CanaCache

CSCI 4100U Final Project

Student Name	Github
Reese Dominguez	fenreese
Cole Mollica	coleman2246
Imran Mustafa	Seg-fau1t
Jack Woolvett	object-Object
~~Troy Garcia~~	See *Grievances*

Grievances

Troy Garcia attended the first group meeting and barely contributed to the discussion, but after that we have not heard from him via any method of communication. He has also not made a single commit or contribution to the project in any way.

Contributions

Reese Dominguez

- Initial Firebase setup (refactored by Jack)
- Internationalization
- Cache create/edit page
- UI advising
- Technical report setup/writing

Cole Mollica

- Creating models, provider and generic widgets for settings
- Serializing and deserializing settings
- Custom theming model used by nearly every widget
- Initial Navigation (replaced by Jack after recommendation at check in)
- Map control logic centering, drawing caches on map, local cache info from map,
- Map event listening (hold to add)
- DB interface functions
- Generic charts that could be used to graph any time based data
- Timers to track users actions (distance moved, steps, time spent) write to db
- Debug tools for generating dummy data for stats
- Geocoding for converting coordinates to nearest address for cache information
- General dialogs and pickers
- Created general snack bars for reuse around the project.
- Setup authentication (Google sign, Github) as well as created sign in page
- Part of Functional requirements in technical report

Imran Mustafa

- Mapbox integration
- Parts of the DataTable listing caches
- User guide

Jack Woolvett

- Project file structure
- Abstract, generic Model-View-Controller classes to help with project structure
- Robust Firestore implementation with abstract classes for collections and documents
- Firestore and Firebase Storage security rules
- User profile/account page with avatar, display name, bio, etc.
- User profile item list
- Reminder notifications
- List of caches/items (now replaced with a DataTable of caches)
- Tab bar navigation
- Lots of helper functions and classes
- Lots of refactoring, code quality improvements, etc
- Created the app logo, which definitely doesn’t look like the Conservative Party logo

Troy Garcia

- Literally nothing

Overview

CanaCache is a geocaching app intended to provide everything you would need to go geocaching. It allows you to see local caches and relevant information to them. It also allows you to create your own caches that other users can see. Canacache also allows you to see your relevant geocaching statistics.

User’s Guide

Getting Started

Before using the app it is important to know that you must sign in through a third party account. We allow for the use of a Google account or Github account. Please select one of these options to use for your login.

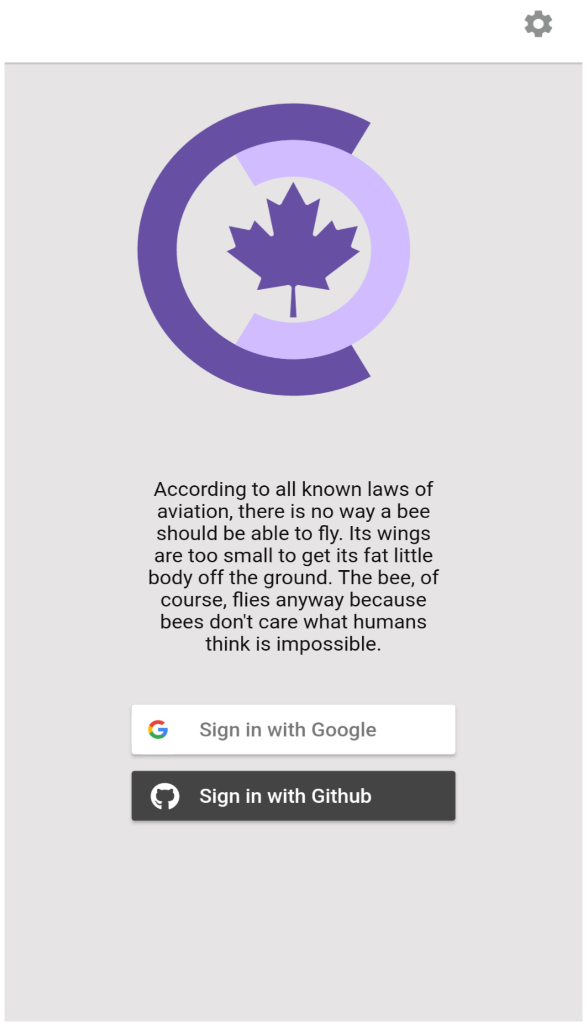

According to all known laws of aviation, there is no way a bee should be able to fly. Its wings are too small to get its fat little body off the ground. The bee, of course, flies anyway because bees don't care what humans think is impossible.

Figure 0: Sign In

If this is the first time starting the app, it will ask you for permissions that are necessary for the app to work properly. Please accept these before continuing.

Home

After you have logged in, you will be deposited onto the home page, this page contains a map that is centered around your current location. You will be able to move around the map and view caches that are located around you. Tapping on them will result in a pop up explaining cache information such as name, location, etc. (below is a figure of the home screen)

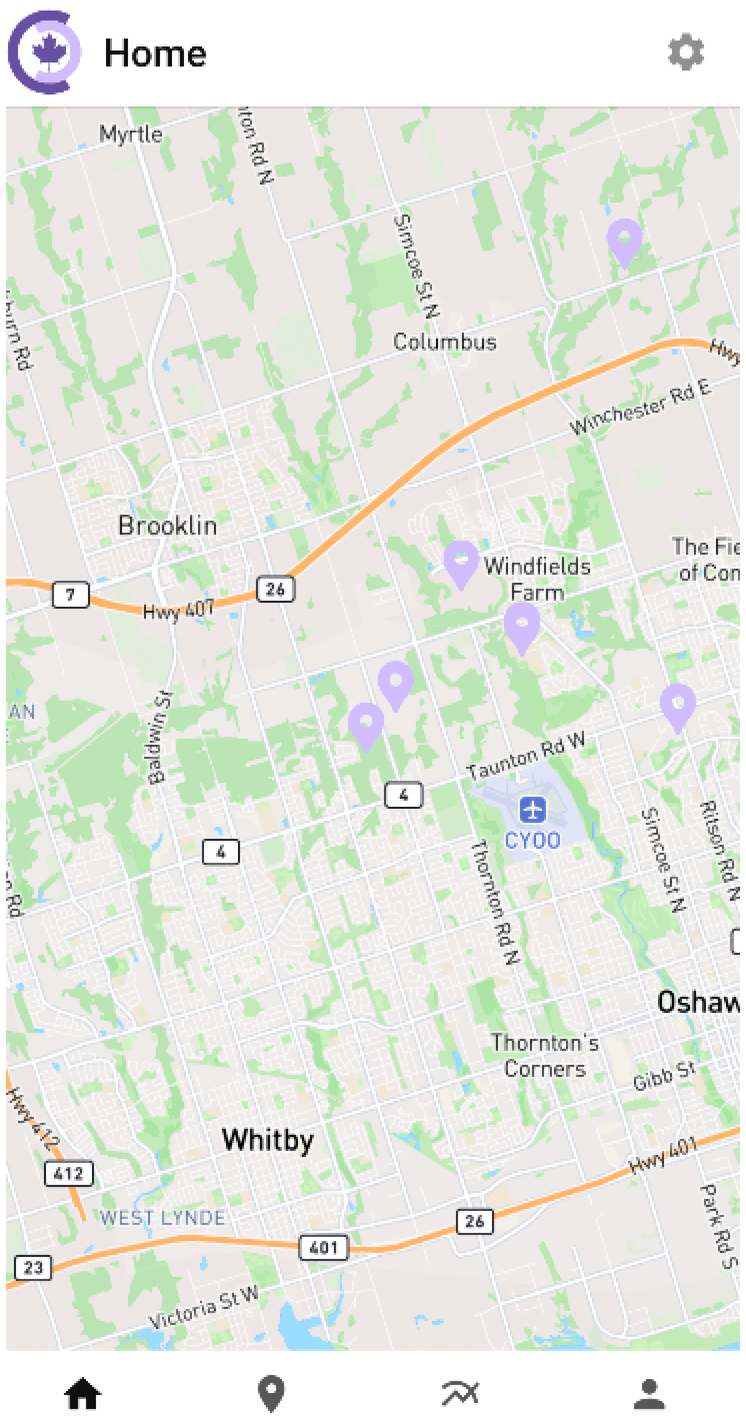


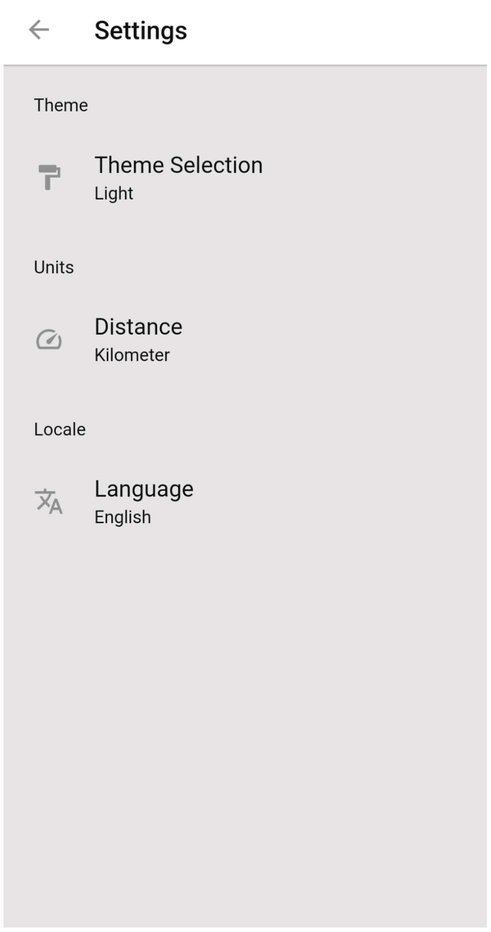
Figure 1: Homepage

From the home screen you will be able to navigate to the rest of the app through the bar at the bottom. Each icon (from left to right) represents respectively the home screen, cache list, stats page and account page. Pressing on any of these icons will take you to that respective page.

Pressing any of the cache icons on the map will open a pop up that will display relevant information about it and allow you to edit the cache or delete it. Though only if you are the owner. Holding down anywhere on the screen in the homepage will prompt you to create a new cache at the center of the map.

Settings

At the top right of the app you will see a gear icon, tapping on this gear icon will send you to the settings menu. This menu allows you to change the theming of the app, the default unit of measurement, and the language that the app uses. If you wish to exit the settings menu simply press the backwards arrow at the top left of the screen.


Settings

Theme

Theme Selection
Light

Units

Distance
Kilometer

Locale

Language
English

Figure 2: Settings

Cache list

This page of the app will show you a list of caches, Both their name and location. You will be able to sort them either on the alphabetical, as well as favorite them.

Stats page

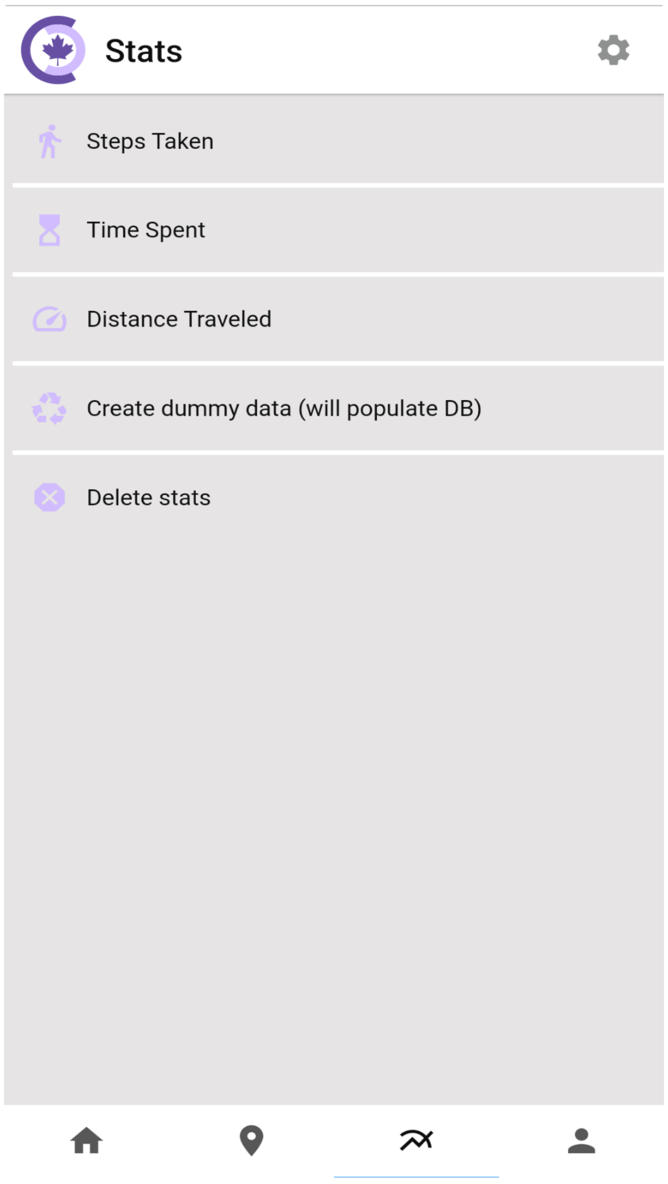


Figure 3: Stats Navigation Page

This page contains all of the data that the app tracks while in use. Pressing on any of the top three buttons will take you to a graph of that specific data point (as seen below) as well as the ability to choose a time frame to view it on (days, weeks, months, years). The app keeps track of the amount of steps taken, time spent on the app, as well as the total distance traveled. At the bottom of the list there is an option to clear all data stored by the app. Note that the stats are not saved in the cloud. Also do note that the button 'generate dummy data' is included in the app to help with development and testing, and will only be visible when the app is built in debug mode.

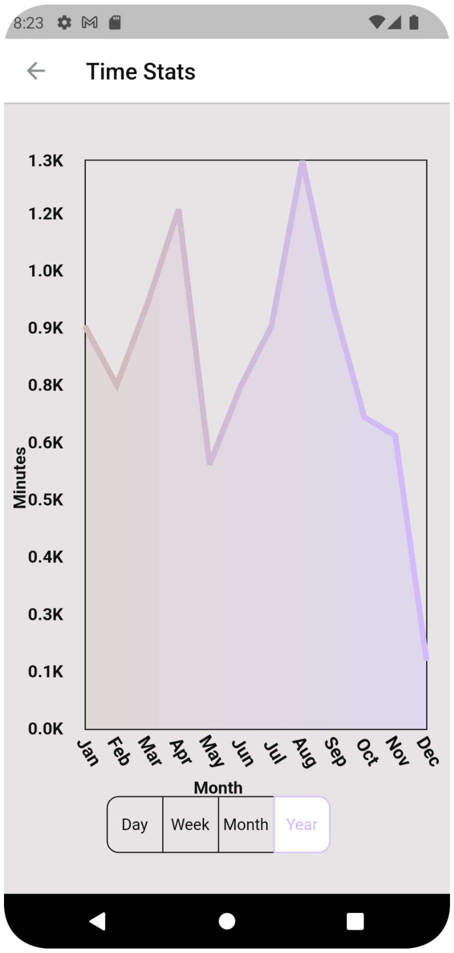


Figure 4: Graph

Account management

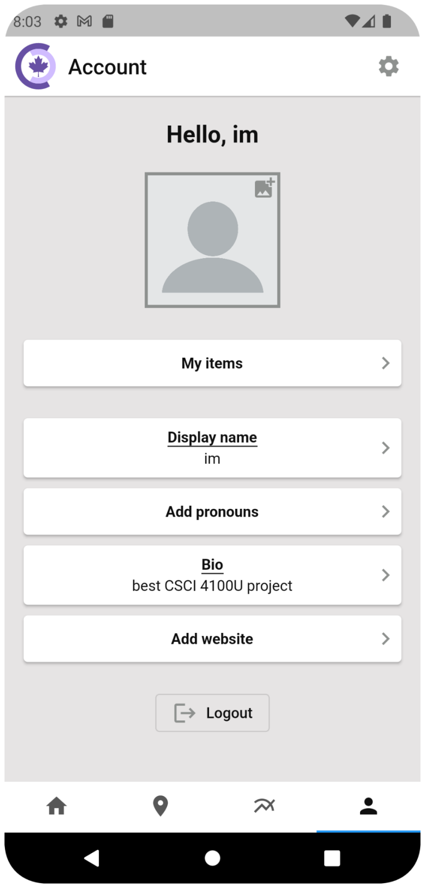


Figure 5: Account Page

This is the page of the app that will allow you to make changes to your CanaCache user account. You will be able to edit your profile pic, username, preferred pronouns, bio, and website link. Simply press on the respective button to do so, or in the case of the profile picture just press on your current profile picture.

This is also the page where you can view items that you have collected from caches you have visited. By pressing ‘My items’ you will be taken to another page listing all of the items collected, along with the date you got them.

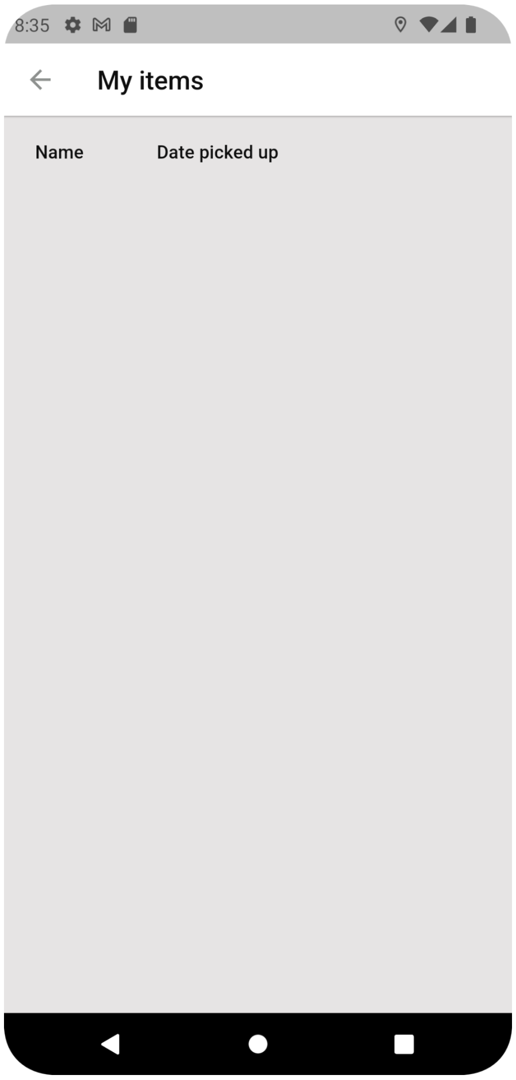


Figure 6: My item list

At the bottom of the account page you will see the logout button. You can use this to sign out of the app with your current account. After pressing it you will be taken back to the login page.

Functional Requirements

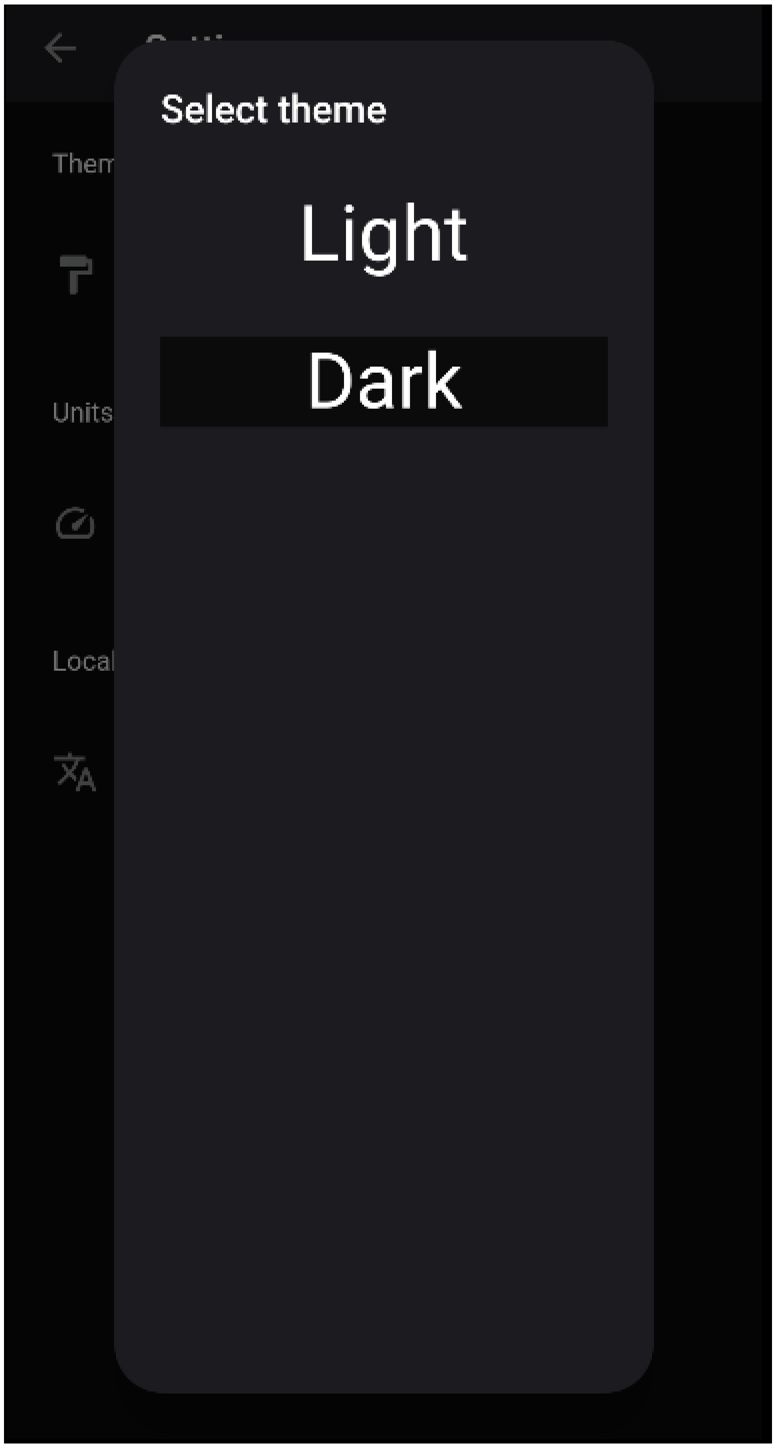
Multiple Screens & Navigation

Before login the only navigation available to you is to sign in or look at settings. The main navigation is available after login. Navigation is done in the app through a TabBar at the bottom of the page. There are 4 main pages (homepage, caches, stats, and user profile)

Dialogs & Pickers

Dialogs & pickers were utilized in the settings section. Every setting can be chosen via a dialog and picker. A screenshot can be seen below.

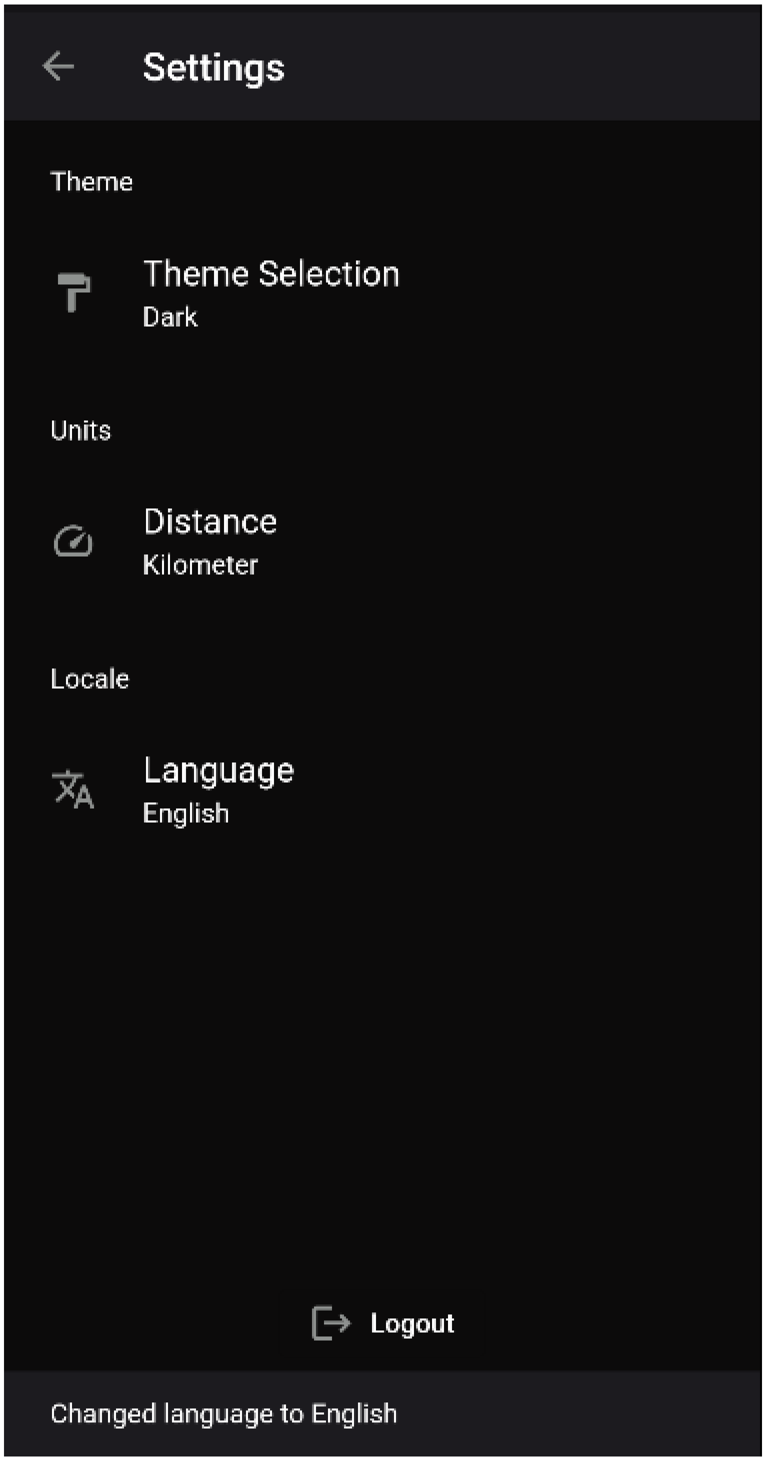
Dialogs are also used in the Account page for editing display name, bio, etc.



Snackbars & Notifications

Snackbars appear on changes such as login state (when a user is successfully logged in or out) or setting change. The snackbar made for that section was also reused in the stats debugging tools.

As for notifications,
It’s been a while
You haven’t opened CanaCache for a long time. Ready for some geocaching?
You can see a screenshot of both features below.



Settings

Theme

Theme Selection
Dark

Units

Distance
Kilometer

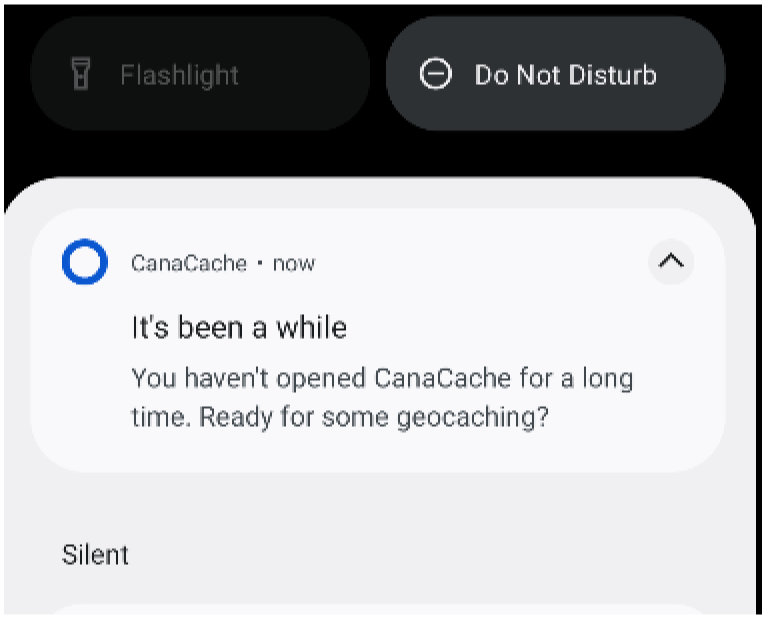
Locale

Language
English

Logout

Changed language to English

Figure 7 : Snackbar



It's been a while

You haven't opened CanaCache for a long time. Ready for some geocaching?

Figure 8: Notification

Local Storage

App settings are stored in a .json file. On load, the app creates a SettingsProvider from settings.json, where the app is able to instantly change appearance, unit display, and language. SQLite is also used to store personal statistics, such as steps taken, distance travelled, and time spent in the app. All statistics are tracked on a per hour basis. There also exists debugging features (will only show up when the app is compiled in debug mode). The ability to populate the statistics tables with random data as well as clear all tables is available.

Cloud Storage

Firebase is used for all of the cloud storage features: authentication is implemented with Firebase Auth, caches and user profile data are stored in Firestore Database, and profile pictures are stored in Firebase Storage.

Data Tables & Charts

Various statistics about the user of the app are tracked (mentioned in the Local Storage section). The flutter library fl_chart (https://pub.dev/packages/fl_chart) was used to generate all of the charts. It was picked for its high level of customization. Every statistic tracked can be viewed on 4 different time frames (last day, last week, last month, last year). A screenshot of some graphs can be seen below.

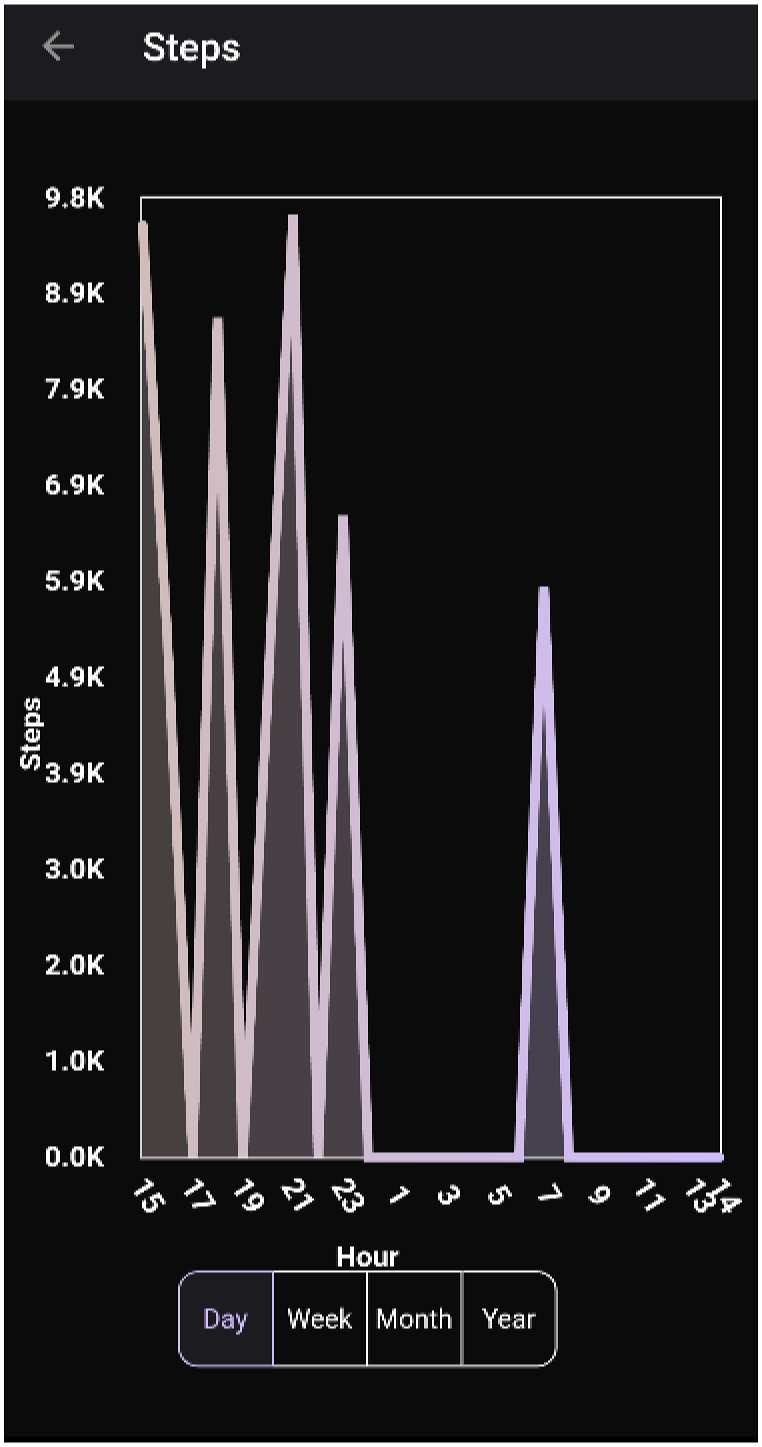


Figure 9: Day Time Frame Chart

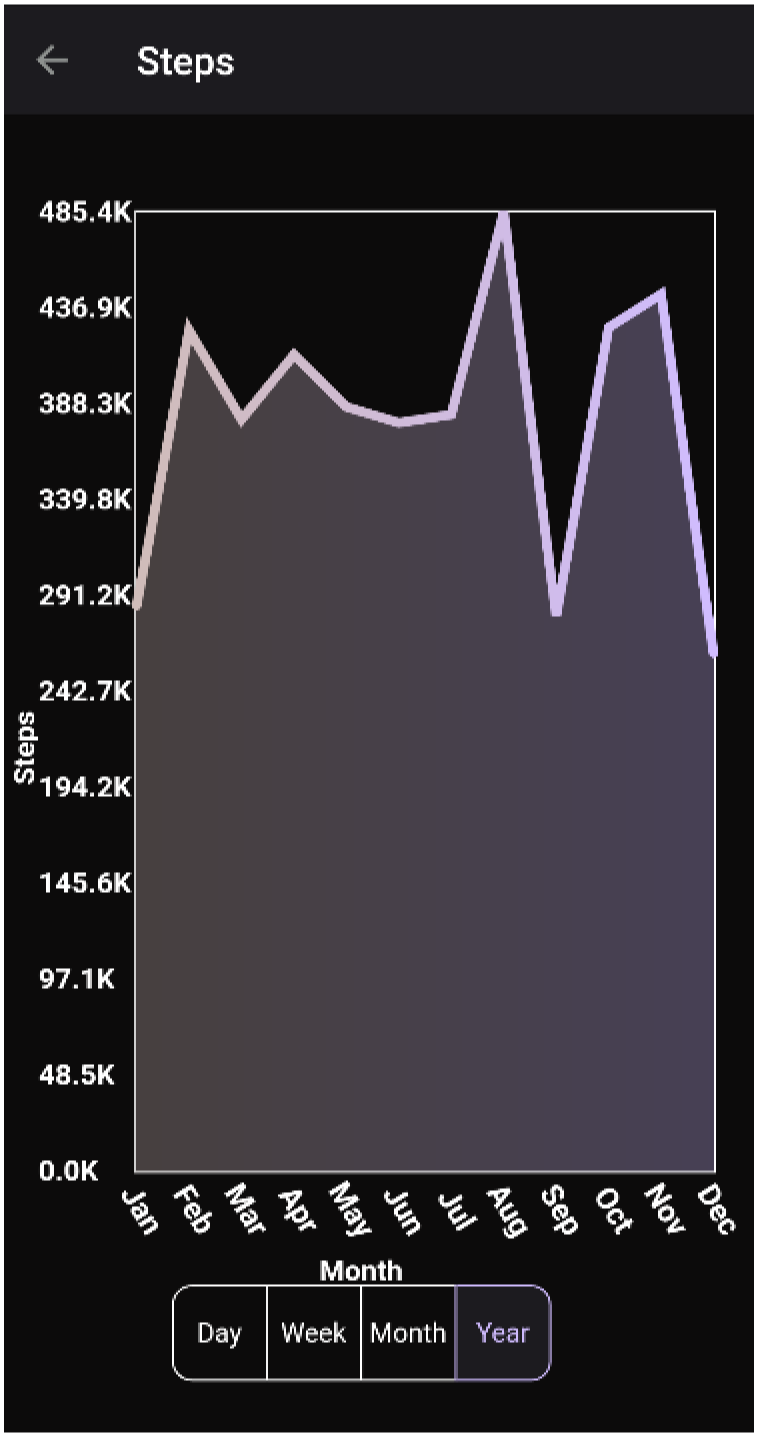


Figure 10: Year Time Frame Chart

Along with the use in the stats page, the app also uses a DataTable when displaying the list of caches. This table displays two columns, the cache title/name along with its coordinates. Both of these columns are fully sortable, the title is sorted by alphabetical order. While the location is sorted by distance to the current location.

Maps

The library flutter_map (https://pub.dev/packages/flutter_map) along with mapbox (https://www.mapbox.com/) was used to draw a map and overlay the nearby caches on it. When the map is initialized it will center to your current location and draw all caches that have a Manhattan distance that is less than 20km. Each cache is tappable on the map and will show relevant information when pressed. Caches should resize on zoom as well. Screenshots are below.

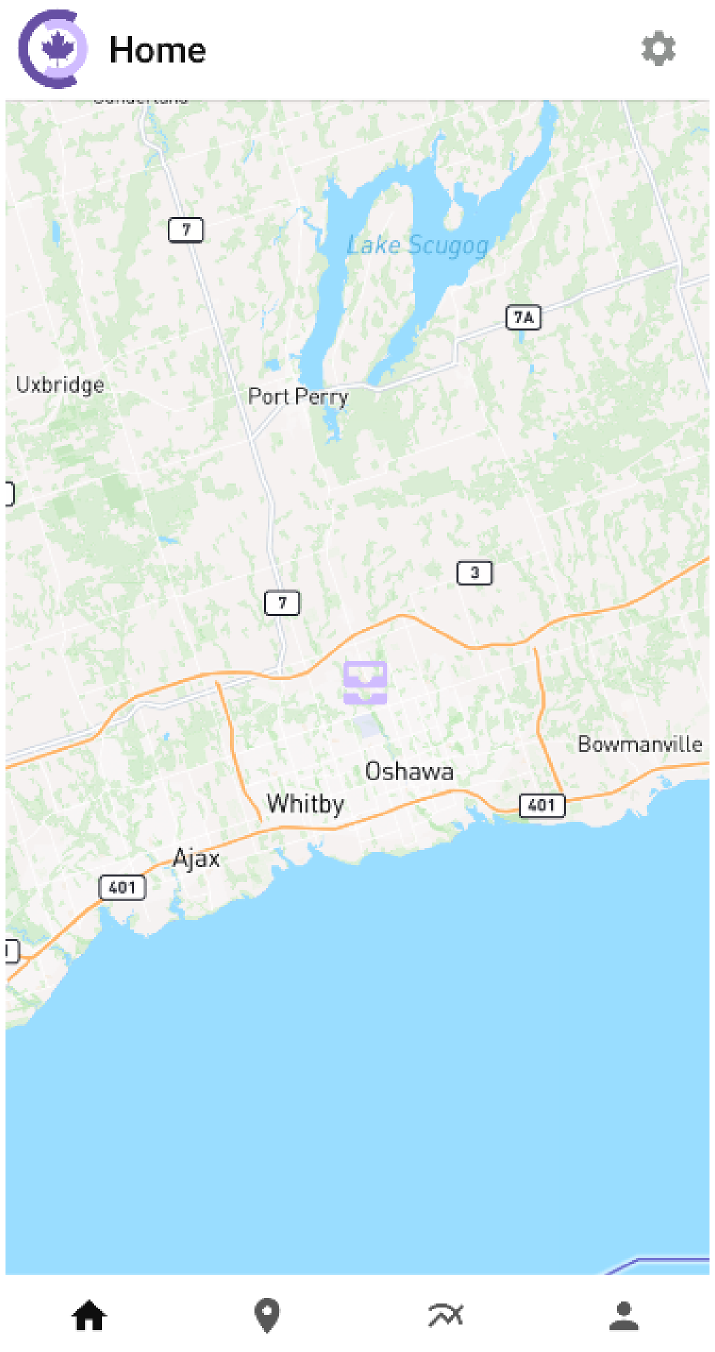
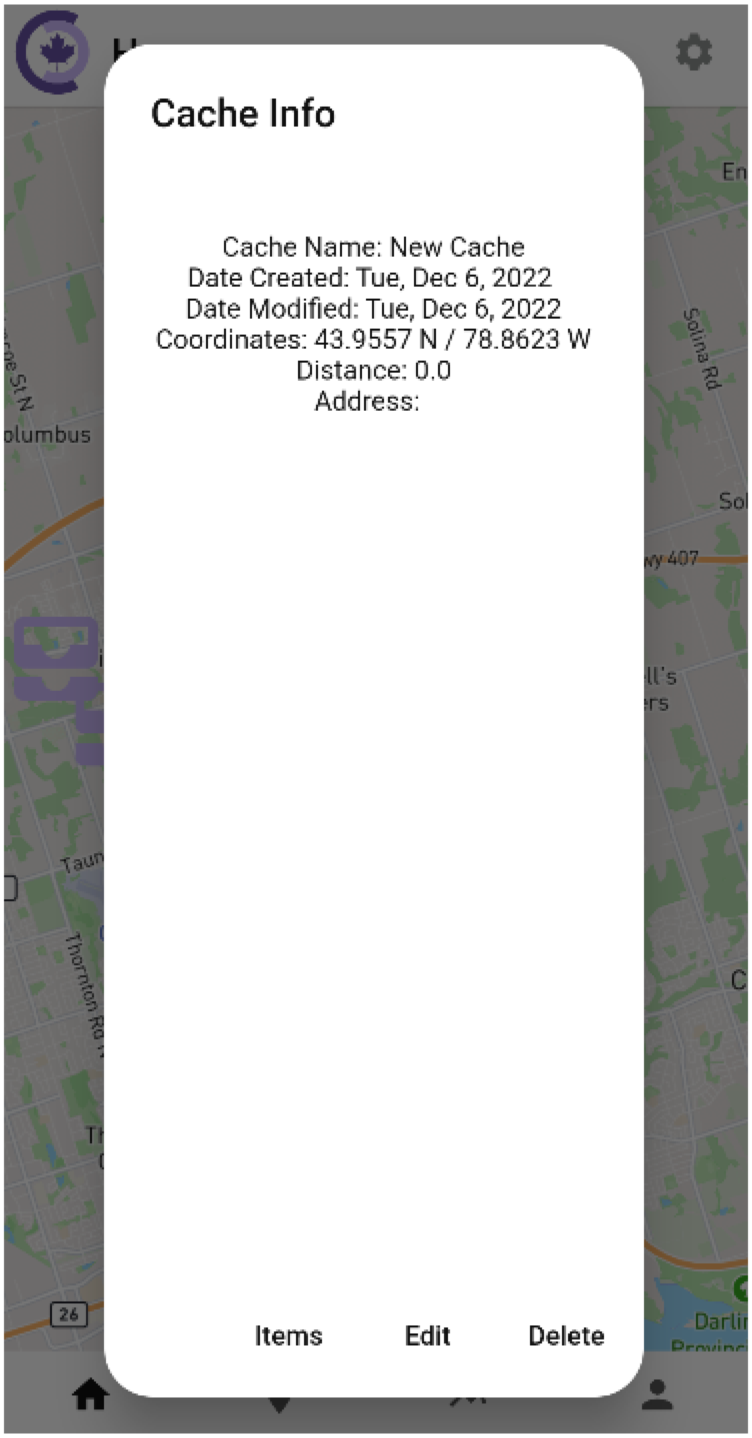


Figure 11: A Cache Drawn on Map


Cache Info

Cache Name: New Cache
Date Created: Tue, Dec 6, 2022
Date Modified: Tue, Dec 6, 2022
Coordinates: 43.9557 N / 78.8623 W
Distance: 0.0
Address:

Figure 12: Cache Info Accessed Via Tapping Cache

Geolocation

The library geolocator(https://pub.dev/packages/geolocator) was used for getting the current location. This is used in the drawing of the map, to track distance for stats, and to get the distance to caches on the Caches page. Both a stream and non steam version of geolocation is used.

Geocoding

The geocoding (https://pub.dev/packages/geocoding) library was used for geocoding. Since the caches are recorded as simple latitudes and longitude geocoding was used to get the address so when you tap on a cache it gives the nearest address to it in the popup.

Internationalization

Internationalization is made possible with the help of the flutter_translate library. (pub.dev link: https://pub.dev/packages/flutter_translate) Compared to the flutter_i18n library used in the lecture demos, this library was more simple to use, and had more clear documentation.

The non-English language we decided to translate the app into was Brazilian Portuguese, due to us knowing more than one person fluent in the language.

Code Design

Overview

With regards to code design every feature was separated from each other to allow modularity. The model view controller design pattern was also utilized within each feature to allow for ever more modularity. Heavy usage of templates and inheritance was used where possible to allow for reusability of code as well.

UML Diagrams

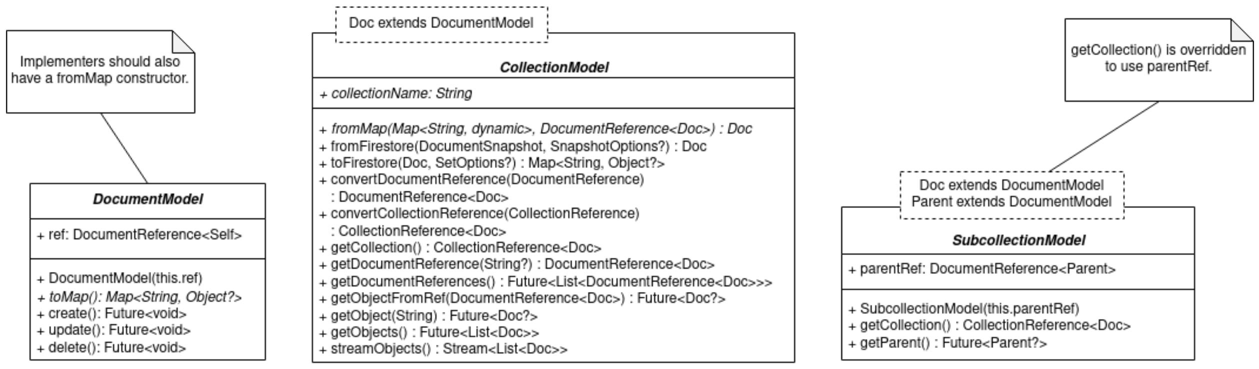

Implementers should also have a fromMap constructor.
DocumentModel
+ ref: DocumentReference<Self>
+ DocumentModel(this.ref)
+ toMap(): Map<String, Object?>
+ create(): Future<void>
+ update(): Future<void>
+ delete(): Future<void>
Doc extends DocumentModel
CollectionModel
+ collectionName: String
+ fromMap(Map<String, dynamic>, DocumentReference<Doc>) : Doc
+ fromFirestore(DocumentSnapshot, SnapshotOptions?) : Doc
+ toFirestore(Doc, SetOptions?) : Map<String, Object?>
+ convertDocumentReference(DocumentReference) : DocumentReference<Doc>
+ convertCollectionReference(CollectionReference) : CollectionReference<Doc>
+ getCollection() : CollectionReference<Doc>
+ getDocumentReference(String?) : DocumentReference<Doc>
+ getDocumentReferences() : Future<List<DocumentReference<Doc>>>
+ getObjectFromRef(DocumentReference<Doc>) : Future<Doc?>
+ getObject(String) : Future<Doc?>
+ getObjects() : Future<List<Doc>>
+ streamObjects() : Stream<List<Doc>>
getCollection() is overridden to use parentRef.
Doc extends DocumentModel
Parent extends DocumentModel
SubcollectionModel
+ parentRef: DocumentReference<Parent>
+ SubcollectionModel(this.parentRef)
+ getCollection() : CollectionReference<Doc>
+ getParent() : Future<Parent?>

Figure 13: Firebase Local Diagram

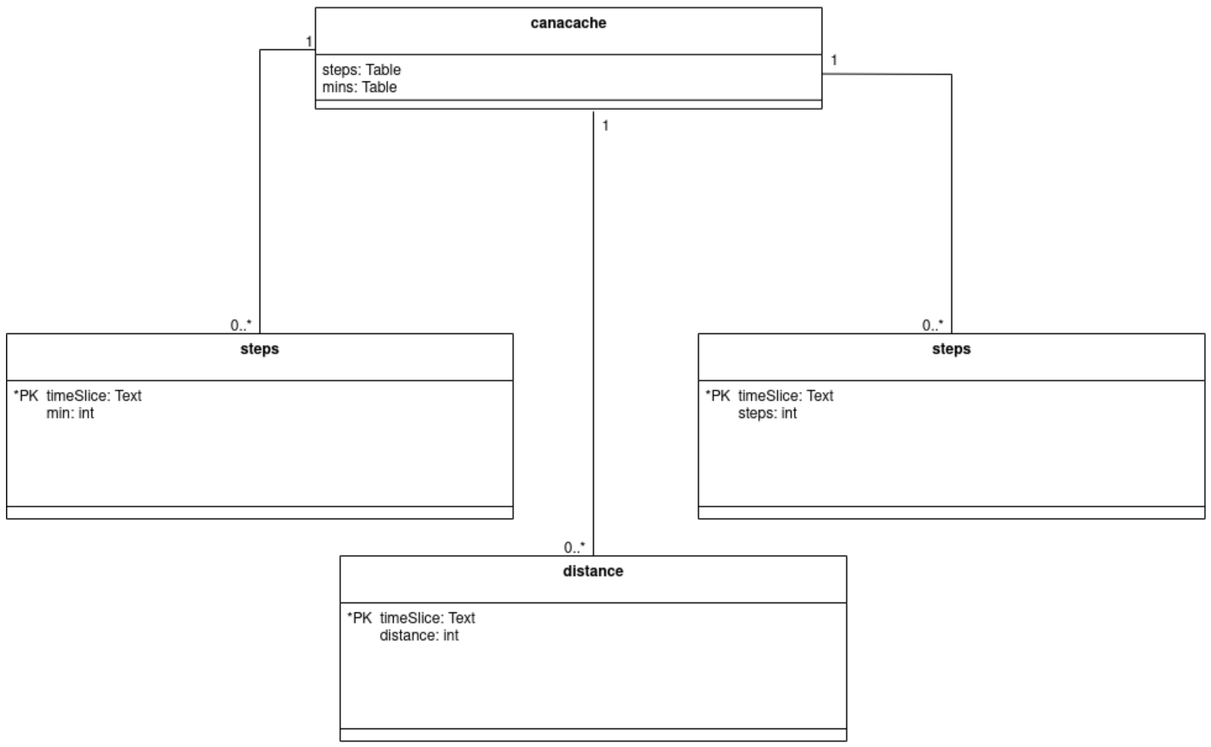
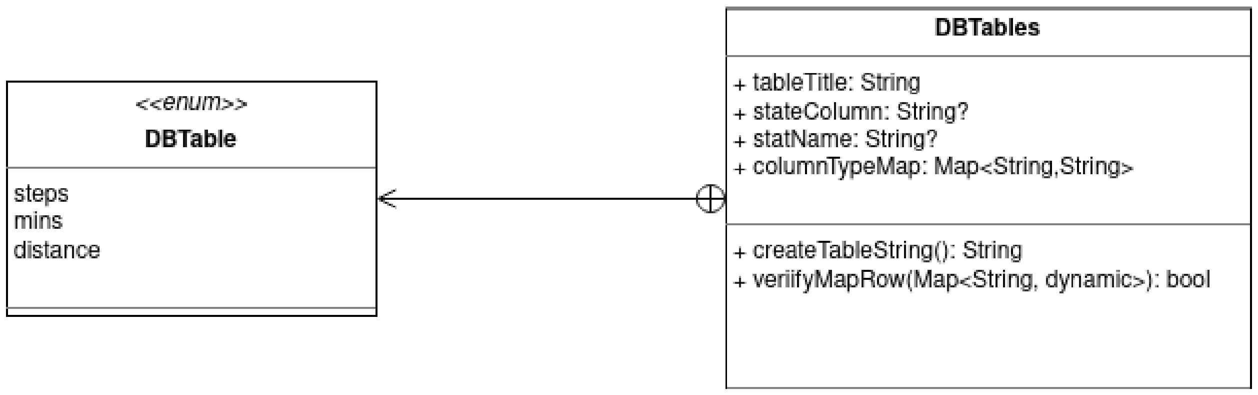


Figure 14: Local DB Diagram



<<enum>>
DBTable
steps
mins
distance
DBTables
+ tableTitle: String
+ stateColumn: String?
+ statName: String?
+ columnTypeMap: Map<String,String>
+ createTableString(): String
+ veriifyMapRow(Map<String, dynamic>): bool

Figure 15: Local DB Code Diagram

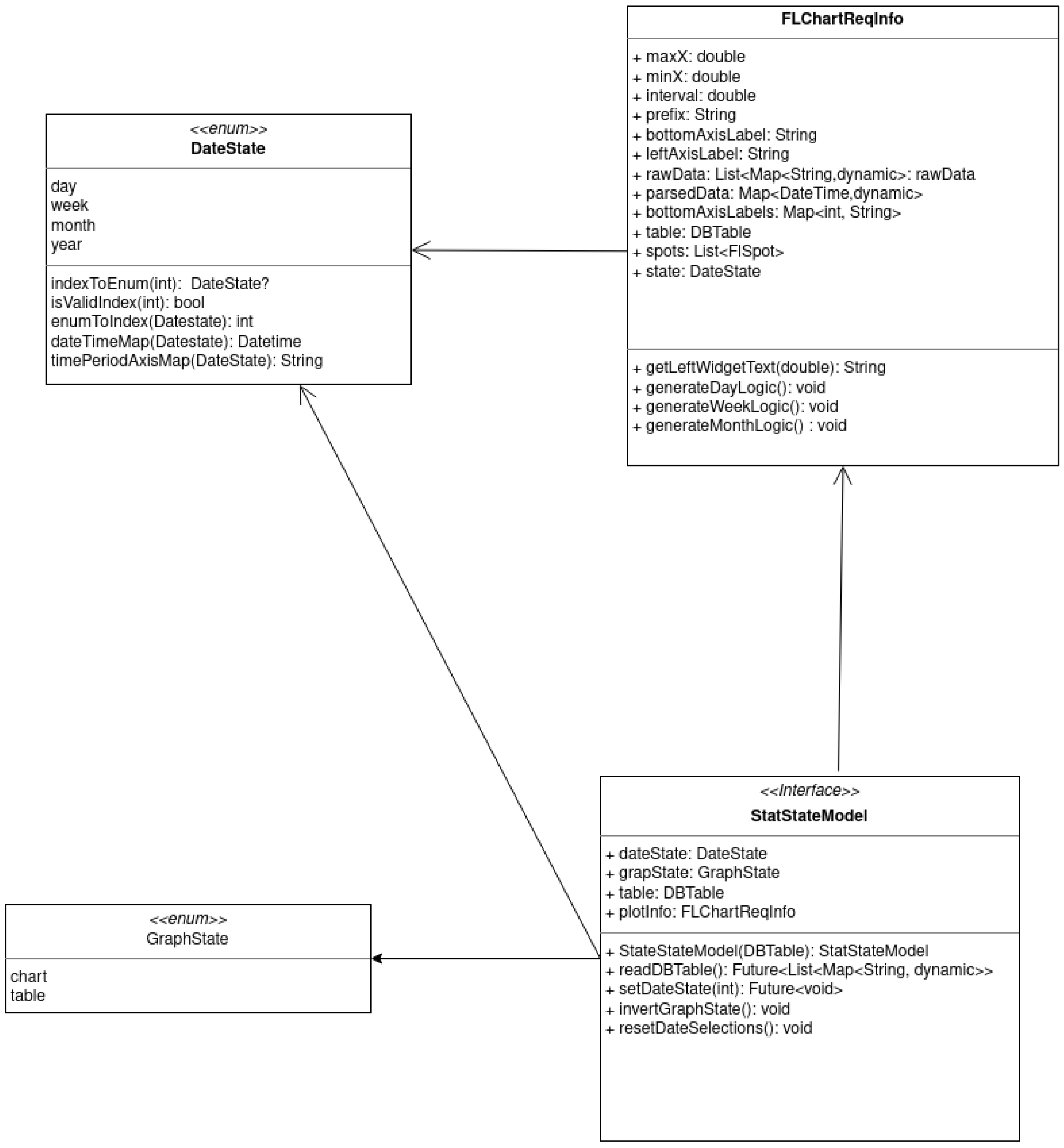

FLChartReqInfo
+ maxX: double
+ minX: double
+ interval: double
+ prefix: String
+ bottomAxisLabel: String
+ leftAxisLabel: String
+ rawData: List<Map<String,dynamic>: rawData
+ parsedData: Map<DateTime,dynamic>
+ bottomAxisLabels: Map<int, String>
+ table: DBTable
+ spots: List<FlSpot>
+ state: DateState
+ getLeftWidgetText(double): String
+ generateDayLogic(): void
+ generateWeekLogic(): void
+ generateMonthLogic() : void
<<enum>>
DateState
day
week
month
year
indexToEnum(int): DateState?
isValidIndex(int): bool
enumToIndex(Datestate): int
dateTimeMap(Datestate): Datetime
timePeriodAxisMap(DateState): String
<<Interface>>
StatStateModel
+ dateState: DateState
+ grapState: GraphState
+ table: DBTable
+ plotInfo: FLChartReqInfo
+ StateStateModel(DBTable): StatStateModel
+ readDBTable(): Future<List<Map<String, dynamic>>
+ setDateState(int): Future<void>
+ invertGraphState(): void
+ resetDateSelections(): void
<<enum>>
GraphState
chart
table

Figure 17: Chart Stat States