This main firestore project/database acts as the directory of all places. A collection for user data is stored here as well, and customer information may have a collection disctinct from place (one customer - multiple places?). The place information stored is all general information about a place: addess, phone, hours, cost, etc. Also image(s), and news/events with their image(s). There will be a reference to the project which contains the item collection for this place. What if a place has very few items? Administrators of this place can edit this general information, and their data is all contained in a single document with subcollections.
The Place collection will be searched by geolocation from the app, possibly within a small - *where am I* - or wide - *places near me* - radius, and also by name (in multiple languages?)

Collection: Place	
id : *Auto-generate*	let firestore generate a unique name
prefix: *String*	in ascii chars to generate a more readable unique ID
name: *Map* {lang, text}	where lang code is the key, and the name (in the language) is the value
description: *Map* {lang, text}	where lang code is the key, and the desc (in the language) is the value
geolocation: *Geopoint*	Lat/Lng coordinates to as close as possible
location: *Map* {lang, text}	where lang code is the key, and the location (in the language) is the value
owner: *Reference*	to the Customer Collection
defaultLanguage: *String*	the language code
languages: *Array*[*Map* {code, name}]	an array OR map of maps with language names in various languages
levels: *Array*[*Map* {id, lang, name}]	an array or map of maps, but level names may be in different languages, so level ID is a number
defaultLevel: *Number*	Should this be a fixed number, like 0?

Each Place is owned by a customer. Prefix is a code to add to this place's item names. It has a subcollection of Images (but they could be in an array since they will not exceed 10 or so.
The default level should have the id 0, so that if the place has no levels, every level-distinguished field just has 0. The UI must manage this.
Many more fields will be in here: Hours of operation, floor maps, etc.

SubCollection: Images
id: *Auto-generate*
url: *String*
caption: *Map {lang, text}*

This should probably be in the document itself as an array. Just showing like this for thought. Same with the images next door.

Subcollection: News
title: *Map* {lang,text}
shortText: *Map* {lang,text}
longText: *Map* {lang,text}
imageURL: *String*
addedDate: *Date*
expireDate: *Date*

SubCollection: Images
id: *Auto-generate*
url: *String*
caption: *Map* {lang, text}

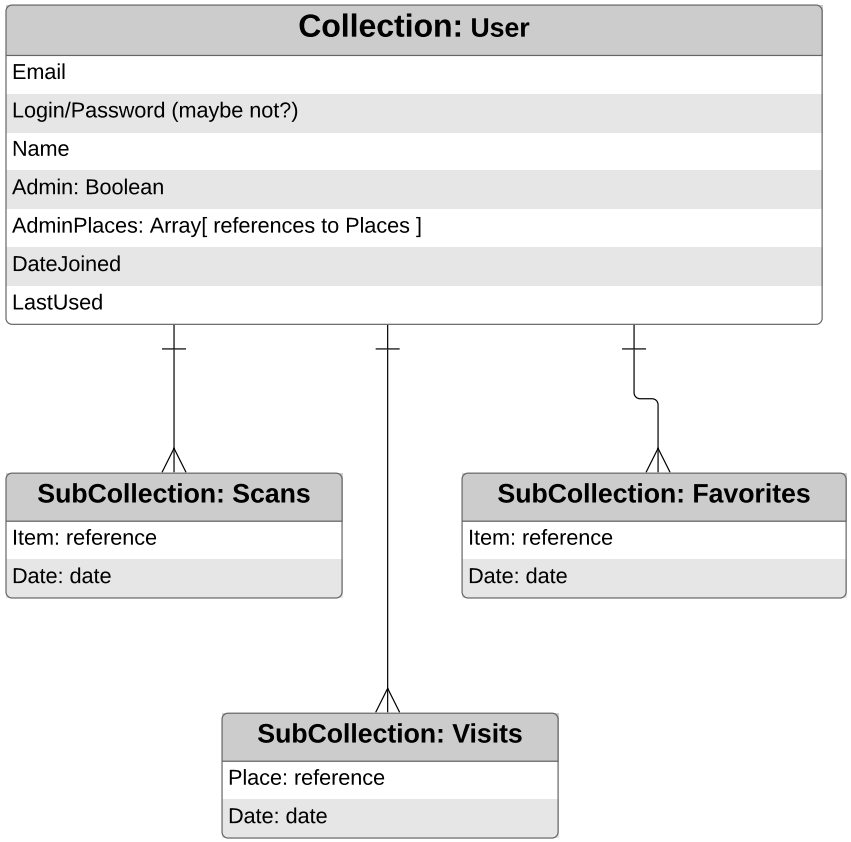
News & events subcollection also may include images within, since there can't be so many. One image for the list screen, many for the details screen: same thing with the short and long text: one will be rich text, the other simple.

REMOVE: SubCollection: Items
name: *Map* {lang,text}
location: *Map* {lang,text}
item: *Reference* to Item Collection

This is not possible with the items stored in a different collection, and not worth the upkeep.

The subcollection of Items helps for populating the list of items for editing, without bringing the full data of each item - these 2 fields are selected for sorting. The denormalization must be managed on changes. The number of items may be huge and should be selected using paging.

Since users exist across places, they are also stored in the top level project.
Users may be admins or regular. Admins may be admins of multiple places. Maybe there are also superuser users (developers), who are also regular users. Although these subcollections are small, they may be very large numbers. Some denormalized data may be stored too, as we develop the history functions for users. In the meantime, we want places visited (based on geolocations or a scan), scans - storing duplicates is fine. Finally favorites is there too. Also the counts should be stored here and there.



Comments are a requested, and natural, feature, but a user-to-item mapping is difficult, as it is even with favorites and maybe some other of the user data. Maybe they are better stored with the item?

Collection: Comment
User: reference
Place/Item: reference
Text: String
Date: date

Collection: Customer
ID
CompanyName
Login

Think about this later. This is customer data: a customer may have multiple admin users, that's different. This customer entity/user is about billing and configurations of possibly many places.

For each place, there is an item collection in its own project.

Collection: Item	
ID: *Auto-generate*	Field
ShortName: *String*	Place prefix + number
Name: *Map* {lang, text}	Field
Location: *Map* {lang,text}	Field
GeoLocation: *GeoPoint*	Field
Description: *Array*[*Map* {lang, level, text}]	Better a map of maps, indexed first by level
AudioURL: *String*	Field
VideoURL: *String*	Field
Images: *Array*{ *Map*{ URL, captions: Map{ lang, text} }]	Or is this a map of maps? Is the index meaningful?
HasQuestions: *Boolean*	Field
Owner: *Reference*	to Place

Here we put the images in an array, which should be OK, instead of a subcollection. The description in all its forms will take up the most space. One audio and one video are allowed.

SubCollection: Questions
Level: Number
Text: Map{ lang, text }
Answers: Array[Map{ lang, text, correctFlag }]
ImageURL: String

Note the denormalized HasQuestions flag to be updated by the UI. All permutaions of the questions and answers are stored together in an array. A question gets one image, which is one that belongs to the Item, but does not require a caption.
Can the answers be a Map of Maps, with each language code as a key? No, not exactly, because of the multiples, maybe a map with an array inside?