Training Project Laboratory Report 2

Jena Woodroffe
Neptun: FNA1A3

Physiotherapy App

Description of the project:

This Android-based application aims to give physiotherapists a single platform from which to run their practice from. The final project will comprise of two applications - one exclusively for doctors and one for their clients, however for this semester's work, I will focus on the doctors' application. The app will contain the following features:

- ability to track appointments, create new appointments and notify the respective clients, display the upcoming appointments to the doctor
- tracking all information related to clients
    - their medical notes made by the doctor during appointments
    - their personal logs that they can make and update any time between appointments
    - text messages - the portal will have a chat portal where clients and physios can talk and sending photos and videos
    - their current exercise regime, the rehabilitation exercises they have been prescribed for the week along with their description, sets and repetitions

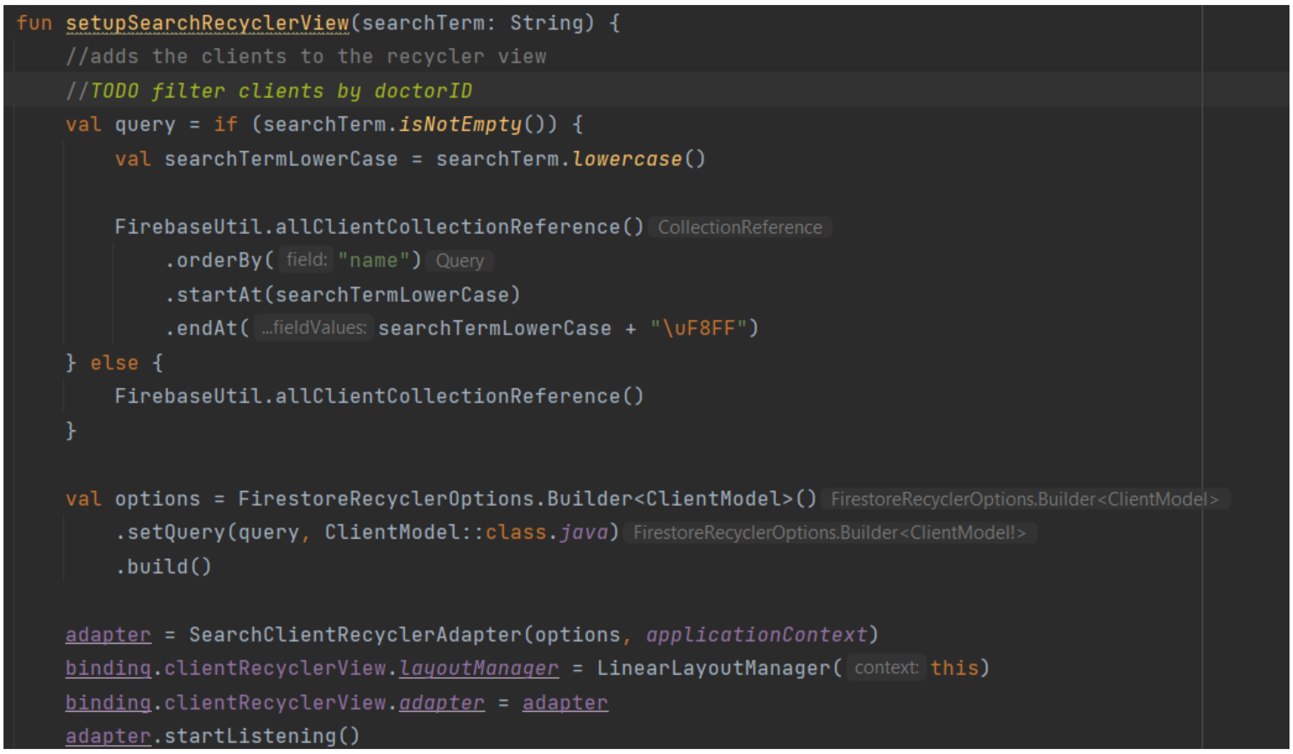
Tasks required for the week:

- ○ add clients for the test doctor into the Firestore
- ○ Create the contacts/ clients list for the app

Tasks done this week:

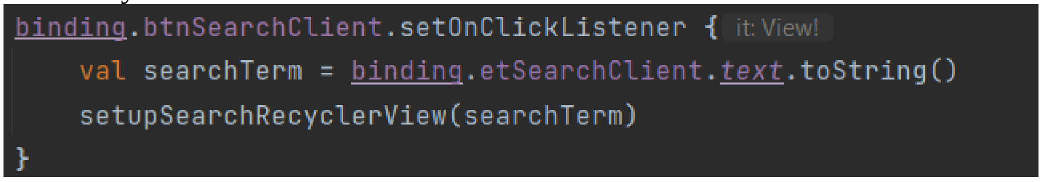
- ✓ Fixed the customized properties for edit text boxes to suit the aesthetic of the application
- ✓ Successfully implemented the registration of doctors through the registration screens
- ✓ Successfully used authentication to log in doctors
- ✓ Linked the firestore for clients to the project
- ✓ Created clients views to display client names and number
- ✓ Successfully implemented a search bar that queries the collection of clients depending on the text entered by the user l

Code Screenshots:



```
fun setupSearchRecyclerView(searchTerm: String) {
    //adds the clients to the recycler view
    //TODO filter clients by doctorID
    val query = if (searchTerm.isNotEmpty()) {
        val searchTermLowerCase = searchTerm.lowercase()

        FirebaseUtil.allClientCollectionReference() CollectionReference
            .orderBy( field: "name") Query
            .startAt(searchTermLowerCase)
            .endAt( ...fieldValues: searchTermLowerCase + "\uF8FF")
    } else {
        FirebaseUtil.allClientCollectionReference()
    }

    val options = FirestoreRecyclerOptions.Builder<ClientModel>() FirestoreRecyclerOptions.Builder<ClientModel>
        .setQuery(query, ClientModel::class.java) FirestoreRecyclerOptions.Builder<ClientModel!>
        .build()

    adapter = SearchClientRecyclerAdapter(options, applicationContext)
    binding.clientRecyclerView.layoutManager = LinearLayoutManager( context: this)
    binding.clientRecyclerView.adapter = adapter
    adapter.startListening()
```

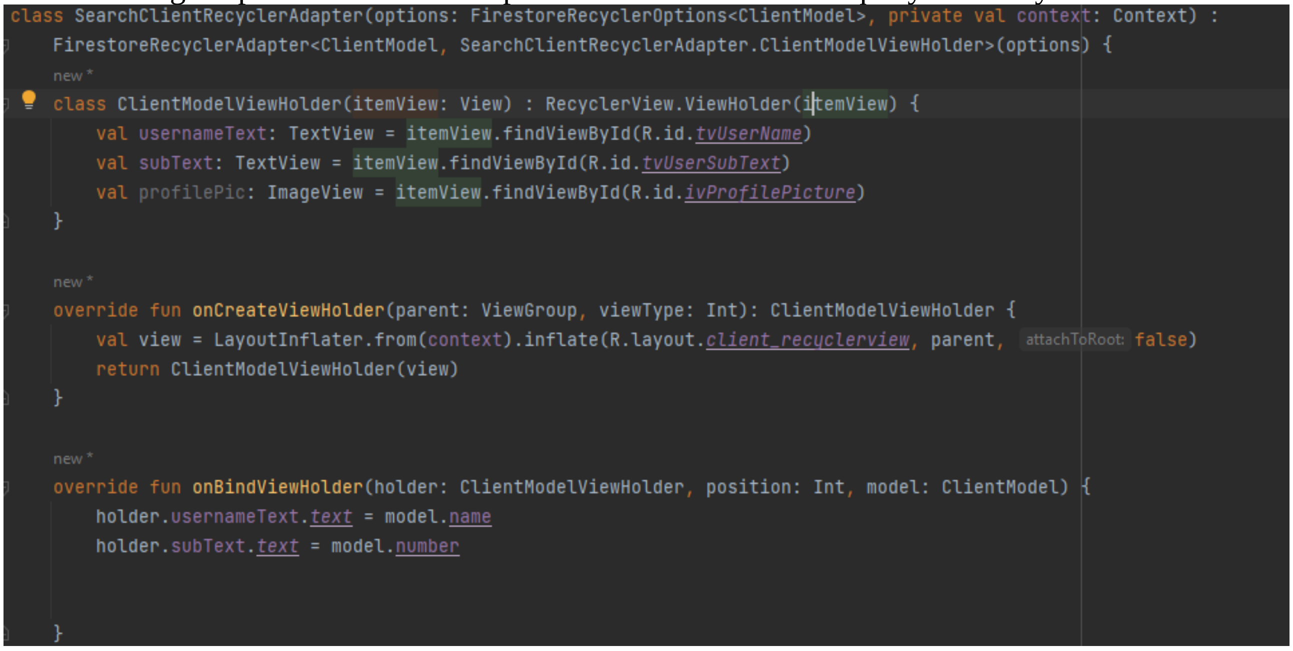
This method is present in the Clients activity – where the doctor can view all of their clients and eventually will be able to click on each user to read all of their information.



```
binding.btnSearchClient.setOnClickListener { it: View!
    val searchTerm = binding.etSearchClient.text.toString()
    setupSearchRecyclerView(searchTerm)
}
```

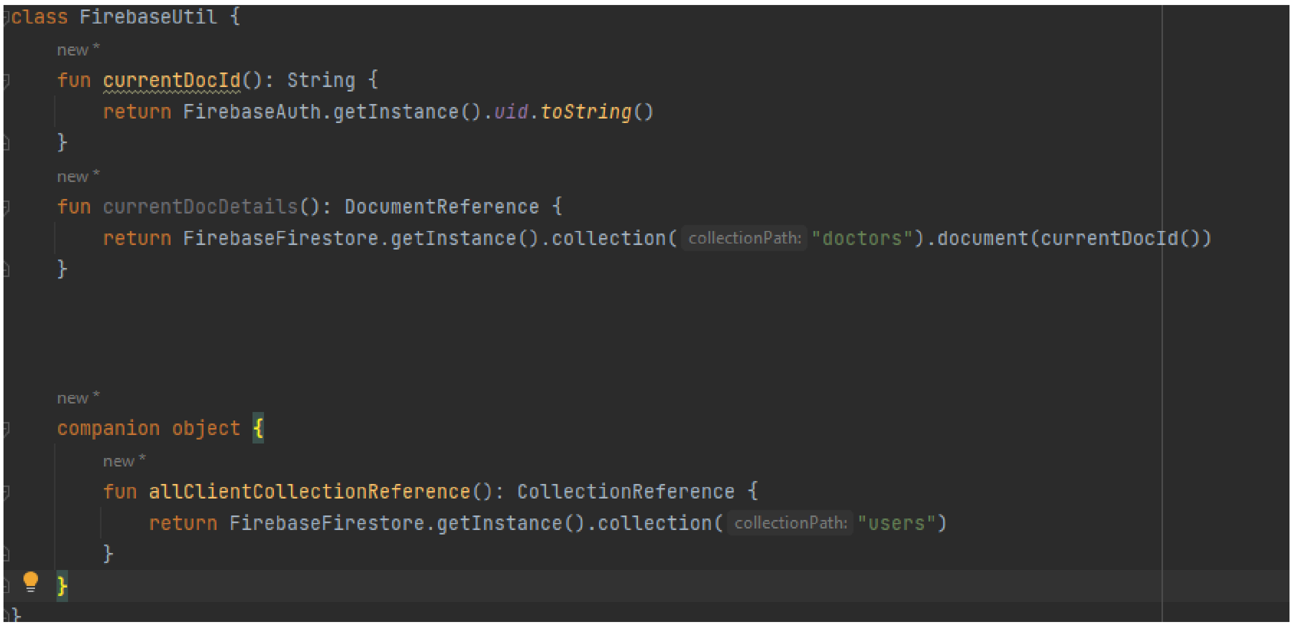
Taking the entered search term and passing it to the former method

The following adapter was made to map the information from the query to the layout



```
class SearchClientRecyclerAdapter(options: FirestoreRecyclerOptions<ClientModel>, private val context: Context) :
    FirestoreRecyclerAdapter<ClientModel, SearchClientRecyclerAdapter.ClientModelViewHolder>(options) {
    new *
    class ClientModelViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val usernameText: TextView = itemView.findViewById(R.id.tvUserName)
        val subText: TextView = itemView.findViewById(R.id.tvUserSubText)
        val profilePic: ImageView = itemView.findViewById(R.id.ivProfilePicture)
    }

    new *
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ClientModelViewHolder {
        val view = LayoutInflater.from(context).inflate(R.layout.client_recyclerview, parent,  attachToRoot: false)
        return ClientModelViewHolder(view)
    }

    new *
    override fun onBindViewHolder(holder: ClientModelViewHolder, position: Int, model: ClientModel) {
        holder.usernameText.text = model.name
        holder.subText.text = model.number


    }
```

I started working on a Util class that will hold methods for engaging with the FireStore. So far it contains:

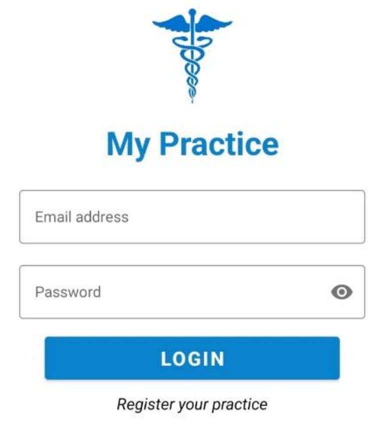


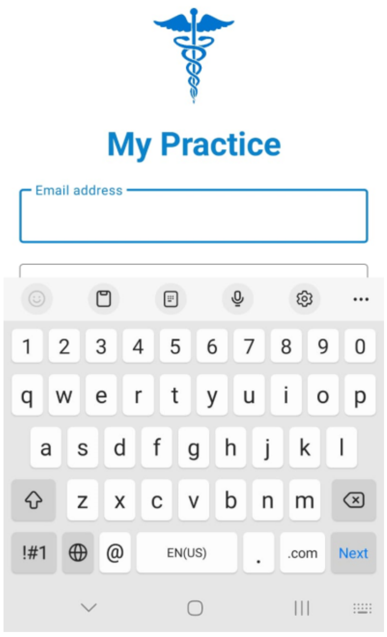
```
class FirebaseUtil {
    new *
    fun currentDocId(): String {
        return FirebaseAuth.getInstance().uid.toString()
    }
    new *
    fun currentDocDetails(): DocumentReference {
        return FirebaseFirestore.getInstance().collection( collectionPath: "doctors").document(currentDocId())
    }



    new *
    companion object {
        new *
        fun allClientCollectionReference(): CollectionReference {
            return FirebaseFirestore.getInstance().collection( collectionPath: "users")
        }
    }
}
```

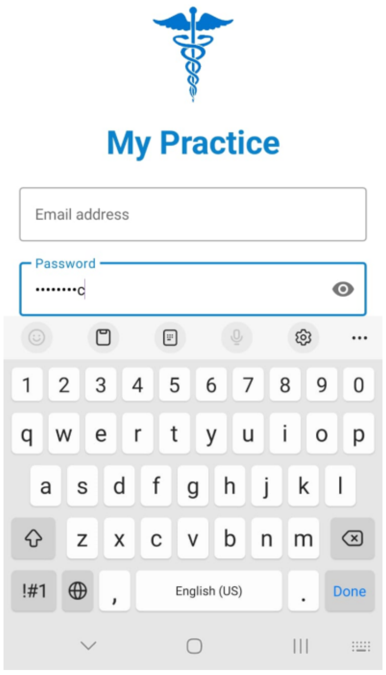
- The allClientCollectionReference is used in the Client activity to return all of the clients registered to the fireStore

Results Screenshots:

[redacted] [redacted] [redacted]

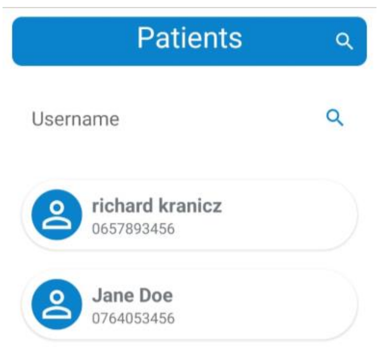


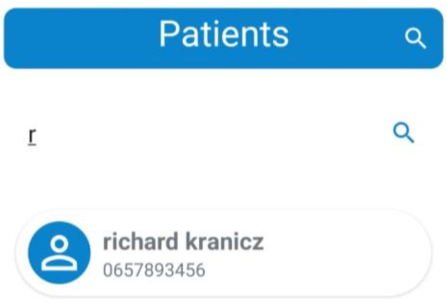




The above three screenshots demonstrate the improvement in the Edit text’s layout

-
- I managed to get the hint to move into the correct position
- Made the edit text grey when out of focus and the app’s main blue colour when in focus
- Ensured the password field hid the text





The screenshots on the left show the patients activity in the main application. Upon opening the activity, all of the clients are shown that currently exist in the client collection in Firestore. After I type r and press the search button, only the users matching the search term will be displayed, as can be seen in the second screenshot

Tasks to be done next week:

- Add intents to the registration activities so that if the user needs to move back and forth between the screens, the data is retained
- Add code so that the search bar for clients only appears when the user clicks the search button, and will disappear when the user clicks the X button
- Ensure that the bottom navigation bar that is present throughout the main application does not stay with the keyboard when an edit text is brought into focus

Notes (optional) :