Meeting 1-3 PM. We're all here. Jie was 7 minutes late. Jie's taking meeting minutes right here, and Nick is updating the sprint planning on the Github as we go.

Aparna presents her map stuff. She's opening up Google Maps and it's able to store the locations.

Action items: Get latitude and longitude. Upload summary document. Contain as many features and we can within our app and not the third party app.

Zarif presents the database schema. Firestore lets you nest collections within other collections. Each user has a document. In there are four fields with lists of user IDs, and a nested collection (subcollection) with more fields, and another collection inside. When we create a document, we need to give it a unique id for the document name. When updating the database, stick with the schema and don't change the names of the collections and stuff.

Nick comments that we should only update the database when we're storing and changing stuff there. Can you pass data into an intent? Serialize the object, send to intent, and when you receive it you have to unserialize and access the object.
~~Nick says it's cleaner to have a data class where we get it from the database and access the properties in the program, instead of accessing the database to get every little thing individually. We're assuming that we want to rely on our classes and rely on Firestore as little as possible. Reduce dependency on Firestore. Don't use Firestore as containers to update our lists.~~
~~Zarif says that the only reason we use the classes is to store it in memory. When we update it, we update in memory and the database, so there's a chance of inconsistency. It might be better to just use the Firestore. Keep fetching from Firestore. It's simplest, but not good for performance.~~
We shouldn't have a parallel copy of the whole database locally. We should fetch from the database, create objects locally which are up-to-date, and then display it. The database does all the memorization, so we don't need to synchronize the parallel copies of the DB. The User, Habit, and HabitEvent classes will bridge the gap between the UI and the DB. We don't really need a FriendRequest class, but we do need a list locally to keep that stuff.

Paola already made the classes and made a PR and added Zarif and John to it, so we can approve and merge that.
Zarif wants to reuse the User class instead of FriendRequest and leave the password blank. Nick suggests, what if we extend? Should we let the password be passed? It'll be hashed and salted and peppered and spiced and f*****, so it's fine if it gets out.
We want to add more methods to User to let them accept friend requests. Take a user ID and remove from one list and add it to another. Do some querying and update each other's lists.

Jie hasn't done much of the images and has nothing functional to show. Following a tutorial to upload and download and display images using Firebase, but it'll have to be adapted to work with Firestore instead. Firebase is recommended for images, and Firestore has a 1 MB limit for each document. But it should work, and suffice for this project.

We discussed how to visually show the progress. Nick is on this. We thought about tracking the progress using a graph showing the target and also a cumulative view. Nick proposes a score system which is basically also showing for each day what’s the target and actual.