GitHub Username: mndiv

GitSearchRepo

Description

GitSearchRepo is a tool to search the github repositories based on user search query and then sorts them by Starred, Forked and Recently updated Repositories. Using this tool User can:

- View Most Starred/Fored/Updated Repositories
- Sort Repositories in Ascending / descending order
- Select All / a Language from settings
- Loads 30 repositories
- Access the details of project repository like Avatar, Owner Name, Repo Url, Created, Updated, Issues and shared repository link .
- Watch/Star/Fork a repository in Detailed View of repository
- Access Source Code of the selected project in a browser

Intended User

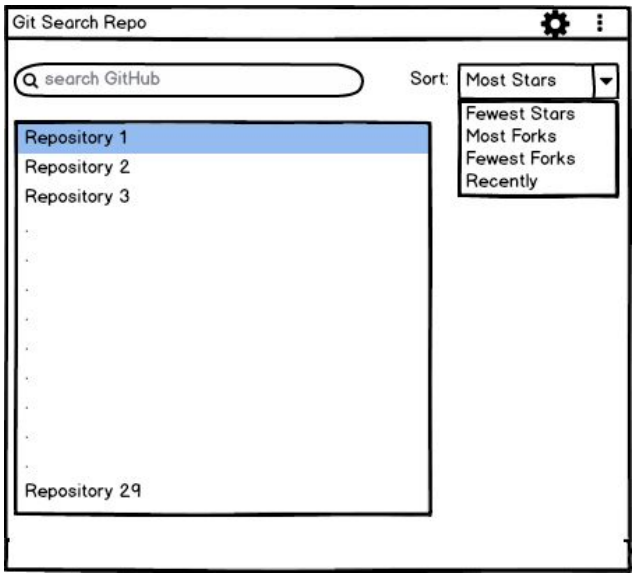
All Developers using Github

Features

- Search Github based on query
- Sort repositories based on Starred/forked/updated
- Has option to select repositories from different languages
- View details of selected repository like - Avatar, Owner Name, Repo Url, Created, Updated, Issues and shared repository link .
- Option to view selected Repository in browser

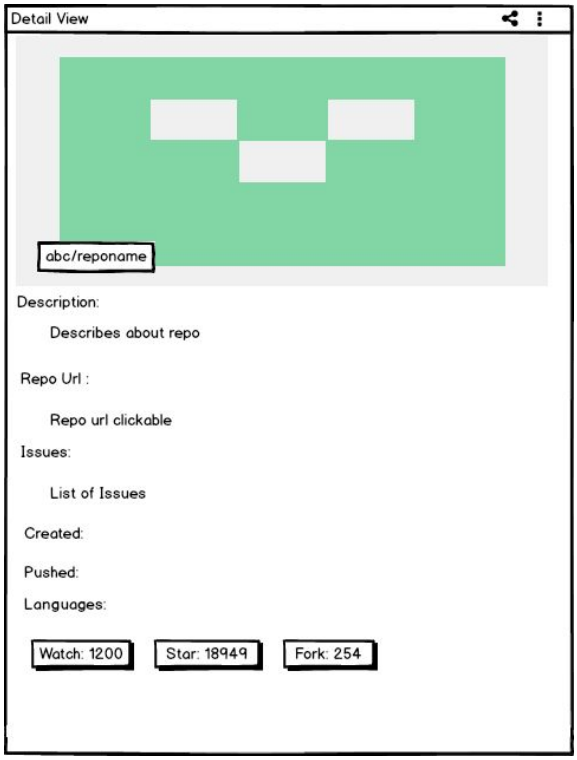
User Interface Mocks

MainActivity



- Search Button searches the Github based on user search query.
- SortBy option sorts the repositories based on Most Stars, Fewest Stars, Most Forks, Fewest Forks, Recently Updated. Settings has option to select Languages (like C++, JavaScript, Ruby.. etc) or all languages.
- By Default this activity shows List of repositories with Most Stars for all languages

DetailActivity



- Displays the details of Repository like Avatar, Full Name, Description, Repo Url, Issues, Created, Pushed, Languages, Watch Count, Fork Count and Star Count
- Share button shares the repo url via selected app (like whatsapp, google+ etc.,)
- On Clicking Repo Url , it loads the repo in browser
- Watch, Star & Fork buttons displays the count for the repo

Key Considerations

How will your app handle data persistence?

Content Provider is provided to handle data and it stores up to 30 repositories of data.

Describe any corner cases in the UX.

Describe any libraries you'll be using and share your reasoning for including them

- Picasso to handle the loading and caching of images

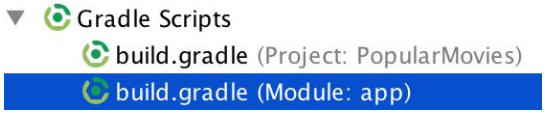
Next Steps: Required Tasks

Task 1: Project Setup

- **Image Library - Picasso**

How to Setup Picasso

We recommend that this project use Picasso, a powerful library that will handle image loading and caching on your behalf. If you prefer, you're welcome to use an alternate library such asGlide.
We've included this to reduce unnecessary extra work and help you focus on applying your app development skills.
You'll need to modify the build.gradle file for your app. These modifications will happen in the build.gradle file for your module's directory, *not* the project root directory (it is the file highlighted in blue in the screenshot below).



In your app/build.gradle file, add:

```
repositories {
    mavenCentral()
}
```

Next, add **compile 'com.squareup.picasso:picasso:2.5.2'** to your dependencies block.

Using Picasso To Fetch Images and Load Them Into Views

You can use Picasso to easily load album art thumbnails into your views using:
`Picasso.with(context).load("http://i.imgur.com/DvpvklR.png").into(imageView);`
Picasso will handle loading the images on a background thread, image decompression and caching the images.

- **Working with the github.com API**

You will notice that the API response provides search results of based on query, sort and order. It's constructed using 3 parameters:

Name	Type	Description
`q`	`string`	The search keywords, as well as any qualifiers
`sort`	`string`	The sort field. One of `stars, forks or updated.` Default: results are sorted by best match
`order`	`string`	The sort order if `sort` parameter is provided. One of asc or desc. Default: desc

The `q` search term can also contain any combination of the supported repository search qualifiers as described by the in-browser repository search documentation and search syntax documentation:

Combining these three parts gives us a final url for Most Stars projects
https://api.github.com/search/repositories?q=stars:>1

This is also explained explicitly in the API documentation for /search/repo.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for Main Activity
- Build UI for Detail Activity
- Build UI for Settings Activity

Task 3: Implement the Content Provider

- Fetch the first 30 repositories json data and store in Content Provider
- Content Provider contains fields like FullName, avatar, Repo Url, Stargazer Count, WatchGazer Count, forks, Open Issues, Created , Pushed, Languages, and description

Task 4: Implement Adapter

- Create custom list layout to populate list View of repositories

Task 5: Implementing SyncAdapter

- SyncAdapter gets latest Most Star Repositories when the activity is launched.