A Pokemon Teambuilder Android App

Built by Kyle Hackett
CS443 - Mobile Applications Term Project
Professor Sheng

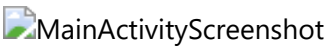
Table of Contents here later 😉

Project Statement

There are many different pokedex (pokemon encyclopedia) apps on the market. The intent of this iteration was to surpass the functionality of those applications. I wanted to combine the concept of the traditional pokedex app with the pokemon team builder web apps scattered across the internet, but with an added twist. I wanted to make finding pokemon that fit your team as easy as possible, as well as provide feedback to the user on their team's weaknesses. I have not seen too many teambuilders out there that have this functionality ***and*** have a user friendly experience fit for casual or advanced players.
This application pulls data from an external database, and stores that data locally. The app's speed comes from it being able to make quick queries to its local database to sort and filter pokemon by name, number, types, abilities, and a number of different data points. Users can then scroll through the desired data and select pokemon to add to their team. I hope that this application can be the go to for users who want to build their next team with all the information they need all in one place.

Application Design

MainActivity



Main View where the user will perform most of their actions. Users can access the other Activities as well as manage their team by either adding or removing members. Team members can be added to their team by clicking on their image in the list, which can be traversed either by search box or by scrolling through. The search box allows users to search for a particular pokemon by name, or by ability. Clicking a species' image in the team will remove that member. Filter and Sorting options are availible by clicking the Filter/Sort Button. Clicking on a Pokemon Species will show the user a menu with additional info.

Pokemon Details

This Activity displays additional information to the user about the specified Pokemon, including Name, Types, Stats, and Abilities.

Sort & Filtering Options

Numerous Searching and Filtering options are displayed to the user. The top spinner provide sorting options,

including by name, ID number, or statistic. You can also order them in descending or ascending order by using the switch next to the sort options. The rest of the options on the page are for filtering. You can filter by a pokemon's generation of origin, or by type by toggling your choice.

Implementation & Evaluation

Implementation

After mapping out the design and what features I wanted included in the project, I thought about what kinds of data structures would be handy. What came to mind was to develop the Pokemon java class, which allows for each Pokemon pulled from the API to be generated into an object for easy reference. I then thought about how to best keep track of selected team members. Which I decided to be a double ended queue, for quick insertion, as well as deletion from either end of the double ended queue for efficiency. Once those were decided, I quickly built the Pokemon java class and began working on the MainActivity.

The first thing I wanted to do in the Main Activity was get each pokemon and its data from the external RESTful PokeAPI. I wrote the code to get the JSON file from the HTTP request, and then parsed that JSON to generate a new Pokemon object for each pokemon requested from the API.

Initially, I planned on just storing each Pokemon in a HashMap, with its ID number as the key and the pokemon as the value, but then I realized using a SQLite Database would not only make searching, sorting, and filtering much easier later on with SQL queries, but would also allow the data downloaded from the API to be persistant between sessions on the app. So then I built a bare bones SQLite Database Handler, and wrote the code to initilize the "pokedex" table with new pokemon object.

The MainActivity's UI was then implemented. I first structured how I initially envisioned the apps layout to be. Then I got to work on getting the values in the Pokedex SQLite Database to the scrollable ListView. In order to do this, I had to build my own ListView adapter class, since I was working with a structure that the default ListView adapter is not used to handling. I was able to load up the entire database to the ListView portion of the screen, but there were so many entries that it was hard to use without any sort or filter options.

Before I worked on sorting, I decided to implement the team mechanics. I wanted to make sure that was completed so that once I started to build the other activities, there would be less bugs coming from the MainActivity. The add to team functionality was easy enough, but the deletion was a little more difficult because of the different threads I was using. I was able to get around this by using a data structure that gets around concurrency errors when trying to remove a pokemon from the middle of the team. I then implemented a method that updates the UI by looking through the team and finding the Resource images that share the same name as the Pokemon, and then setting the ImageView's image to that Resource ID.

I then did some testing to try and find any bugs in my algorithms and I immediatly fixed them so then I could move onto the next function: sorting and filtering. To do this, I began by adding more methods to the Database Handler class I had created. These methods allowed for the the new activity to send a query string to the handler, and it would then return a new object which can then be used with the custom ArrayAdapter to update the ListView.

The next step was to create the layout for the sorting activity. I used a combination of toggleButtons, spinners, and switches to give the user options to sort and filter by. Upon the activity's close the choices will be put into a SQL string which is recieved by the sort/filter method previously created in the Database Handler.

Next, I created the PokemonActivity to display more of a specific Pokemon's data to the user. I went back to the ListView implementation and added a clickListener so that when a specific row is clicked, that pokemon is then bundled and sent to the PokemonActivity as an argument, and then is parsed to the display.

The last thing I implemented was another database handler. I thought it would be nice for the user to be able to close the app and come back to the same team they were workshopping from their last session. So anytime a member is added or removed from the team, the Team Database Handler is updated accordingly.

A number of users tesedt out the app and recorded any bugs they found. I then took that list of bugs and tried to generate fixes for them, and then have the users try it again, until I decided the application was complete.

Testing Methods

The app was strenuously tested by trying to predict undesirable outcomes that a user could possibly preform, such as trying to add more than six pokemon to a team, removing team members that do not exist, filters that result in zero results, SQL strings in the searchbox, and many more. When such tests failed they were noted and were immediately solved.
Perfomance was tested by running as many operations at once as possible, such as a sort/filter and then immediately adding and removing team members. There were times where this did slow down performance, and when that occured I looked to see if adding an additional thread would be appropriate, and that usually solved the slowdown.
The application's usability was tested by a couple of test subjects on different android virtual devices. Having users try out the app on different devices allowed me to see where different areas of the application needed touching up, whether it was touch box size, a clunky UI, or even just font size. The testers reported the application to be clean, efficient, and useful.

Known Remaining Bugs

There is a situation where the ListView and database will contain duplicates of some of the Pokemon. This bug occured once and has yet to be replicated, so it is unknown if it still exists.

References

This Project would not have been possible without the PokeAPI. Which is a RESTful API that was created to be a single source of data for people wishing to use data on all currently existing Pokemon. When called for a given pokemon name or ID, the API returns JSON objects containing the statistics and relevant data points of that pokemon.

The images used for this project were compiled by github user *msikma* and their project PokeSprite.

Experience and Thoughts

This project was a great learning experience. I got the chance to experiment with integrating SQL queries into Java, RESTful API calls, JSON parsing, as well as many different multithreading techniques. Having an object that could be affected by concurrent threads made implementation complicated, but very rewarding. If I had more time I would have liked to implement more user features, such as a team weakness calculator, team exporting, and add more data points that the user can filter and sort by. I will most likely continue this project

in my free time and implement the features that I did not have time to add. Overall, this project and this class both taught me a lot about Mobile Application Development on the Android platform.

Install Instructions

Requirements

API level 24 or above required for full functionality.

Installation

Android Studios

Download zip file and extract. Open in Android Studios and run on your favorite android virtual device.

Android Sideload

Android Sideload is not directly supported but with the .apk is is possible.

Windows 11 Sideload

Windows 11 sideload is not directly supported but with the .apk it is possible.

License