GitHub Username: oluwasegunmaths

Patient Record

Description

This app stores vital clinical information of patients and makes it easy for medical practitioners to find clinical information of any desired patient they have interacted with in the past. This is very useful to serve as a backup for medical data so that practitioners would have access to such data whenever they need to make critical lifesaving decisions.

The app eliminates the possibility of lost patient information which occurs whenever case notes of patients go missing. This is especially true and useful in third world countries where due to poor health records, almost all medical practitioners would have to deal with loss of previous and sometimes critical patients record.

As a medical doctor with specialization in mental health, I face the problem of loss of vital patient records due to missing case

notes almost on a weekly basis. This is one of the driving factors behind developing this app.

Intended User

Medical practitioners and other health workers who interact with patients should find the app useful

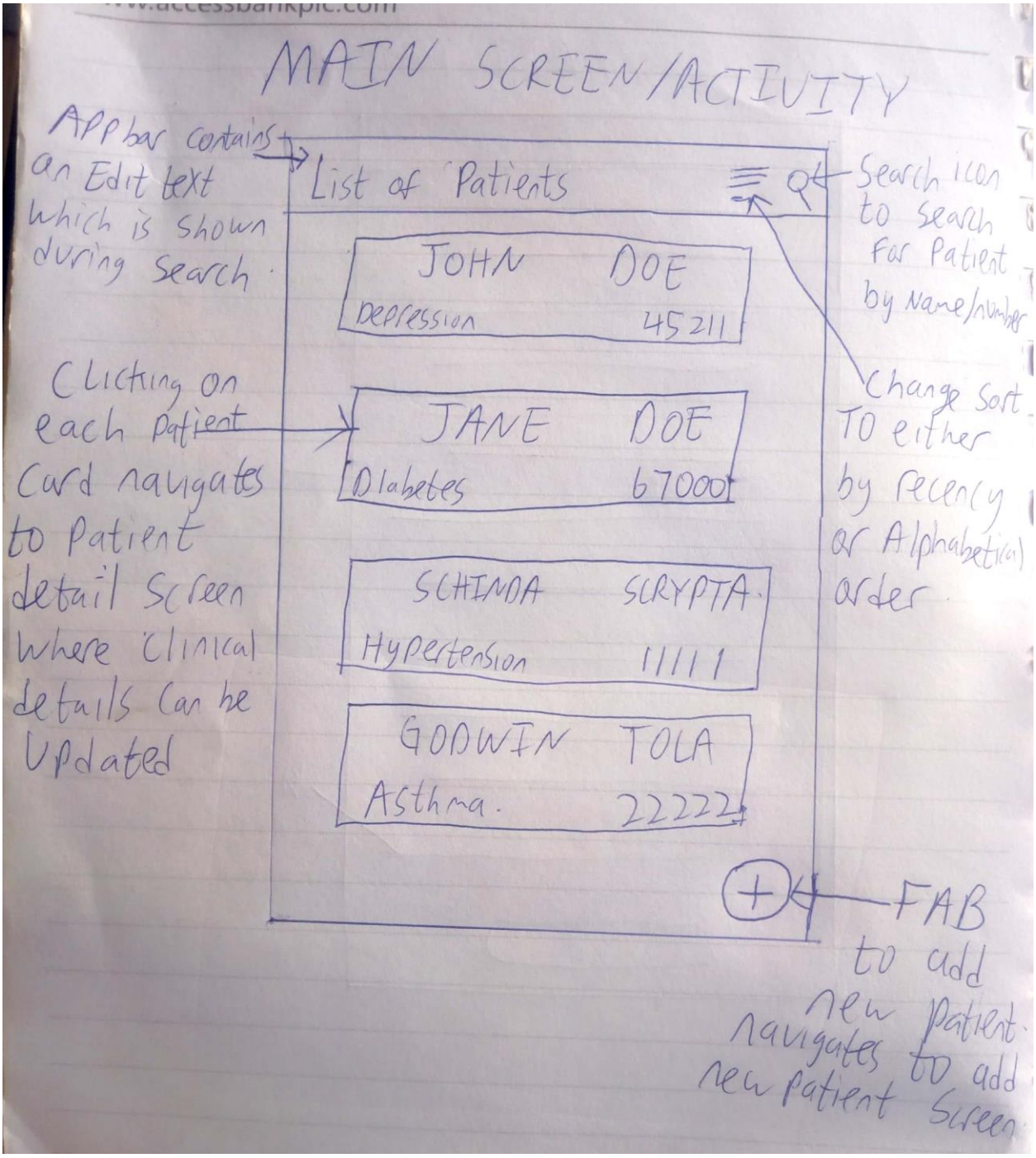
Features

- Saves Patients Details
- Users can easily search for any patient details stored in the app
- Patient details are also backed up online in a firebase database

User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

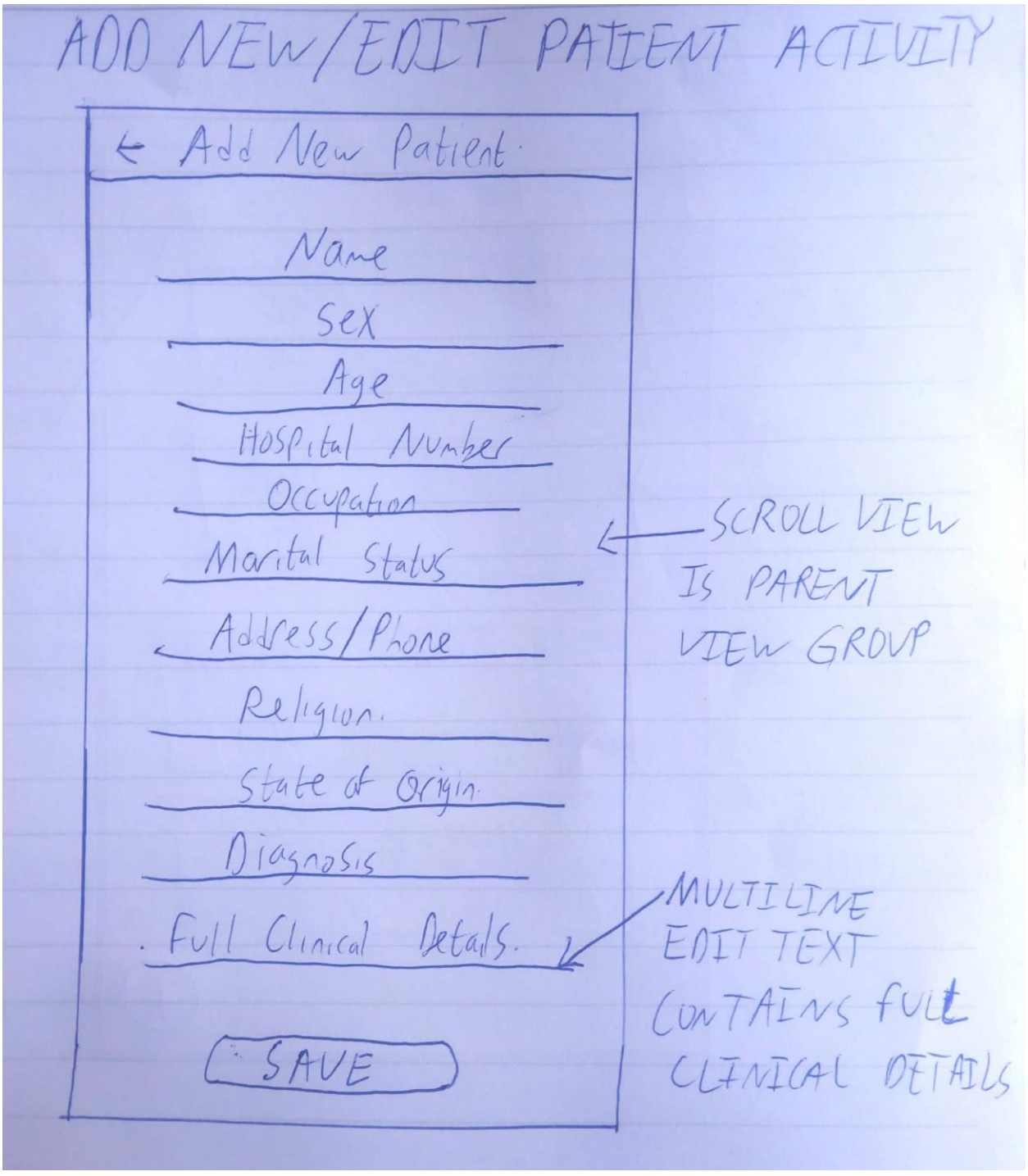
Screen 1

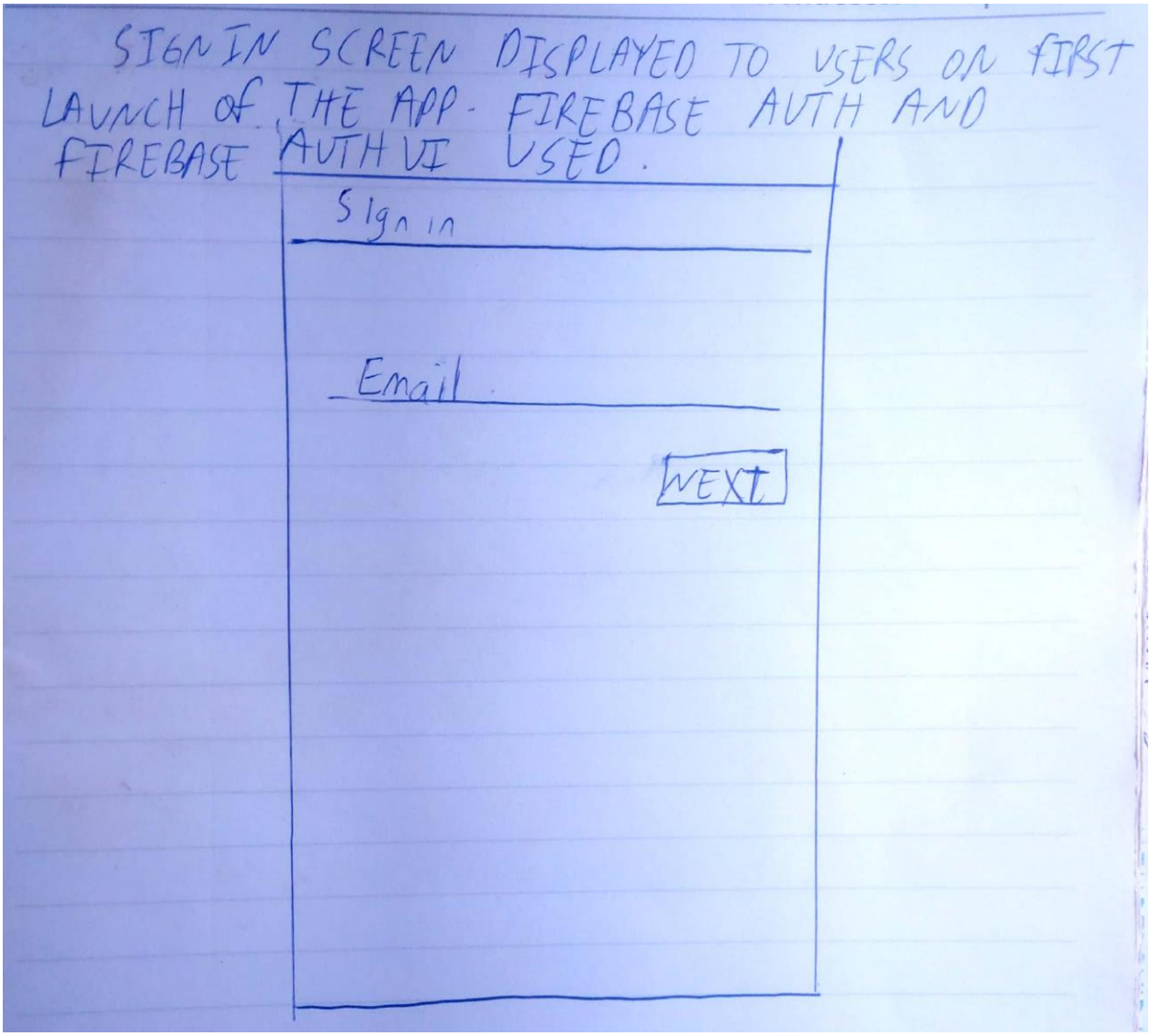


Main screen contains list of stored patients, FAB to add new patient, sort icon in toolbar to change sort order of stored patients (sorting could be by recency or alphabetical order), search icon which when selected, makes an edittext visible in the appbar making users to search for patients by name or hospital number.

Screen 2

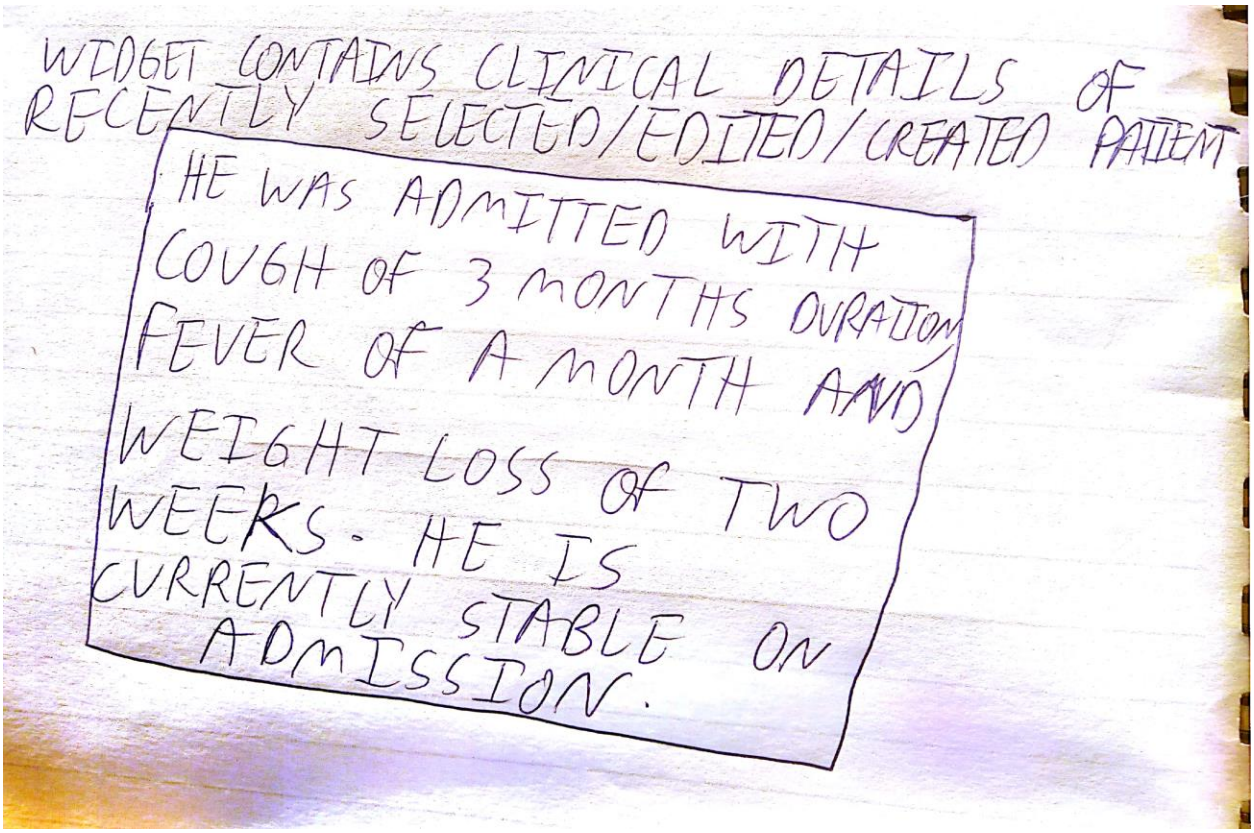
Add new/edit patient activity appears upon clicking the FAB or clicking any stored patient card. Save button is to save the patient details in the database and would show a dialog if vital patient details are missing





Screen 3

Sign in screen on first launch of the app in which users can sign in by email to get access to read/write from the firebase backend database



Screen 4

Widget displays clinical details of recently selected patient

Key Considerations

How will your app handle data persistence?

I would use room to store the data locally and would sync the data with firebase database using firebase jobdispatcher. Each Patient object would contain variables representing patients details. A user can add, update, delete a particular patient from the room database or read all patients stored in the database to be displayed on the main screen

Describe any edge or corner cases in the UX.

The app would basically be a three screen activity after the initial sign in. Parent activity contains the list of patients, the other two screens (add new or edit patient screen) will be hosted by the same activity and returns to parent activity on backpress.

Describe any libraries you'll be using and share your reasoning for including them.

I would be using butterknife library to bind views rather than calling findviewbyid.

Describe how you will implement Google Play Services or other external services.

I would use firebase realtime database to serve as a backend to store the patients details. I would also incorporate firebase authentication service to enable users sign in at app launch.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

- Configure libraries
- Setup firebase backend

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity which contains list of patients
- Build UI for AddNew/EditPatient Activity

Task 3: Setup room database

- Create Patient Pojo,
- Dao,
- room database
- Use livedata to query data from room and store in viewmodel so that it would survive configuration change
- Test adding, updating, querying and deleting patients inn the database on the emulator.

Task 4: Implement other UI functionalities and handle error cases

- Implement search for patient by name in main activity
- Implement sort functionality in main activity
- Prevent addition of incomplete patient object to room database

Task 5: Sync with firebase backend and setup authentication

- Implement firebasejobdispatcher to sync room database with firebase backend
- Implement firebase authentication

Task 6: Implement app widget

- Implement app widget to display recently selected patient's clinical details