GSoC ‘18 Proposal for OpenFoodFacts Offline Android App

(Link to the latest version of the Proposal)

Name and Contact Information:

- **Name:** Samagra Sharma
- **Email**: samagra14@gmail.com
- **Phone Number**: +91 910957 8251
- **Slack Chat Room**: samagra
- **Github :** samagra14
- **Skype :** live:samagra14
- **Timezone :** UTC +05:30
- **Blog** (Non technical)

University and Current Enrollment:

- **University**: Indian Institute of Technology Roorkee.
- **Field of Study**: Computer Science and Engineering (sophomore)
- **Expected Graduation Year:** 2020
- **Degree:** Bachelor of Technology

Meeting with mentors

- Before May 14:
  - Available anytime between 3:30 am to 7:30 pm UTC (9:00 am to 1:00 am IST) through Email/Gitter.
  - Pre Planned meetings if required, any time during weekends.

- After May 14:
  - Available most of the time except while sleeping.
  - Typical waking hours: 4:30 am to 8:30 pm UTC (10:00 am to 2:00 am IST)

Coding Skills

I am a sophomore at IIT Roorkee (one of the topmost institutions of India). I am an Android Developer at Mobile Development Group IITR. I have contributed to some major and successful projects in Java such as Trianglify and to the source code of

Stuart Russell and Peter Norvig's book (AI a Modern approach). Besides this, I have a plethora of different projects on my Github account. I have been a regular contributor to Open Source since 2017.

Apart from the main requirement of this project - Java, I am fluent in Python and Cython. Alongwith these, I have an experience in Full Stack Development (Flask, Django and Angular).

Programming Languages and Frameworks

- JAVA and native Android App Development
- Object Oriented Programming(OOP), MVVM, MVP, Dependency Injection
- C/C++, Python, JavaScript - moderate knowledge
- Frameworks - Angular, Flask
- Databases - SQLite, RealmDB, Room
- Testing frameworks : JUnit, Espresso and Mockito

Development Environment

- Ubuntu 16.04 LTS (preferred) / Windows 10
- Android Studio 3.0.1 with latest version of Android SDK

Version Control

- Strong concepts of Git, can also catch up with Mercurial and SVN if required.

Pre GSoC Involvements

Link to Pull Requests: here

I came across OpenFoodFacts via their ideas page listed on the GSoC website. I was amazed by the idea of a crowdsourcing database aimed at day to day things. I was amazed at the immense amount of possibilities this database can provide. I immediately came in contact with the mentors and started working on a few small issues.
After making a few small contributions to the projects, I thought of taking up a big issue at hand. Therefore, I took the task of optimizing the code. This PR acquainted me with almost all sections of the code. (This PR is not merged at the moment) This was a big PR and currently I am working on the same PR to improve it further.

Project Introduction

Open food facts is an open, collaborative and free database of food products from around the world. Most of the data in the Open Food Facts database come from crowdsourcing through mobile apps: users scan barcodes of products and upload photos and data for missing products. We need Android and iOS apps that bring a lot

of value to users so that we gain mass adoption, and that have powerful features to contribute photos and data as easily and quickly as possible.

Currently, there exist native android apps for Open Food Facts. However they lack some of the key features (product wise) and have a large scope of improvement (software wise).

Current Shortcomings

- Usually crowdsourcing and data based apps have a small amount of their data cached locally. This data need not include the entire database but a small amount of data in the form of key value pairs. So that all products can be kept in sync and the user still knows what kind of information can be found online in the database and if the user is in proximity with a product that is not available online, he /she can make contributions while offline. Further extensions to this idea can be made.
- There is no personalisation in the app. This can be included as a major feature in the app so that it benefits the customer as well.
- Crowdsourcing apps such as Google Opinion Rewards usually have a monthly questionnaire which contributes to the crowdsourcing . Openfoodfacts can include this functionality as well.
- Programmatically speaking, OpenFoodFacts is a large database constantly evolving. This makes the android app quite crucial. However for long term maintenance, the app must be designed to follow a certain architectural and design pattern. Currently, there is as such no strict architectural style followed for the project which will eventually make maintenance and further development difficult.
- There is a lack of testing suites due to which bugs remain uncaught here and there.
- There is a plethora of third party libraries used without utilising them properly. (Despite using dependency injections, findViewById calls are not entirely removed.)
- Incorporating technologies like Pastec in the workflow.

Required Changes :

For a short timeline of GSoC , I guess it would be not wise to refactor the design of the entire project and hence I will try to incorporate architectural patterns in the project step by step while focussing on the main features stated above.

- To solve the offline data problem, I propose to use the Room Persistence library. The library helps you create a cache of your app's data on a device

that's running your app. This cache, which serves as your app's single source of truth, allows users to view a consistent copy of key information within your app, regardless of whether users have an internet connection.

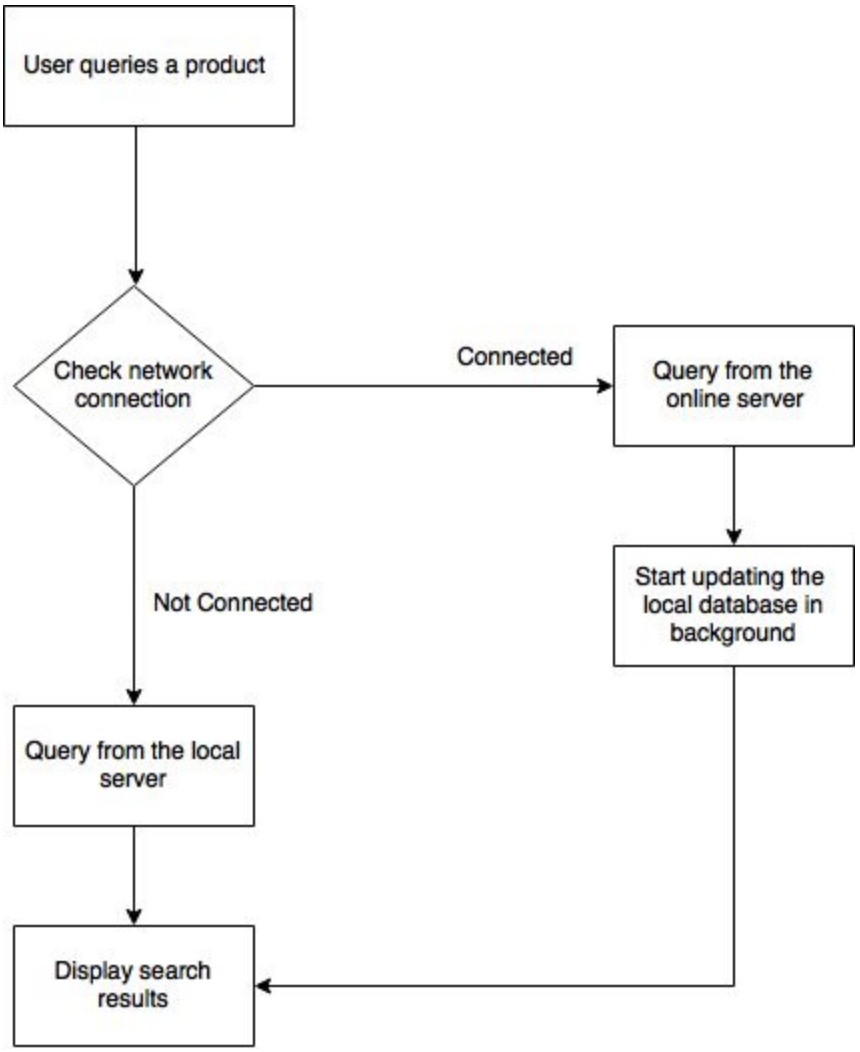
- The room library will also help in designing user personalisations as well. Including a small database for user preferences. Based on this small database we will be able to filter out the database locally and cache out things.
- Google has released new architectural components to enhance the software development aspect of Android projects. These can be slowly and gradually incorporated in the workflow. These include
    - Room
    - ViewModels
    - LiveData
    - LifeCycle aware components
- Drip editing can be included to enhance the quality of crowdsourced contributions.
- A robust testing suite should be gradually developed

Achieving project goals

The first and foremost thing to do is to add an offline database designed along the guidelines of MVVM architecture. This database will serve as a store for personalisation as well as caching minimum required data offline. Here is a detailed plan on how I aim to achieve the following tasks.

Offline mode

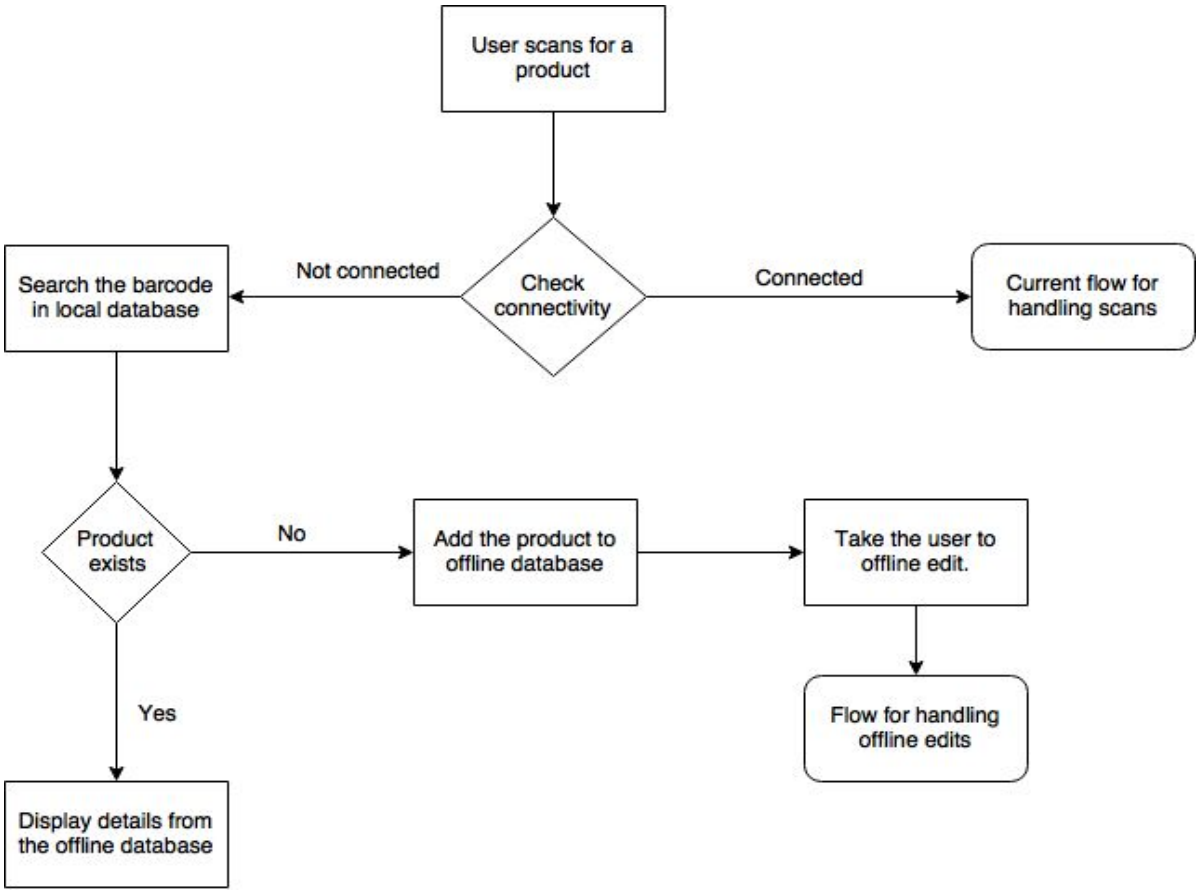
1. I shall use the Room persistence library (a wrapper around classic sqlite) to get started with. I plan to include the following minimum parameters to the offline database :
    - Product Id / Barcode
    - Product Name
    - Product quantity
2. This schema can however be altered in case of the history of visited products and users favourite products. Favorite products can be stored with all their details whereas the products in user history may or maynot have some additional details
3. This data will have to be updated periodically. I am thinking of adopting an **online first** approach. In this approach the flow will be as follows



4. Owing to the above approach the user's data will be synced as and when the user wishes to access the data without unnecessary downloads and refreshes. All the crucial data will be stored locally.

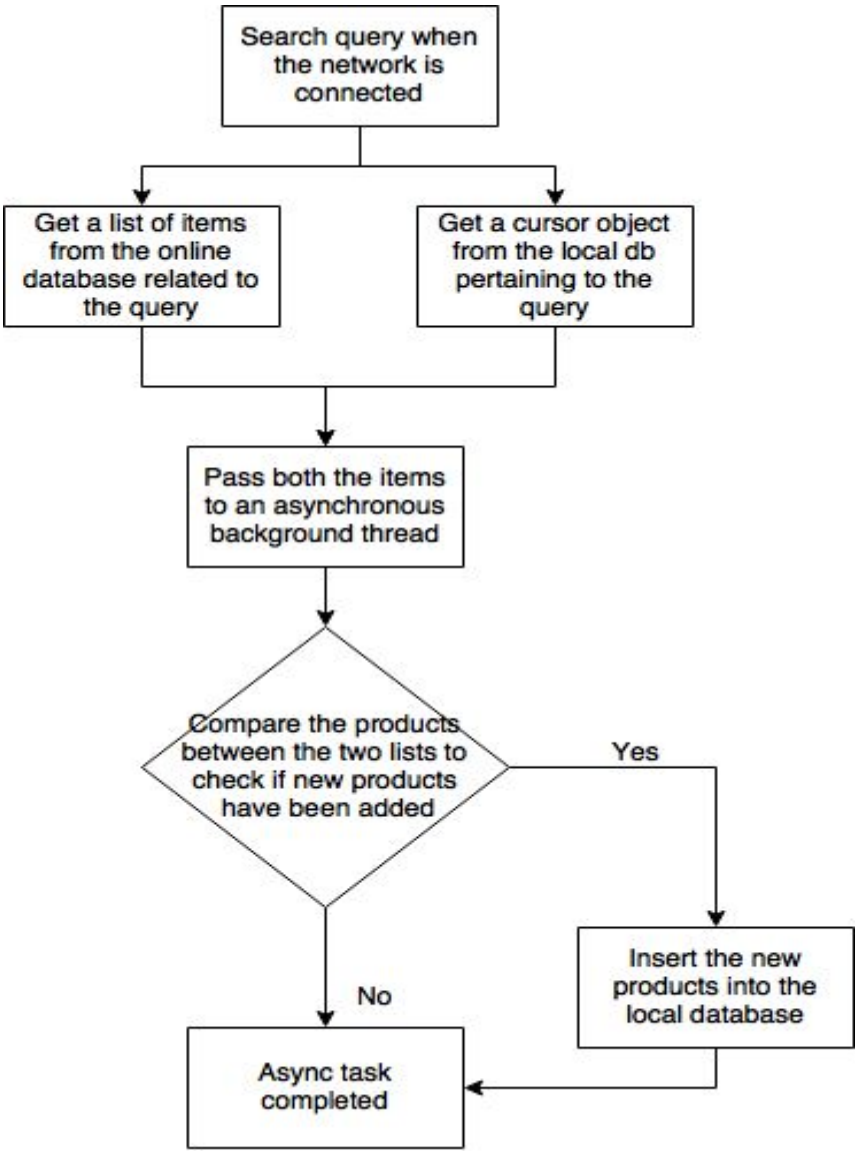
5. I am Planning to use LiveData and MutableLiveData components to design the workings of the above architecture. As a result there would be no need to handle UI events separately and the data would automatically become lifecycle aware.

6. Owing to the above database it would be easy for the user to track those products which are already present in the database and which products need to be uploaded. As a result there would be no hindrance even in the offline mode.



7. To make these changes possible a few of the existing implementations would have to be changed. The history fragment should be transformed as per the MVVP pattern to support LiveData and should be implemented in Room.

8. The updates on the local server will be carried out via the following flow :

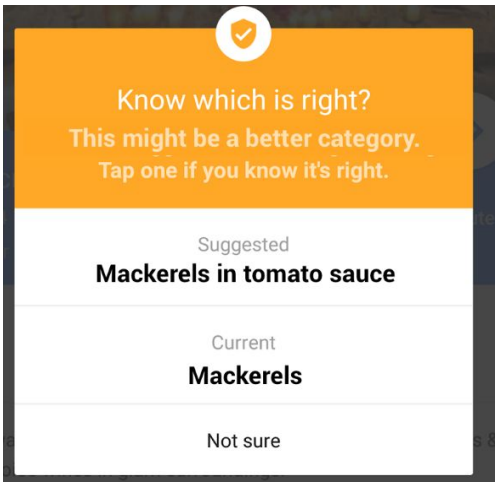


Drip editing

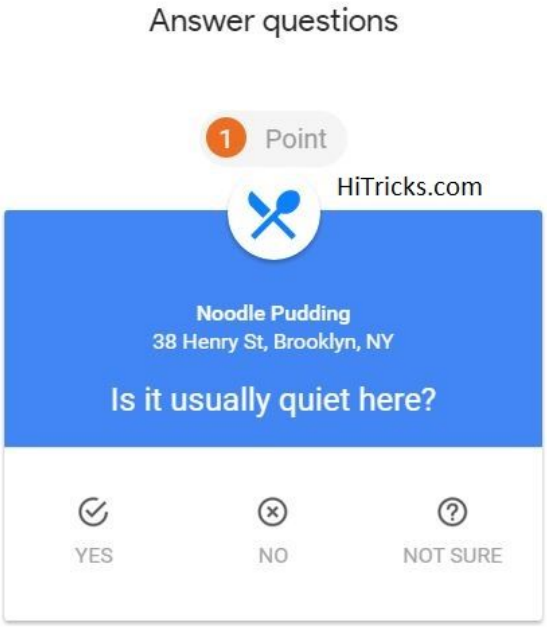
1. Drip editing means asking Open Food Facts users, little questions about the product they are looking at. They should take a split second to answer. Put together, they helps complete products quicker, update existing products and ensure quality.
2. Users can be asked several questions regarding whether the existing information is correct or if they wish to correct some of the existing information. They may be asked their feedback regarding a particular product at certain intervals of time.
3. The following are a few layout suggestions for drip editing :

a. Material Dialogs can be prompted when the user navigates to a particular product

.



b. The user can choose to answer a few questions on his own. We can provide a separate option to the user so that he can contribute to the database via answering few questions. The questions can be then presented to the user in the form of question cards as follows.



4. I propose two ways in which this can be achieved :
    a. Monthly surveys regarding a particular product (inspired from Google Opinion Rewards). This will allow OpenFoodFacts to take monthly surveys (if allowed by the user) by allowing him/her to answer a few question regarding some of the products. For this backend APIs will have to be developed. I have a fair experience at developing backend APIs and I will be happy to collaborate with the backend team.

b. Short on the spot questions regarding a particular product when the user is viewing them (inspired from Google Maps). These can be handled offline as well and can be implemented within the app itself.

Personalisation and recommendations

1. Although recommendations are usually carried out on the server side in most of the applications, however in the case of openfoodfacts this can be carried out on the device itself. I propose the following two ways to implement such features.
2. The first one is to allow the user to enter his/her personal details (age, sex, weight etc.) which can be stored using shared preferences . After storing the details of the user, the following steps can be carried out.
    - While showing the search results, the search adapter can scan the incoming list as per the user preferences.
    - While the user is looking at a particular product, alternative products with better features curated to the needs of the user can be displayed to the user (by scanning the local cache). For instance : if a user focuses on calories in a particular product, the local cache can sort the existing products of the same utility in order of increasing calories and hence the user will be provided with a plethora of better options.
3. The second way is to develop a small recommendation engine out of the users history of scanned and visited products. This recommendation engine can then later be used to display products relevant to them. This can be achieved by scanning through his history and storing crucial information about the kind of products he searches. For better results, computationally light machine learning algorithms can be applied. (Such as kNN) Such models will provide us products similar to the favorite products of the user (via clustering).
4. Besides including a recommendation feature, a favorites tab must also be implemented. While browsing through the products, users can flag their favorite products. This flag can be stored along with the offline product database in the form of a bool. A separate fragment will then show all the favorite products of a particular user.
5. Along with this, an option for applying filters in the list of products (after searching). As a result the user can tailor the search results as per his use. The filters can be based on ingredients, allergens or categories.

Developing a testing suite

1. For the long term support of a particular product, a well implemented testing suite is of utmost importance.
2. Though writing a test suite for the entire product is unrealistic, given the time frame of GSoC since it requires an appropriate architecture for the app. However, I plan to passively involve myself in writing tests. This will aid in developing a robust testing suite for the project.
3. I plan on using Espresso testing framework for UI testing and junit tests for testing other layers of the app.
4. These tests are quite concise to write and will simplify the process of reviewing PRs. An example of an espresso test that checks if the navigation drawer works perfectly

```
@Test
public void clickOnYourNavigationItem_ShowsYourScreen() {
    // Open Drawer to click on navigation.
    onView(withId(R.id.drawer_layout))
        .check(matches(isClosed(Gravity.LEFT))) // Left Drawer should be closed.
        .perform(DrawerActions.open()); // Open Drawer

    // Start the screen of your activity.
    onView(withId(R.id.nav_view))

.perform(NavigationViewActions.navigateTo(R.id.your_navigation_menu_item));

    // Check that you Activity was opened.
    String expectedNoStatisticsText = InstrumentationRegistry.getTargetContext()
        .getString(R.string.no_item_available);

onView(withId(R.id.no_statistics)).check(matches(withText(expectedNoStatisticsTe
xt)));
}
```

Proposed Timeline (WIP)

Current - 02 Apr	Working on the testing suite of the existing project. (Trying to merge my existing open PRs, one of them has lots of corrections.)
03 Apr - 09 Apr	Developing a schema for the offline database. (Passive : Minor reformatting to MVVM architecture)
10 Apr - 07 May	Incorporating LiveData to the flow (This time period will clash with my end terms, so I will not be available full

	time)
08 May - 14 May	Get the unfinished work of database including implementation of online first approach (My college break starts, can devote my entire time for GSoC)
15 May - 28 May	Coding phase starts. Start exploring for Drip editing options ((Passive: try out ways to handle offline recommendations)
29 May -11 Jun	Include product wise surveys for drip editing. **Evaluation No 1.**
11 Jun - 25 Jun	Completing basic search and filter offline recommendations. (Passive : Adding tests and documentation for drip editing)
25 Jun - 09 Jul	Improving the recommendation engine. Adding algorithms for recommendations from Search History. **Evaluation No 2**
10 Jul - 23 Jul	Completing the testing suite for the above mentioned task. (Passive : Adding Pastec support to the workflow)
24 Jul - 30 Jul	Integrating and adding Pastec. Completing final tests. Improving Camera and Final changes as suggested by mentors. Trying to implement the wishlist.
31 Jul - 06 Aug	

Wishlist: Can be implemented if both Primary and Secondary Goals are completed successfully and the programme is yet to end, also desired to be taken up post GSoC.

- Include Pastec and automation tools to the workflow.
- Complete the UI testing suite.

Other commitments

I will be having my semester examinations between 25 April - 5 May so I will be slightly inactive between 11 April to 5 May for exam preparations, which is why I have kept the timeline relatively light for this period.
During vacations (7 May - 18 July) I have no other commitments and I seldom get tired of coding - even for recreation I keep coding. I can easily devote more than 50 hours per week till 18 July. After that my institute reopens and I will have to spare some time for regular academics, hence I could then devote about 35-40 hours a week, but according to my timeline, the primary and secondary goals would be

almost complete by that time, so everything is manageable throughout the timeline. I believe that the allotted work per week is completely doable by me and neither overloaded nor slacked.

Why me for the Project?

My first contribution to open source was about an year ago. I have looked over to repositories of the products I use / come across, trying to give my part of contribution back to the organization whose product has been an asset to me. It gives a sense of immense pleasure to see my code changes going into production.

My contribution activity can be viewed in my Github profile, significant highlights are contributions in aima-java, Cognizance 2017, ,trianglify and Calling-Text.

Apart from my love for Android as a platform, another reason that I would love to work under your guidance this summer is that I have always wanted to make apps that can actually bring a change in the world. I tried making a community app called Closer for a hackathon. I realised that an app with such goals can only be made by dedicated efforts of a competent team of people who are motivated to make the world a better place. I was happy to find an organization like OpenFoodFacts in Gsoc's list which was dedicated towards common good. .
Some plus points about me:

- I have excellent Googling skills. Therefore, I can usually google my way out of problems.
- I have a thorough understanding of git. I can successfully do a rebase, cherry pick, revert, etc.
- I make small commits that can be reviewed one by one with clean commit messages and well structured PRs. I never commit a change that deviates from the original motive of the PR.
- I always review my own work thoroughly before asking for reviews. As a result my work is mostly error free.

I can easily devote more than 50 hours per week during my timeline. I make atomic commits with clean commit messages and well structured PRs. **I shall not leave anything undocumented.**

References

- https://developer.android.com
- https://en.wiki.openfoodfacts.org/Student_projects/GSOC/Proposals
- https://github.com/codepath/android_guides
- https://www.youtube.com/user/androiddevelopers
- https://summerofcode.withgoogle.com/