AnkiDroid's GSoC 2022 Ideas List

[>> Direct Link to Ideas List <<]

AnkiDroid is a free Android flashcard app which makes remembering things easy[1] via use of spaced repetition. Because it's a lot more efficient than traditional study methods, you can either greatly decrease your time spent studying, or greatly increase the amount you learn. Anyone who needs to remember things in their daily life can benefit from AnkiDroid. Since it is content-agnostic and supports images, audio, videos and scientific markup (via LaTeX), the possibilities are endless.
For example:

- Learning a language
- Studying for medical and law exams
- Memorizing people's names and faces
- Brushing up on geography
- Mastering long poems
- Even practicing guitar chords!

We are proud to have 4.7 stars on the Play Store, and 2.3 million active installs (as of February 2022). It has an active community of developers, with 39 contributors to code in 2021, two new maintainers, and even new localizations. Many of the contributors discovered Android programming with AnkiDroid and we are always happy to help people contribute, independently of Google Summer of Code.

Community

You can reach us on the Anki Discord (#ankidroid-gsoc & #ankidroid-dev) or our mailing list.

We primarily use Discord for real time communication, and GitHub issues for context on issues which would be useful for future contributors.
Our GSoC Mentors/Org Admins are as follows:

• Akshay Jadhav	https://github.com/Akshay0701	akshay5343#5089
• Arthur Milchior	https://github.com/Arthur-Milchior	Arthur#0801
• David Allison	https://github.com/david-allison	DavidA#0813
• Shridhar Goel	https://github.com/ShridharGoel	SG#6013
• Mani	https://github.com/krmanik	kr_mani#9295

Mike Hardy (https://github.com/mikehardy) is a maintainer and Org Admin

Our GitHub repository is: https://github.com/ankidroid/Anki-Android/ and our Wiki contains a lot of useful information, including our Getting Started page.

[1] The first paragraph comes from anki's website.

Ideas List

In this document, you'll find a list of projects that AnkiDroid would offer to mentor for if we are selected for GSoC in 2022.

We will not accept a project which is not on the ideas list unless you discuss it with a mentor. Please see [Your Project Here] for details

Table of Contents

Visual Editor (175 hours)

AnkiDroid supports entering arbitrary HTML to be displayed on cards, but this is currently entered by users as text. This has been a large frustration for users, as many are non-technical and do not understand HTML. Even newlines (
) have been the source of pain for users. Buttons have been added to allow the user to allow basic formatting such as italic, bold, etc... But this is a stopgap, as users do not see the visual impact of their changes until they either “preview” or “save and review”.

The ideal would be to have a WYSIWYG editor to allow a user to see the formatting of “fields”, both to allow for easier editing, and to allow for better feedback on card creation. This is especially important for users who do not have access to the Desktop version of the software.

We foresee this as initially adding in a separate “field editor screen”, which would allow a user to edit specific “fields” within a note. WYSIWYG is a complex feature, and rendering bugs are likely to exist, even with extensive testing. A separate screen will allow power users to discover any bugs that this introduces without impacting regular users.

Once bugs are handled and the feature is stable, we can replace our “note editor” screen with these “field editor” components, and allow a full WYSIWYG experience.

This project has already been started but is not yet fully functional. The GSoC candidate would bring this project to completion.

Note that there is a risk that in a few years, we totally change the note editor in order to use anki desktop’s one (in Svelte, using protobuf). This project is still totally worth a contributor’s time, because it will ensure that our millions of user will have access to a visual editor in 2022 and not in 202X.

Expected Outcomes

- Field Editor Screen in AnkiDroid, allowing full WYSIWYG editing of fields
- (extension) Augmenting our “Note Editor” screen with this “Field Editor”

Language:

Kotlin/Java, Android XML, HTML

Difficulty:

Medium

Mentor(s):

Akshay & David

CSV Import (175 hours)

Currently, AnkiDroid allows only users to import decks in the format .apkg, a format used only by Anki. While it is great to share a deck that already exists, it is not always ideal to create a new deck.

As an example, let's say that you want a deck to help you learn Chinese and your mother tongue is Persian. A Chinese-Persian deck may not already exist and you may wish to create one.

One of the easiest ways to do this is via a CSV file, a file displaying tabular data. A user may use the `=GOOGLETRANSLATE()` function in Google Sheets to quickly generate a CSV file for import.

Anki Desktop (our upstream project) has support for importing a CSV, and we want the same.

The problem with this feature is that it is probably our most complex user-facing functionality. We have enlisted the help of a UX designer, who has provided fantastic designs, but we have not had the manpower to implement these.

The task would be to implement these UX designs, via creation of bespoke Android widgets and Views to make the CSV import process as simple and pain-free for our users as possible.

Expected Outcomes

- A user may import CSV and TSV files into AnkiDroid
- Our UX/UI designs are implemented, and tested to reduce user friction when importing CSV

This task requires:

Experiences with Kotlin, Android XML, CSV, TSV

Difficulty

Medium

Mentor(s):

David & Mani

Notification Subsystem (350 Hours)

AnkiDroid had notification support many years ago, but significant changes to both the Widget API and the Notification API has meant that these are no longer viable for users. We should start from scratch on this, and design & develop a notification system where users can select certain conditions (remaining cards/time of day) where users can receive notifications.

As this is a complex feature, we will require a UI to make this as easy to understand as possible, to encourage more users to use it, and reduce support load

AnkiDroid cares deeply about privacy and attention. We may want a prompt to inform users that notifications can be enabled, but we do not want these to be enabled by default

Expected outcome:

- A notification subsystem is designed
- A GUI for the the notification subsystem is implemented, with focus on user experience and ease of use
- A user can create and remove notification triggers depending on user-provided parameters
- Notifications which are translatable by our translators will appear, and will provide a pleasant experience to our users who would like to study daily

Skills required:

Kotlin Development
Understanding of Android Notification APIs
Android UI/UX

Difficulty:

Medium

Language:

Kotlin

Mentor(s):

Arthur & Akshay & David

Migrate AsyncTask to Coroutines (175 Hours)

AnkiDroid currently uses AsyncTask for asynchronous tasks. Since AsyncTask has been deprecated and we have better alternatives available, we have decided to migrate our project to perform tasks asynchronously using Kotlin coroutines. This includes tasks like local database access and network calls.

AnkiDroid Wiki on asynchronous computation

Kotlin coroutines on Android: Kotlin coroutines on Android

Expected outcome:

Implementing the usage of coroutines in a structured way in the project. Proper documentation on how the migration has been done and guidelines for other developers, if required.

Skills required:

Knowledge of Android development using Kotlin

Difficulty:

Medium

Language:

Mainly Kotlin, maybe still some Java

Mentor(s):

Shridhar

Profiling AnkiDroid Operations (175/350 hours)

AnkiDroid has a number of users who use it for multiple hours every day. We aim to keep the critical “loop” of AnkiDroid as fast as possible, but we do not assert that this is fast in a scientific manner.

One notable example is that we added a regression which added milliseconds of latency if users tested and recalled their knowledge before 1s had elapsed. This was picked up almost immediately by our users and made it past all of our testing.

We would like to add profiling support, both via analytics and tests to the codebase to ensure that we do not add regressions, and reduce the worry that we add performance regressions when making code changes.

We currently have visibility over “App not responding” (ANR) events in AnkiDroid, but we have not had the developer resources to begin tackling it. As an extension to the project, a framework to protect against these may be considered.

Expected outcome:

- We can see profiling data for critical tasks that AnkiDroid performs
- We have automated test coverage, and can
- (optional) We start formally tracking the number of ANR events in AnkiDroid
- (optional) We reduce the number of “ANRs” reported, ideally to zero

Skills required:

Kotlin Development
Unit Testing
Analytics

Difficulty:

Medium

Language:

Kotlin

Mentor(s):

Arthur

Improve Card Browser (175/350 hours)

AnkiDroid's Card Browser allows users to browse, search and edit cards quickly.

Currently, the Card Browser offers limited functionality to users in comparison to the desktop version and it can also use an improvement on its User Experience to make it more accessible to users. This project consists in improving its UI and porting features from Anki Desktops, such as:

- Sidebar for filtering
- Browsing for both Cards and Notes
- Exporting selected notes

Expected outcome:

- Addition of a sidebar, to allow for much better filtering options
- Functionality to switch between "cards" and "notes" in the browser, thus unifying both types
- Fixing view sizes of each card
- Being able to export notes selected in the browser

Skills required:

Kotlin Development
Unit Testing
Android XML/UI/UX

Difficulty:

Medium

Language:

Kotlin

Mentor(s):

TBD

[Your Project Here] (175/350 hours)

We welcome you to propose your own projects, but **please get in touch with our Mentors before writing your project proposal.**

When you contact our mentors, you should provide a brief project description for consideration (similar to the high-level descriptions in this document).

Your project description should contain:

- Why the project matters to AnkiDroid
- What you expect to do by the end of the coding period
- Expected difficulty level
- Expected time (175, 350 hours, or maybe less than that if it is combined with another project)
- Technologies it will use. We use mostly Kotlin, but some JS/CSS/Rust/Java may be involved for some projects

We will not accept a project proposal which is not on the ideas list unless you have discussed it and found a mentor.