TRAKR

Project overview

Trakr is a minimalistic time tracking app to record what you do throughout the day, so you can look back on time spent, which could be useful for planning or reflection at the end of the day. I was motivated to make this app as it feels like something I need and will use. Previously, I have tried other time tracking services like Toggl, however I felt by making my own app I have more control over the features and can customize it to fit my needs.

Features of the app include:

* Authentication and account management, for example sign in & out, delete account, updating profile details and changing password
* Adding time entries, editing their details & deleting them
* Time entries can have a title, and a color. How the color is used to describe the activity is up to the user’s interpretation. For example, it can represent the user’s mood while doing it, or the type of activity, such as homework.
* Replay button to conveniently repeat common tasks (at the same time it finishes the active time entry)
* A user’s colors can also be added or removed.
* Analytics to help the user review and better understand how he spent his time.
* A list of the entire history of user’s recorded activities. To allow easier navigation, users can select a desired date from a calendar view to see what they did that day
* A light and dark mode

Code explanation

The code is sorted into 4 packages:

* models – for common models to store data on users and a time entry
* ui – holds all fragment classes and their corresponding adapters or other related classes
* utils – miscellaneous tools including formatting and data validation
* viewmodels – details of the firebase API are abstracted away using UserViewModel (for authentication and persisting the current user across the app), DbViewModel (for read and write operations with cloud firestore) and StorageViewModel (for interfacing with firebase storage)

The firebase schema is as follows:

* users collection
  + username
  + photoURL
  + activeTimeEntry
  + colors
  + createdAt
  + timeEntries collection
    - id
    - title
    - startTime
    - duration
    - color

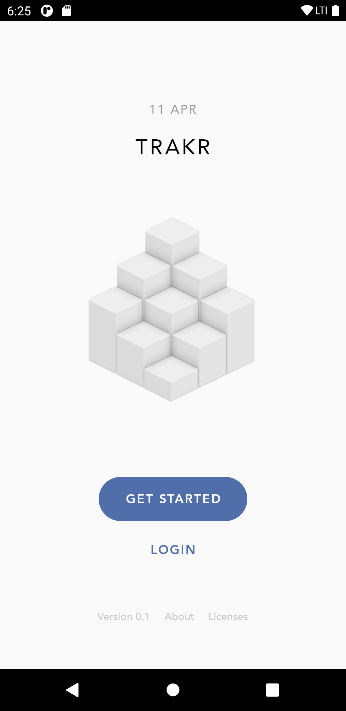
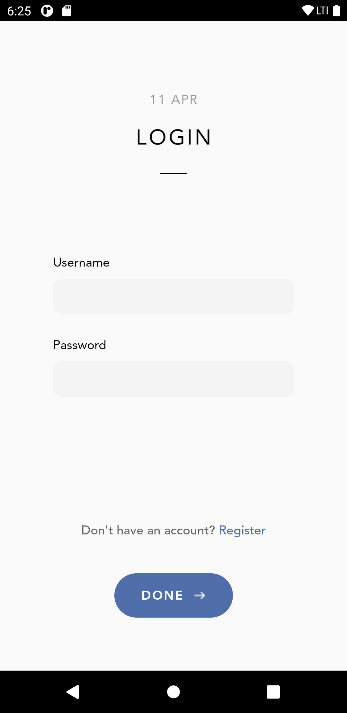
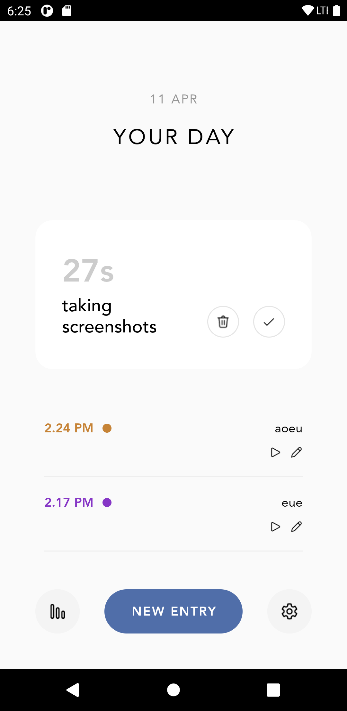
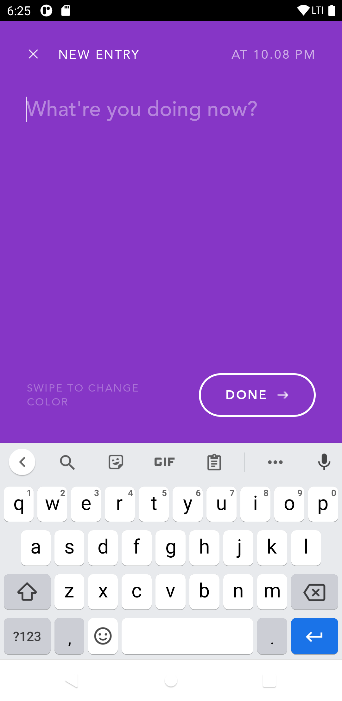
For each model that is fetched from firebase, I wrote serializers to convert them to and from firebase objects, namely fromDoc and toHashMap, thus allowing for easier construction of the objects and writing to the database.

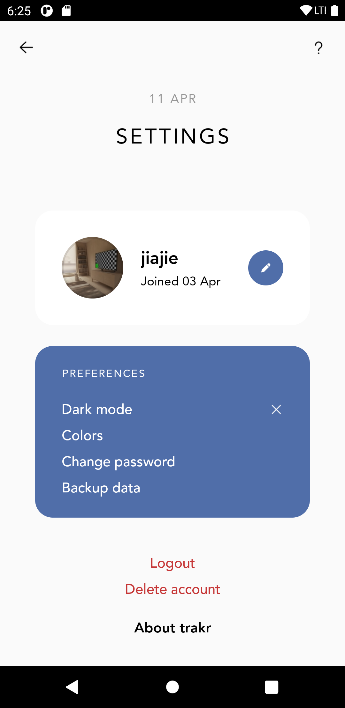
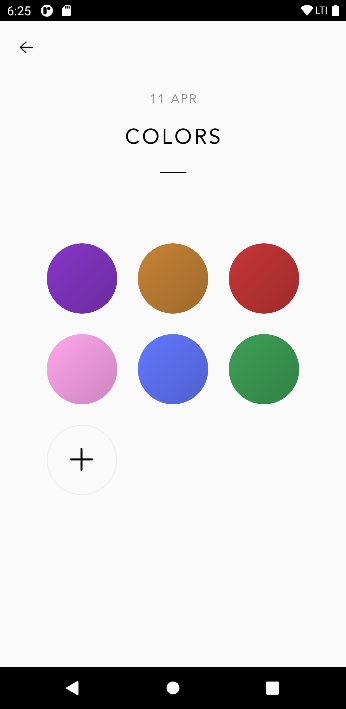
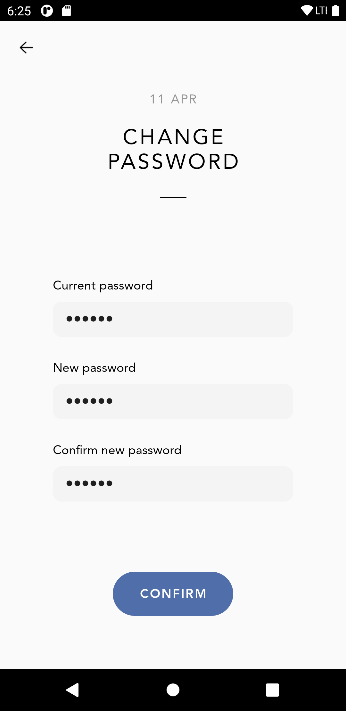
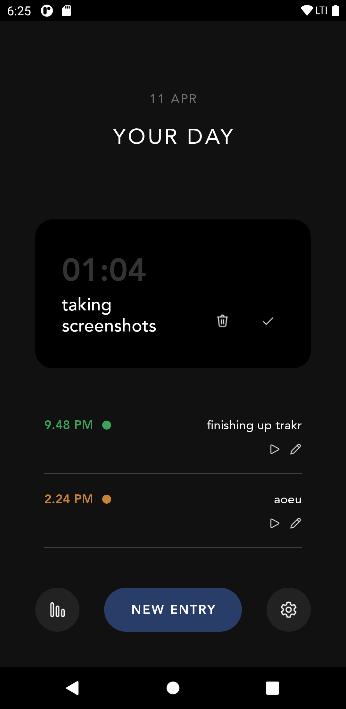
The authentication state is persisted by firebase authentication automatically on the device, and in terms of the app architecture the current user can be accessed from various fragments by UserViewModel. The same userviewmodel is provided by using “by activityViewModels()”. As of now, all the logic is contained in fragment classes.

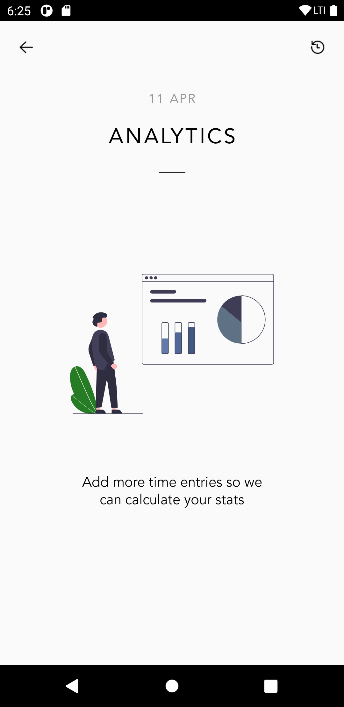
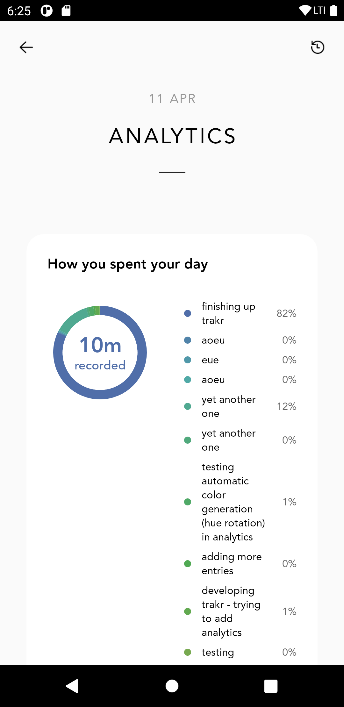
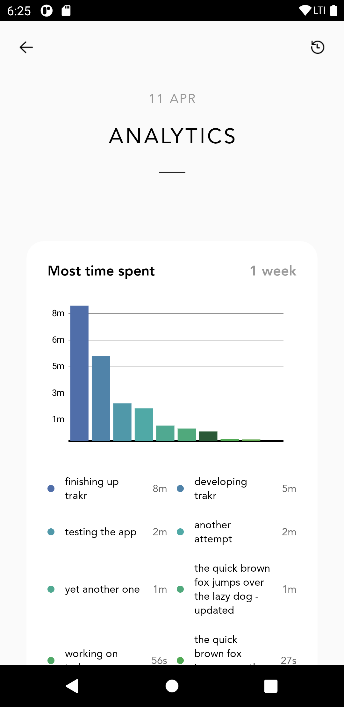
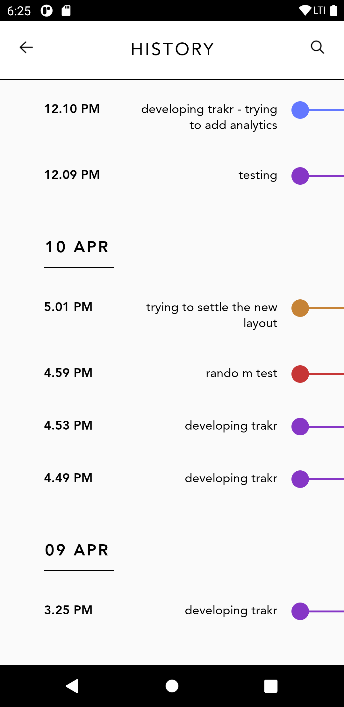
User data is streamed from firebase, allowing time entries to by synchronized across multiple devices. dbViewModel (accessed in fragments using the same method described above) provides the method listenToTimeEntriesToday, which uses a firebase snapshot listener to subscribe to updates. When anything changes, the UI is therefore updated. To prevent an UI update from being triggered after the fragment has been stopped, which would crash the app, the function returns a ListenerRegistration. When a fragment stops, calling onStop(), listenerRegistration.remove() is called to cancel the subscription.

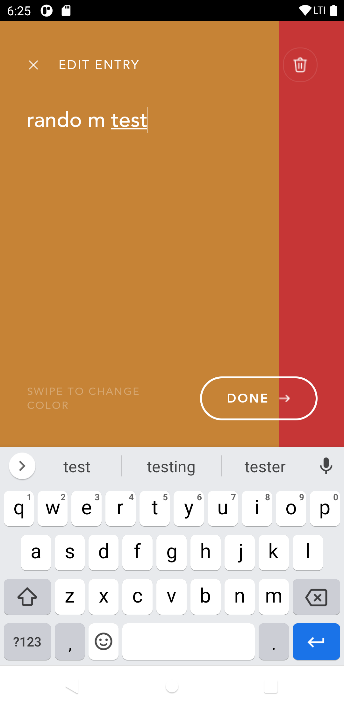
At the top of almost every fragment there is the current date. This reminds the user of the current time, stressing its importance. I decided to create a simple custom view which could be easily duplicated, by extending AppCompatTextView.

Screenshots



Survey of usability

I showed my app to Dave, Yaw Tia, Vernon, Francis and Vikram (at the same time), and asked them to give feedback. In general, their opinion was that the UI was good, however the app lacked certain functionalities, and it would be good to be able to plan one’s time as well as track activities.

Reflection

I faced a few obstacles in this project.

Firstly, implementing the viewpager seen when adding a new entry. Initially I used code from onboarding, however the colors were dynamically loaded and using a viewpageradapter resulted in strange bugs. After some researching online I realized I had to use a recyclerview instead.

Secondly, I decided to use an the MPAndroidChart library to render charts. Unfortunately, the API was unintuitive and in my opinion, quite poorly documented. There were many similarly named functions and classes and it was difficult to guess which did what. After a while I realized it would have been better to draw the charts from scratch, especially since they had a very simple design.

Future improvements include adding tags to time entries, letting the user backup their data to google drive, and searching and filtering time entries.

Overall, this project has deepened my understanding of android development as I learned new ways of using Android Studio such as importing vector graphics, generating app icons and showing open source licenses for a project. I also learned to use selectors to dynamically change text colors as well as backgrounds when a button is pressed.