GSoC

2020

Google Summer of Code 2020

Wikimedia

Commons-App Android

Personal Information

Name:	Kshitij Bhardwaj
Github:	kbhardwaj123
Email:	kshitij256kencryption@gmail.com
Phone No:	+91-9634843951
Location:	India 🇮🇳
Time Zone:	India (UTC +5:30)
Slack:	Kshitij Bhardwaj
LinkedIn:	Kshitj Bhardwaj

University Information

University:	Indian Institute of Technology, Roorkee
Majors:	Electronics and Communication Engineering
Current:	IInd Year (batch 2022)
Degree:	Bachelor of Technology (4 Year Program)

Contact and Working hours

Reachable anytime through **email**, **slack** or **contact number**

Typical working hours :-

- UTC 0430 - 0730 hrs (IST 1000 - 1300 hrs)
- UTC 0930 - 1230 hrs (IST 1500 - 1800 hrs)
- UTC 1530 - 2030 hrs (IST 2100 - 0200 hrs)

Coding Skills

Programming Languages:

- Proficient **PHP**, **Python**, **Dart**, **Java**
- Intermediate Knowledge in **C++**, **Android**, **Javascript**
- Worked with **Django**, **React**, **Flutter**, **Angular**, **Yii 2.0**, **Figma**
- Databases - **MySQL**, **PostgreSQL**, **SQlite3**

Experience

Full Stack web and native app Developer, currently Hub-Coordinator at **Information Management Group** (IMG), IIT Roorkee. During my first year at college my workspace was confined to web development, during which I worked on a few applications, for instance IMGSched (application for scheduling meetings and sync to your google calendar), in my second year my workspace expanded to Native app development and some major projects listed in the following section.

My Projects:

- **CMS** :-
  CMS is abbreviation for Content Management System which is manages more than 6000 static pages of the official website of IIT Roorkee, the use of a Content Management system is to make a template for certain type of page (say news announcement page) so that anyone without coding knowledge (say some staff member or professor) can make changes to all those pages of that particular type by just changing the base template and publish changes.

  - ❖ Feature additions to the Institute **CMS** (10 commits)
  - ❖ Role: Backend developer
  - ❖ CMS handles the official website of IIT-R website
  - ❖ It manages more than 6000 static pages on the website
  - ❖ Tech Stack includes **Yii 2.0**, **PostgreSQL**, **RabbitMq**, **Apache2**

- **Placement Flutter** :
  At the Institute level it is imperative that the information flow is smooth and fast, the official placement app aims to provide the information regarding the placement and internship programs. Additionally the app facilitates direct submission of their resume directly to the companies, the app was migrated from native android to flutter due to fast development times
  - ❖ The official Placement app of IIT Roorkee
  - ❖ Role: Full stack developer (81 commits)
  - ❖ Currently under development (pre-alpha)
  - ❖ For Security reasons it implements **AES-256** encrypted tokens for auth and resume handling
  - ❖ Shows results for placement and internship seasons
  - ❖ Facilitates submission of Resumes to companies
  - ❖ Github Repo link: Placement-flutter

- **Flutter Login Template**:
  An Open Source boilerplate application which is completely packed with essentials for flutter development such as Stream Providers, Implicit Routing and flutter hooks.
  - ❖ Open Source Login Template
  - ❖ Role: Repository creator and maintainer (22 commits)
  - ❖ Includes Form validation
  - ❖ Complete implementation of Routing and Data Streams
  - ❖ Github Repo link: login-boilerplate

Project Overview

At its core the Underlying Goal of This Project is to incorporate the concept of **Gamification** into the Commons Android Application with the purpose of leveraging User's interest to increase their Wikimedia edit count. This project aims to realise such kind of Gamification by establishing a Leaderboards for the users wherein they will be ranked based on their edit count via the Commons-App and not the Web portal.

The app already implements the level system which can be combined with their ranks (indicative of their edits) and based on that we can award them **Rank Badges/Titles** whose distribution algorithm has been proposed in detail in this proposal later.

Contributions

- **Commons-App, Pull Requests:**
    - ❖ 3422 - For v2.13, handle zoom in media detail view **(merged)**
    - ❖ 3344 - TextUtils.isEmpty creates problems when unit testing with Mockito **(merged)**
    - ❖ 3378 - Ultimate achievement: Too many contributions **(merged)**
    - ❖ 3407 - Do not display pins at all when "Needs Photo" is selected **(merged)**
    - ❖ 3351 - Pending GCI task: Add java docs to methods which have it missing **(merged)**
    - ❖ 3326 - Make category search non case-sensitive **(merged)**
    - ❖ 3366 - Adds a Test for Method A categories search **(merged)**
    - ❖ 3332 - add javadocs to the file CategoryItem **(merged)**
    - ❖ 3512 - Fix App Crashes on opening Contributions from MainActivity **(merged)**
    - ❖ 3505 - Media Detail design Overhaul **(open)**
    - ❖ 3481 - Implement Progress Bar for Zoom Activity **(open)**
    - ❖ 3513 - Image height remains zero for some time on bad network **(open)**

- **Commons-App, Issues :**
    - ❖ 3343 - TextUtils.isEmpty creates problems when unit testing with Mockito **(closed)**
    - ❖ 3436 - Switch to Material Card design for the Media Detail Fragment **(open)**
    - ❖ 3506 - Image View Height remains zero for some time on a bad network **(open)**
    - ❖ 3507 - App Crashes in MediaDetailsFragment with ClassCastException (**closed** by PR 3512)
    - ❖ 3414 - Handle zoom in media details view (**closed** by PR 3422)
    - ❖ 3355 - Do not display pins at all when "Needs Photo" is selected? (**closed** by PR 3407)
    - ❖ 3295 - Ultimate achievement: Too many contributions (making our script time out) (**closed** by PR 3378)
    - ❖ 3278 - Add java docs to methods which have it missing (**closed** by PR 3351)

For Contributions After 29th March visit here

Benefits to Commons App

- **Leaderboards:**
  - ❖ Would implement Gamification in a much better way as compared to the currently implemented User level system.
  - ❖ Would encourage users to make more edits, compete for higher ranks in the leaderboards thereby improving the Commons image repository.
  - ❖ The additions of limited time span leaderboards will prevent users with less overall edits from getting discouraged as they can still top the weekly or monthly charts.

- **Rank Badges/Titles:**
  - ❖ Will serve as an incentive for long term contributions and encourage Users to maintain their pace of contributions to avoid losing their Rank Title/badge
  - ❖ This badge could be differentiated from the badge which appears on the Achievements page by including the leaderboard ranks along with the user level which would imply that the user can lose their rank badge/title
  - ❖ Introducing nearly 10 or 20 ranks will make our Gamification more discrete, to be more precise if we have thousands of users then there might be users who could get completely discouraged if they obtain ranks in order of thousands on the other hand those titles/badges which will be based on their rank will make it more comprehensible and motivating

Technical Discussion

Features that can be Implemented:

- Implement database schema which is entirely separate from the existing edit counter as it includes counts made from the web portal. The APIs which interact with this particular schema will have their End-points served only to the Commons Android app.

- Leaderboards calculated from the above database schema will serve their results via a single End-point which would accept the following three parameters:
  - ❖ Uploads/Nearby/Used (as single integer viz. 0,1,2)

    - Weekly/Monthly/All (again as integer parameter)
    - UserName

- Entire Calculations will be accomplished on the server side and the Android app's leaderboard section will have the sole purpose of displaying the results.

- The Leaderboards will be cached and stored on the server in a database and will be refreshed at regular intervals.

- The leaderboard database includes only those users who have explored the Achievements activity at least once.

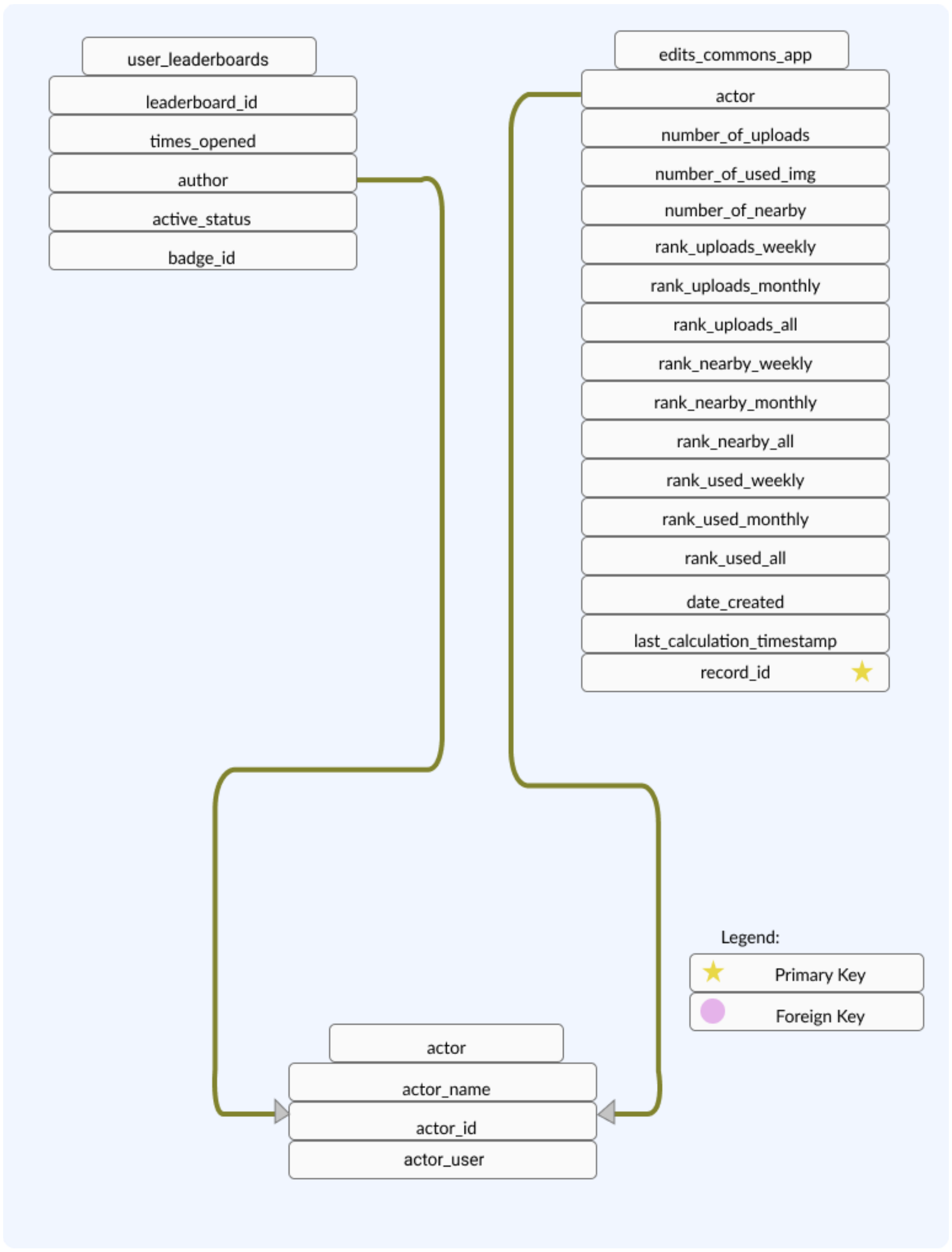
- Server will perform complete re-calculation of leaderboard ranks at regular interval of 1 hour, additionally for users who are really intrigued in the leaderboards and check it more than 5 times a day (on average which will be calculated on a per month basis) will get updated leaderboard after every 5 minutes interval.

- Implement Rank Badges and/or titles to motivate users to maintain their pace of contributions so as to get a higher badge , apart from that it will prevent users with ranks in orders of thousands to get completely unmotivated as badge distribution will be much more discrete in order of 10 or 20 badges/titles in total. The inspiration for this feature has been from Competitive games like CS:GO, this feature is highly effective in leveraging long term user interest.

Realisation of mentioned Features:

- The database schema will provide an end-point to the app which consequently ensures that only the android app users enter the leaderboard database. The Schema can be implemented as mentioned in the model structure diagram sketched below:

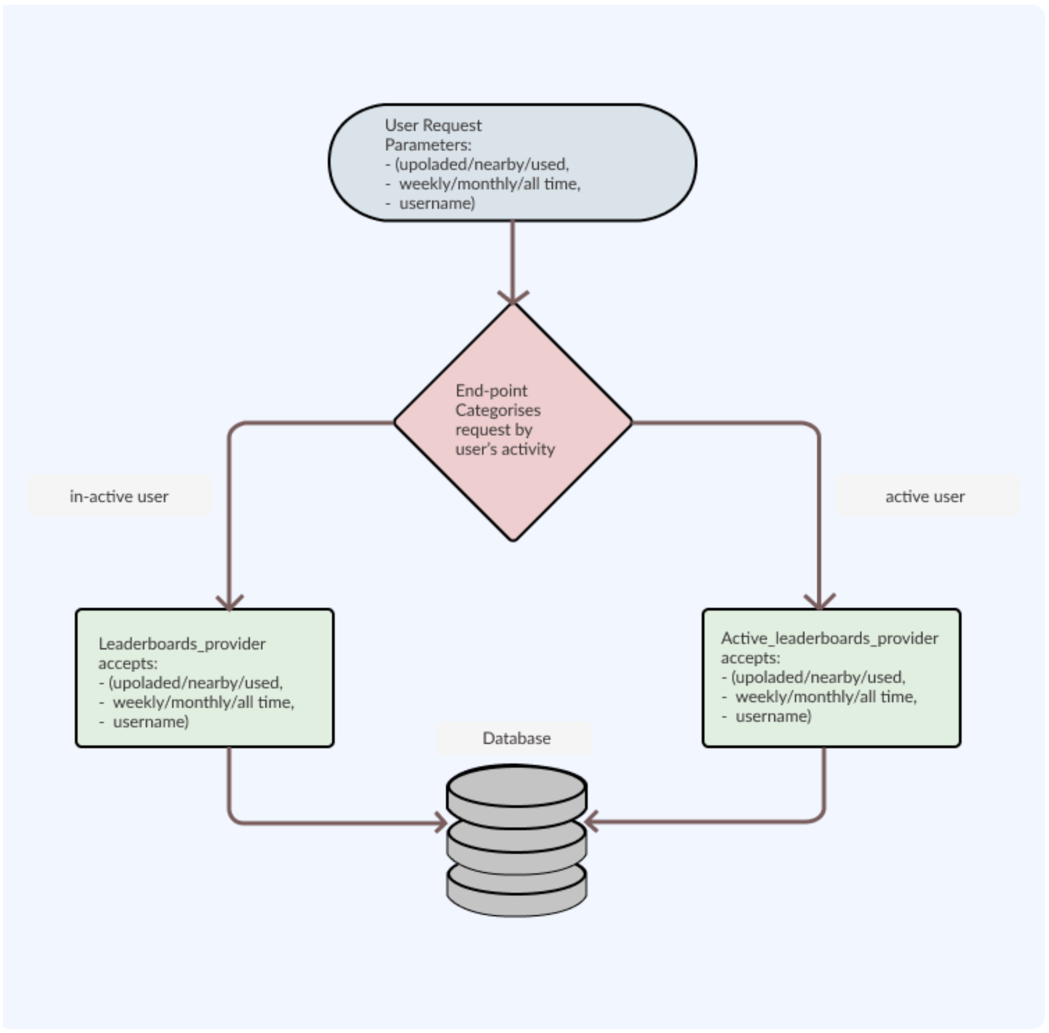
Model Structure Diagram:



The Actor table in the Database refers to the user table in the database named `wikidatawiki` as mentioned in wikidataedits.py

- The results will be served using a single API End-point as discussed previously, the server side request flow will implement the following pattern:

Request Flow Diagram:



- How can this division of requests improve performance ?

  The users who are check leaderboards quite often will be marked as **Active users**, as a matter of fact only a fraction of the total user base will be highly active so even if they are provided with the feature of more frequently updated Leaderboards the server would not have any extra load or request traffic

  The **Non-Active users** would be provided with the leaderboards which are refreshed once in an hour.

- How can **Rank Calculation** architecture be implemented ?

  The API hosting server can run certain services for periodic rank re-calculation, these services can be implemented using the **Advanced Python Scheduler** (APScheduler) or we can also use **Cron Jobs(** description **)**.

  ❖ Uploads Leaderboards:

The commons applications provides public logs of the users which when displayed with conditions type=upload and with tag filter=`android app edit` we only get the uploads made by the android app as can be seen here .

The API would implement this as follows:

```
Use commonswiki_p;
Select count(*) as edits
From logging
Join change_tag on log_id = ct_log_id and
ct_tag_id=22
# Tag id 22 is for Android App edits
Where log_type='upload' and log_actor = ACTOR_ID;
```

❖ Nearby Leaderboards:

  These are counted when images are uploaded while in Nearby section so they are automatically from the mobile app.

❖ Images Used in articles Leaderboards:

  These are calculated from the globalimageslinked table under the GlobalUsage extension, to sum it up:

```
Use commonswiki_p;
Select count(distinct gil_to) from
globalimageslinked
Where gil_to in (Select log_title from logging
Where log_type='upload' log_actor=ACTOR_ID);
```

- How can **Badge Distribution** be implemented:

  For badge distribution we combine user's rank (monthly/weekly rank, could be based on uploads) with their achievements so as to incorporate **long term** (the level) and **short term** (weekly/monthly rank) so we define a quantity x whose formula has been described below and we find its standard deviation and the badges are distributed as per the Badge distribution diagram .

Badge calculation Algorithm :

Ri - Rank of i-th user

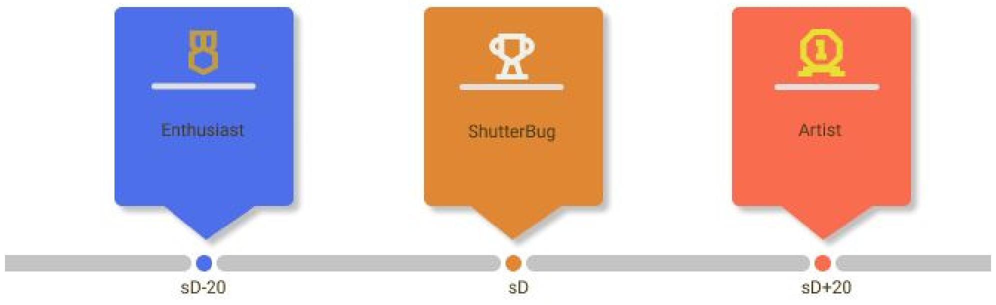
Li - Level of i-th user

N - Total users

$$X_i = \sqrt{R_i * L_i} \qquad \text{mean of X, } u = \frac{\sum \sqrt{R_i * L_i}}{N}$$

$$\text{Standard deviation, sd} = \sqrt{\frac{1}{N}\left[(X_1-u)^2 + (X_2-u)^2 + \cdots (X_N-u)^2\right]}$$

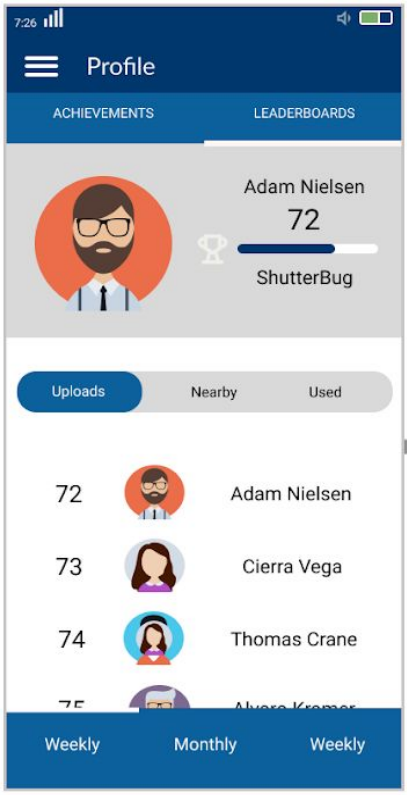
Badge distribution diagram :



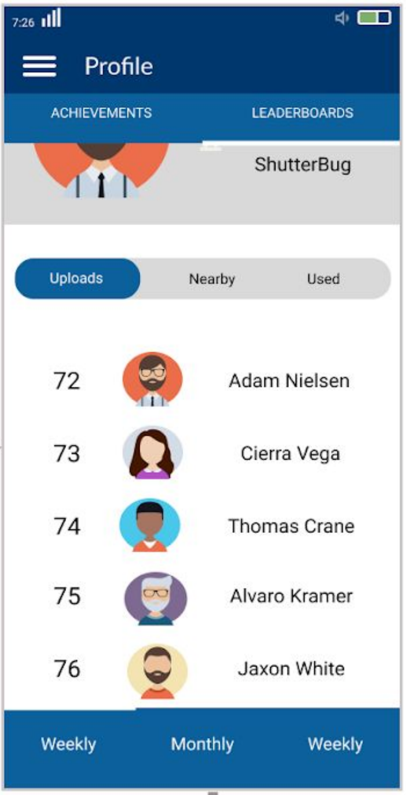
The badges can be distributed with a gap of 20 (tentative) in the Standard deviation.

- All Leaderboards will implement lazy loading, technically this can be realised by implement pagination in RecyclerView using RxJava operators

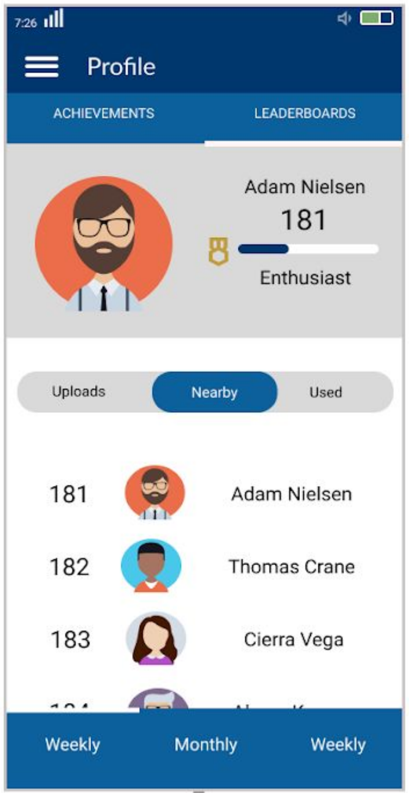
UI - UX Research



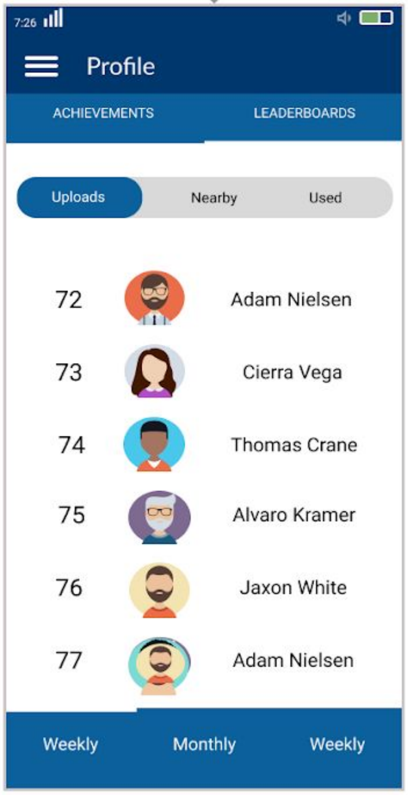
72	Adam Nielsen
73	Cierra Vega
74	Thomas Crane



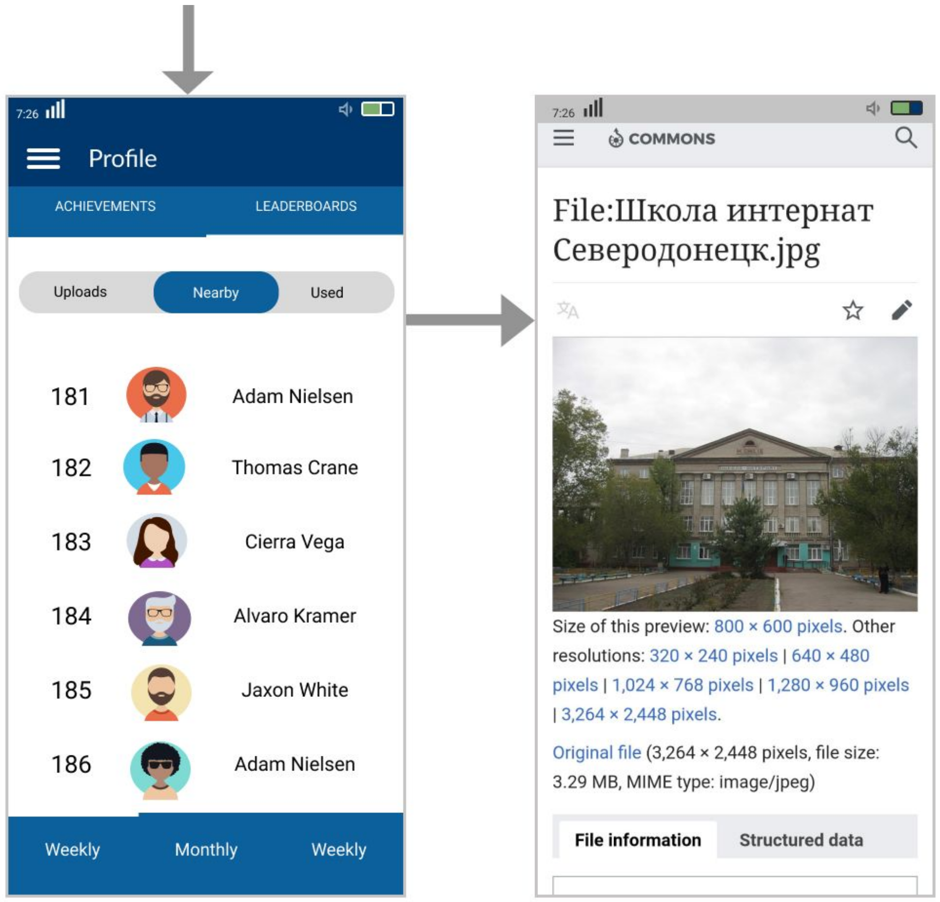
72	Adam Nielsen
73	Cierra Vega
74	Thomas Crane
75	Alvaro Kramer
76	Jaxon White



181	Adam Nielsen
182	Thomas Crane
183	Cierra Vega



72	Adam Nielsen
73	Cierra Vega
74	Thomas Crane
75	Alvaro Kramer
76	Jaxon White
77	Adam Nielsen



181	Adam Nielsen
182	Thomas Crane
183	Cierra Vega
184	Alvaro Kramer
185	Jaxon White
186	Adam Nielsen

File:Школа интернат Северодонецк.jpg

Size of this preview: 800 × 600 pixels. Other resolutions: 320 × 240 pixels | 640 × 480 pixels | 1,024 × 768 pixels | 1,280 × 960 pixels | 3,264 × 2,448 pixels.

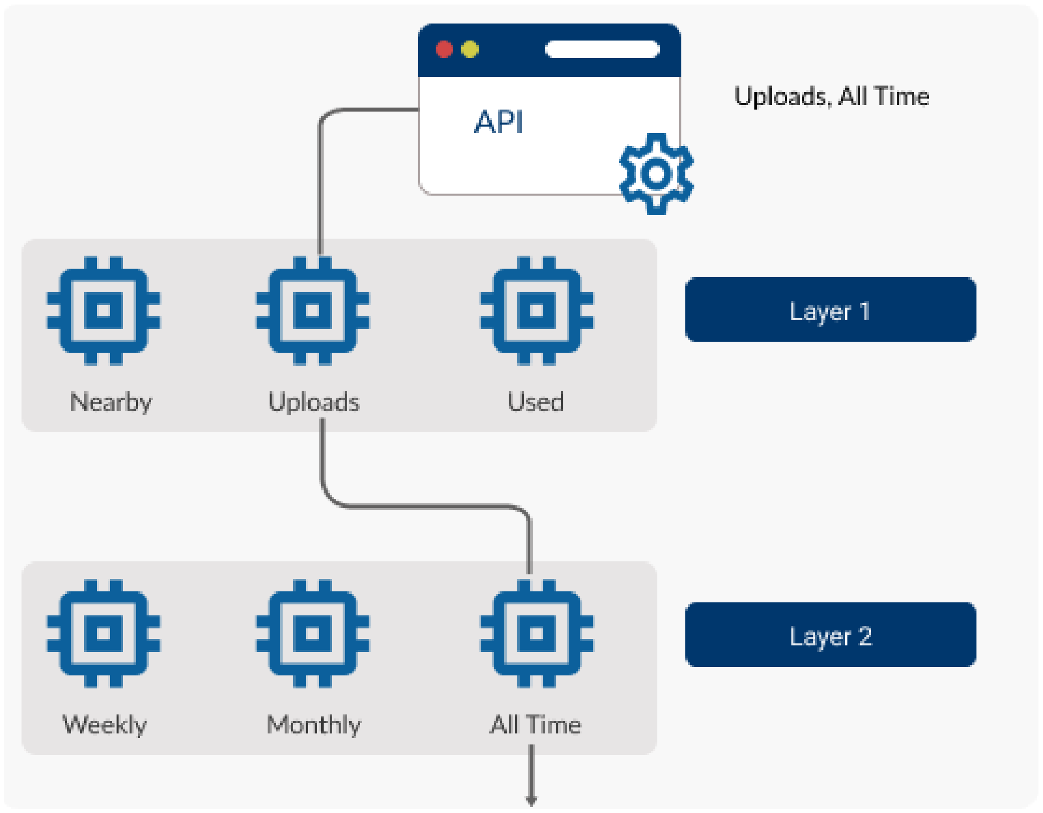
Original file (3,264 × 2,448 pixels, file size: 3.29 MB, MIME type: image/jpeg)

The Design Screens shown above demonstrate the flow of the app in the leaderboards section, the last screen shows the user screen when the user has clicked on one of the leaderboard position holders, this first opens the commons gallery page of that particular user from where users can open any of the uploaded images (this is exactly what has been shown in the last screen, one of the gallery pictures)

CRUD Operations

An Overview of how the crud operations will work has been drawn below

For Evaluating uploads of Single user:



The Leaderboards will be processed in two layers as shown above:

- Layer 1:

  The uploads will be derived from joined logging and change_tag tables now we will also have the timestamp in the combined result records, in all the cases (Nearby/Uploads/Used). These results are calculated on the basis of the logic discussed under the heading **Rank Calculation.**

- Layer 2:

  In all the three cases (Nearby/Uploads/Used) the final record list will have **timestamp** (log_timestamp) now this timestamp will be used to filter the leaderboard results as follows:

  - **Weekly:** If Current TimeStamp - Log TimeStamp < WeekDuration

    Where WeekDuration is the difference between two timestamps with one week gap.

        - **Monthly:** If Current TimeStamp - Log TimeStamp < MonthDuration

          Where MonthDuration is the difference between two timestamps with one month gap.
        - **All Time:** No additional calculation required.

- Layer 3: Sort the results as per the upload numbers and then store the rank into the particular case (for instance uploads/all) into the field **rank_uploads_all** in the record of the corresponding user, inside the table **edits_commons_app.**

The first two Layers will be repeated for every user and for all the (3*3) 9 cases we will store the corresponding fields of the **edits_commons_app table.**

After the completion of all the three layers we will have a complete leaderboard record of a particular user, this will be repeated for every user.

The **Rank Badge calculation** kicks in at this point, from the algorithm it might seem like the process would be highly taxing for the server but actually it's quite straight-forward. The 9 final ranks per record in the edits_commons_app table leveraged into the standard deviation formula mentioned in the **Badge Calculation** section.

Project Timeline

Community Bonding	
4th May - 1st June	• Get to know the Community more, and bond with mentors, admins and developers. • Get feedback if something in the project needs amendment. • Finalize the Database Schema and APIs. • Read more about RxJava, APScheduler and other relevant android technologies and concepts. • Get familiar with Commons database schema and ideate on how it will implement the new tables proposed in the project • Finalise a barebones implementation of the ui using viewpage and recyclerView in the Achievements activity • Deliverables: ○ Report on Community Bonding period
Phase - 1	
Week 1 1st June - 7th June	• Start UI implementation of primary filters (nearby/used/uploads)with mock data • Implement basic view pager • Get approval on database schema changes and gather necessary knowledge on how to implement it.
Week 2 8th June - 14th June	• Implement Capsule navigation bar (primary filters)in leaderboards view page.

	• Get approval on MySql and MariaDb script code changes (script 1 and script 2) • Create Pull Request for script changes, iterate upon received reviews
Week 3 **15th June - 21st June**	• Commence coding the python API algorithms for Layer 1 filters(nearby/used/uploads)and storing rank in the database. • Discuss API code structures with mentors. • Create Pull Request on commonsmisc repo with the Layer 1 implementation, iterate upon received reviews.
Week 4 **22nd June - 28th June**	• Start Code implementation of APScheduler services (responsible for periodic updation of leaderboards). • Discuss Services code related issues with the mentors. • Create Pull Request for services on commonsmisc and iterate upon received reviews.
Phase - 2	
Week 5 **29th June - 5th July**	• Debug the database and Layer 1 API changes, ensure that they are working in synchronization. • Ensure only those who visit achievements enter the database. • Commence recycler view implementation of leaderboards • Get Approval on the final Leaderboards design along with code logic(upto this juncture).
Week 6 **6th July - 12th July**	• Implement RxJava Observable pattern for the received user list.

	• Discuss the final Observable pattern code implementation of leaderboards. • Create a Pull Request for the work done on a different(from master) branch, amend as per the inputs received.
Week 7 **13th July - 19th July**	• Debug the database-API(Layer 1)-Userlist chain. • Start implementation of Layer 2 for API to implement time filters.
Week 8 **20th July - 26th July**	• Complete Layer 2 implementation along with the Rank Badge/Title calculation. • Synchronise services with the updated APIs(Layer 1 + 2). • Create Pull Request(not master), iterate on inputs.
Phase - 3	
Week 9 **27th July - 2nd August**	• Commence functional implementation of the time filters in the App. • Debug combined implementation of all filter combinations in the application
Week 10 **3rd August-9thAugust**	• Finalise code structure the commence implementation of pagination to facilitate lazy loading. • Iterate upon the reviews received on the code changes so far and get it merged with master.
Week 11 **10th August - 16th August**	• Finalise how avatars would be stored(possibly local storage or cache). • Implement avatar display along with Rank Badge display in the UI.

	• Get reviews on the final ui code, amend as per the suggestions before merging to master
Week 12 **17th August - 23rd August**	• Fix the bugs. • Document as necessary and write unit tests. • Implement the Avatar change feature.(if time permits)
Final Evaluation 31st August - 7th September	• Complete Implement the Avatar change feature.

Activity Deliverables

- Write blog posts on medium every alternate week.
- Write weekly report of accomplished tasks on the mailing list (or something more relevant)
- Prepare and submit a presentation for this project.

Motivation

Google Summer of Code is an excellent platform to get acquainted with the open source community and their skillful mentors. It gives one a professional work experience in their college years where they collaboratively build a product for the welfare of the society. In this process, both the individual and the community progress and flourish.

Contributing to **Commons** has been an enriching experience, the mentors and the community have been really friendly and helpful in giving me constructive criticism making me learn new stuff and improving my coding skills.

Post GSoC

A couple of months have passed since I have been contributing to commons-app and must say that I have happened to develop a keen fondness towards the open source scenario. After the final evaluation I still wish to be a part of the commons app and continue to contribute to its cause of visually document this fine old world. In the long term if the community ever happens to wonder about creating the app on a non-native platform (flutter or react-native) I would be more than excited to hop on board.

Availability

During the GSoC period I do not have any other summer internships or any other sort of occupancy so I would be available throughout the gsoc period at the times mentioned in the working hours section.

Resources

- GSoc task page
- Github issue for GSoC
- WikiMedia Database Schema
- Quarry-beta
- APScheduler

Design Links

- **Model Structure Diagram:**
  Link
- **Request Flow Diagram:**
  Link
- **Badge Distribution Diagram:**
  Link
- **UI - UX:**
  Link
- **Layer Diagram:**
  Link