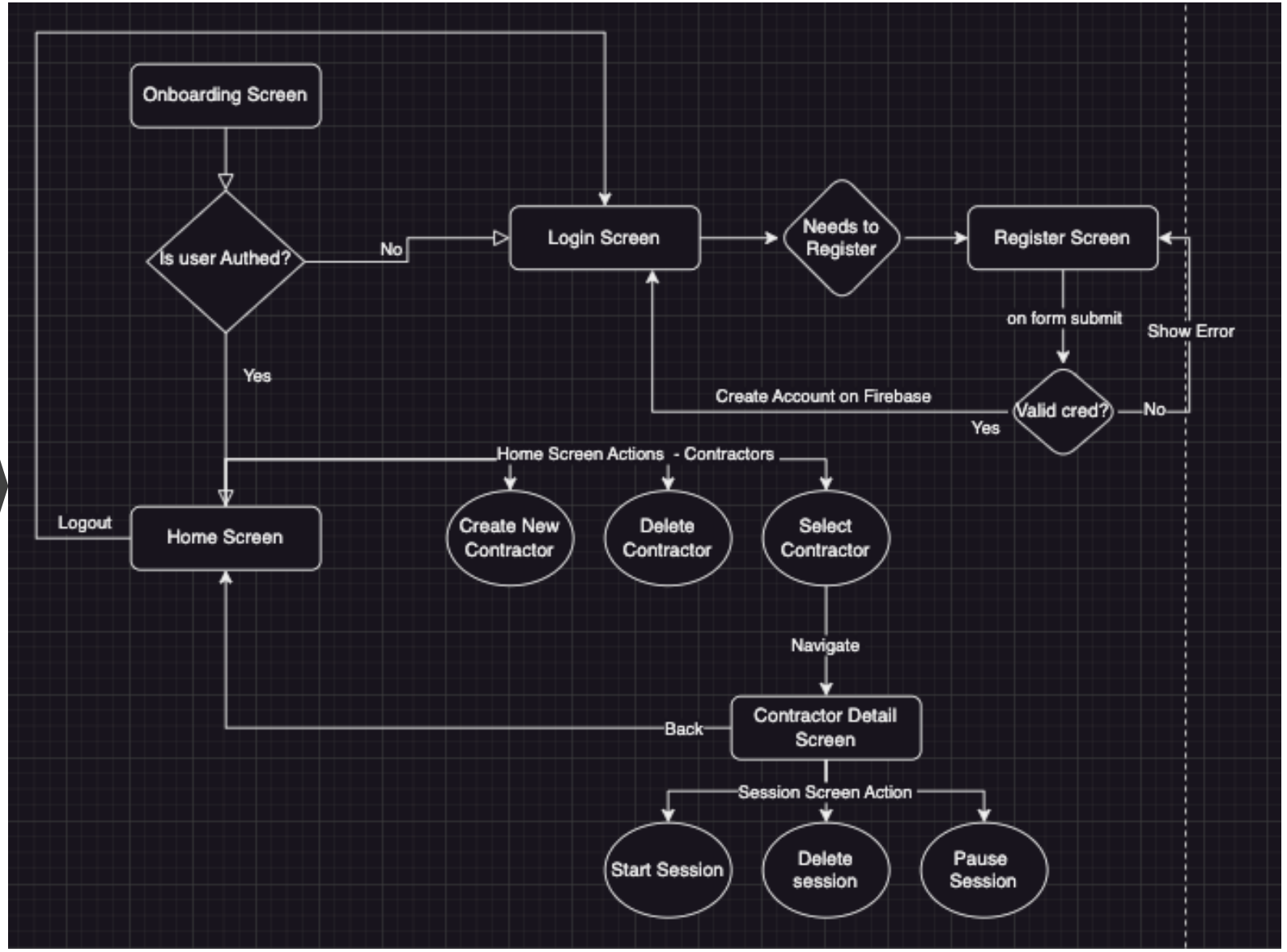
Plannor App

- CMP309 – Android Mobile Application | Moustapha Diaby
- 2001890

What Does the App Solve?

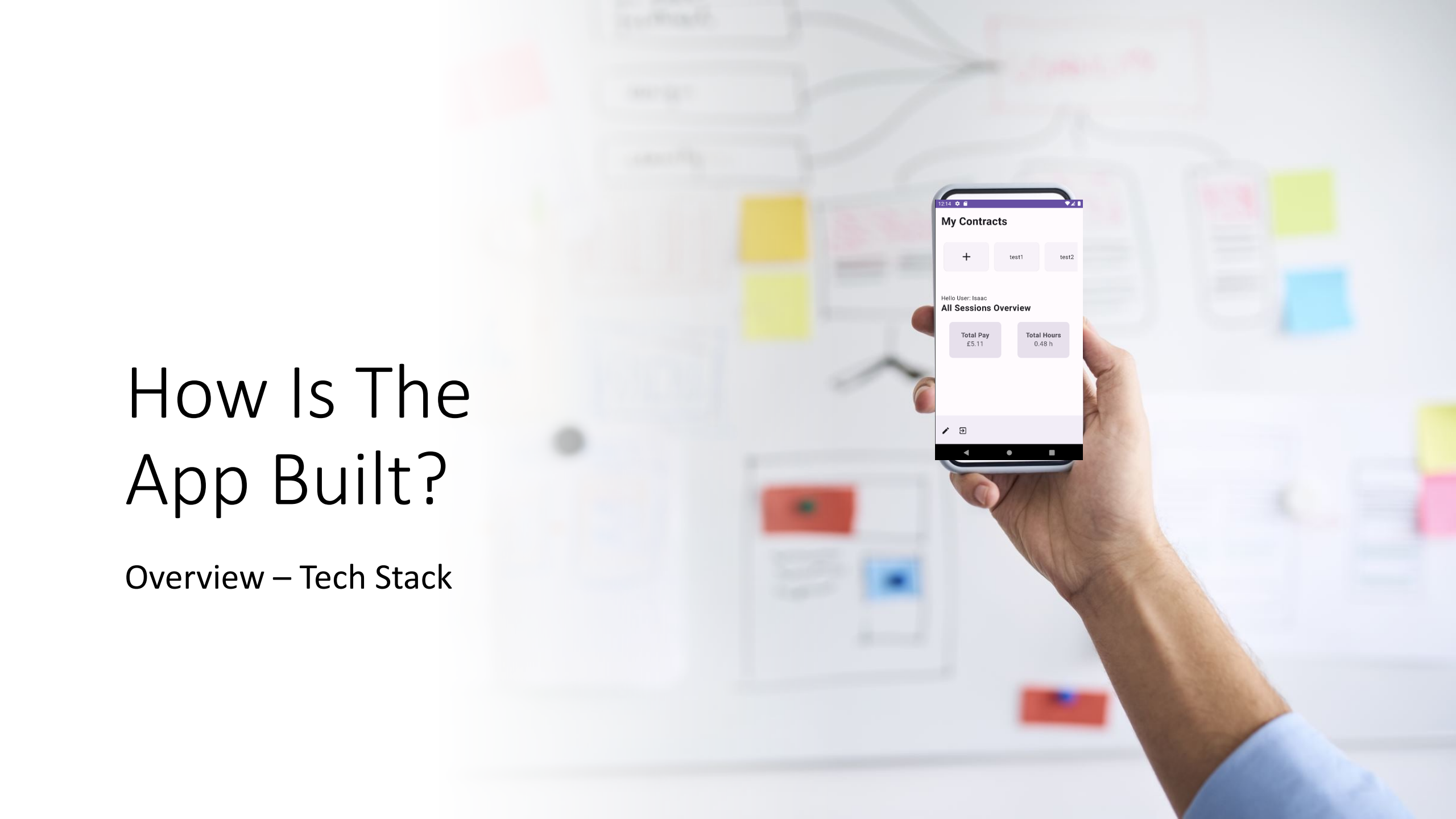
- As a contractor, I want an easy way to track and manage the times that I work for each of my contractors.
- I want to be able to easily track the hours worked and my expected pay.

App Screen Flow and Actions



How Is The App Built?

Overview – Tech Stack



Fundamentals

- Kotlin (personal preference)
- Firebase – Authentication, Firestore (Non-Relational Database)
  - Compatibility and Scalability
  - Uses internet access permission – INTERNET, ACCESS_NETWORK_STATE
  - Authentication
    - allowed the user to log in with email, password
  - Firestore
    - Account information
    - Contactor
    - Sessions

Jetpack Compose

- Jetpack Compose – UI Component and Design Library
  - Used Material Design and best UI Design best practices
  - Allows me to quickly develop UI
  - Navigation
- ViewModel - State Management - Stores UI State + Business Logic
  - UserViewModel
  - ContractsViewModel
  - ContractSessionViewModel

Notification and Services

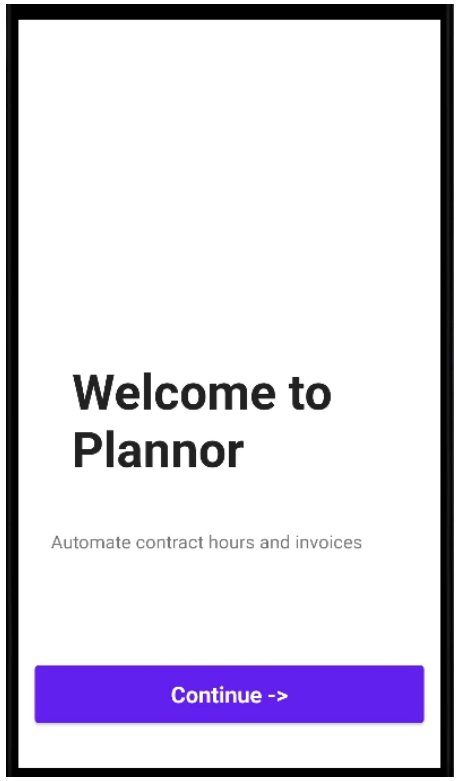
- Notifications
  - When the timer is started the user can close the app and still be able to see the timer counting up.
- Bound Service
  - Manages the state stopwatch and the notification
  - Timer Bindings to the UI and the notification
  - Doesn’t perform any database called as that is why we have the view model

How Is The App Built? Part 2

Demo + Code

Onboarding Screen

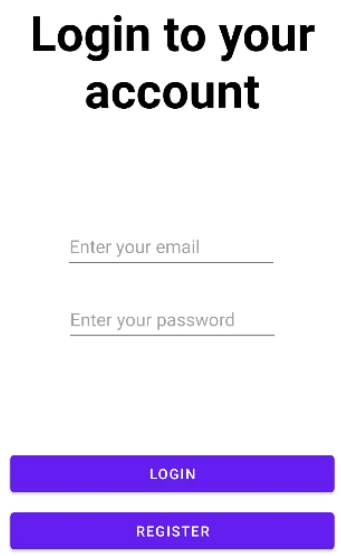
- Uses XML with the **ConstraintLayout** - **onboarding_main.xml**
- Continue button goes to the **login screen** **OnboardingActivity.kt:40**
- Uses an animated **TextSwitcher** to automatically Scroll over the onboarding messages. **OnboardingActivity.kt:60**
- If the user is already authenticated, they will navigate directly to the main activity




```
50      val txtSwitcher = binding.subtextctx
51      txtSwitcher.setInAnimation(applicationContext, android.R.anim.slide_
52      txtSwitcher.setOutAnimation(applicationContext, android.R.anim.slide
53      val textToShow = arrayOf(
54          "Plannor help contractors and companies manage their con...",
55          "Automate contract hours and invoices",
56          "Plan your work week for every project"
57      )
58      txtSwitcher.setText(textToShow[currentIndex])
59      // loop through the text until the user clicks next with a 3 second
60      loopDescCtr.scheduleAtFixedRate(object : TimerTask() {
61          override fun run() {
62              Log.d( tag: "OnboardingActivity",  msg: "onCreate: $currentInde
63              currentIndex++
64              // If index reaches maximum reset it
65              if (currentIndex == (textToShow.size)) {
66                  currentIndex = 0
67              }
68              runOnUiThread { txtSwitcher.setText(textToShow[currentIndex]
69          }
70      }, delay: 3000,  period: 3000)
```

Registration Screen

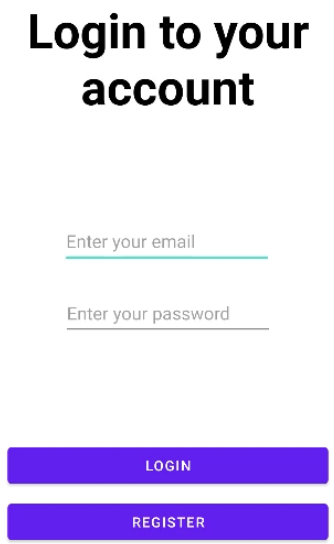
- Before a user can log in they must register an account
- We Check that the input is valid!
  - **AuthHelper.kt:5**
- This is saved by using FireStore.
  - **SignUpActivity.kt:73**




```
73  private fun signUp(name: String, email: String, password: String, contractor: Boolean) {
74      auth.createUserWithEmailAndPassword(email, password)
75          .addOnCompleteListener(this) { task ->
76              if (task.isSuccessful) {
77                  // Sign in success, update UI with the signed-in user's information
78                  Log.d( tag: "SignUpActivity",  msg: "createUserWithEmail:success")
79                  val user = auth.currentUser
80                  // add user to database with their name, email, and contractor status
81                  val newUserData = hashMapOf(
82                      "name" to name,
83                      "contractor" to contractor,
84                      "email" to email,
85                      "uid" to user?.uid
86                  )
87                  db.collection( collectionPath: "accounts").document(newUserData.get("uid") as String)
88                      .set(newUserData) Task<Void!>
89                      .addOnSuccessListener { documentReference ->
90                          Log.d(
91                              tag: "SignUpActivity",
92                              msg: "DocumentSnapshot added with ID: ${documentReference.toString()}"
93                          )
94                          // go to home Authed screen
95                          val intent = Intent( packageContext: this, MainActivity::class.java)
96                          startActivity(intent)
97                          finish()
```

Login Screen

- When a user has an account registered they can login with their email and password
  - **LoginActivity.kt:52**
- We also perform user input checks (valid email and password) before allowing the user to go to the

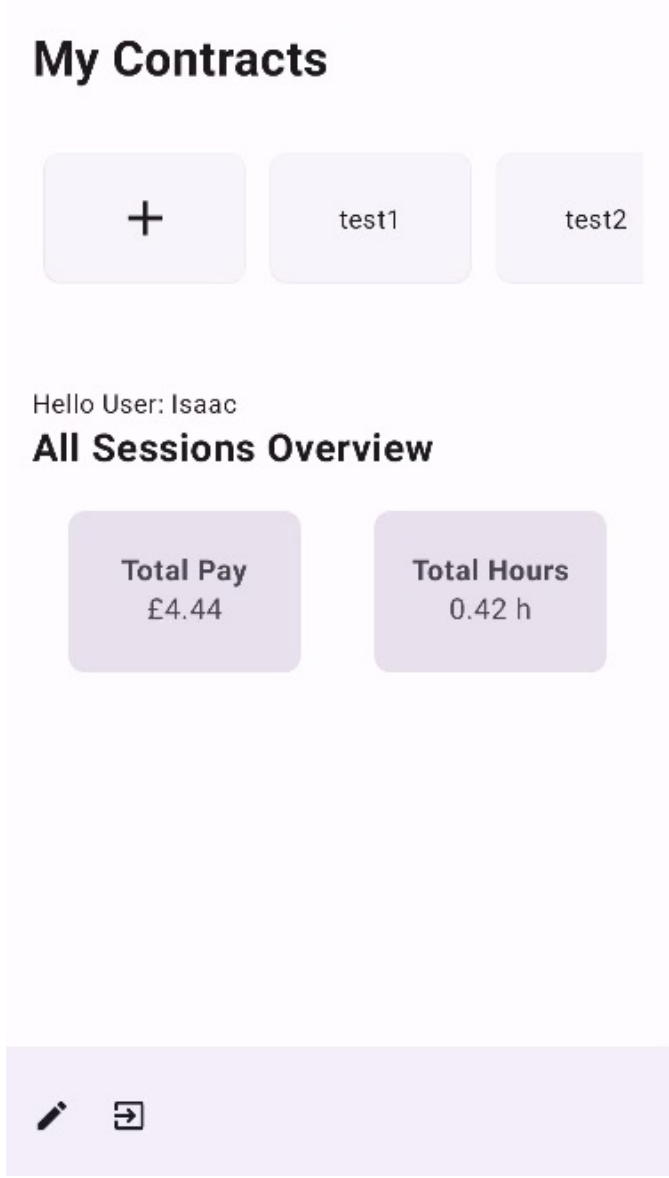



```
52      private fun login(email: String, password: String) {
53          auth.signInWithEmailAndPassword(email, password)
54              .addOnCompleteListener(this) { task ->
55                  if (task.isSuccessful) {
56                      // Sign in success, update UI with the signed-in user's information
57                      Log.d( tag: "LoginActivity",  msg: "signInWithEmail:success")
58                      val user = auth.currentUser
59                      updateUI(user)
60                  } else {
61                      // If sign in fails, display a message to the user.
62
63                      Log.w( tag: "LoginActivity",  msg: "signInWithEmail:failure", task.exception)
64                      Toast.makeText(baseContext,  text: "Authentication failed.",
65                          Toast.LENGTH_SHORT).show()
66                      updateUI( user: null)
67                  }
68              }
69      }
```

Home Screen

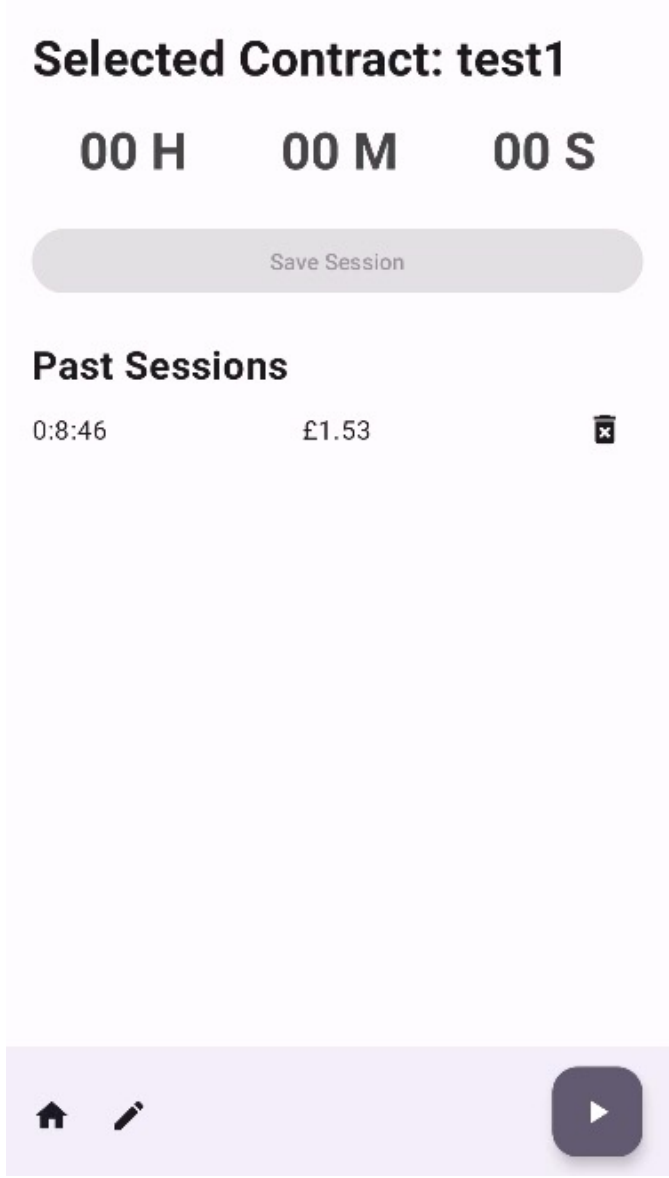
- Used Jetpack Compose
  - HomeScreen.kt
- Uses The Compose Navigator
  - navRouter.kt
  - Which houses all of the view models and shares the same instance with all routed Screens

- What can you Do on this Screen?
  - Create a new Contractor
  - Select a Contractor to navigate to the Contractor detail Screen
  - Enter edit mode and Delete a contractor
  - Log out



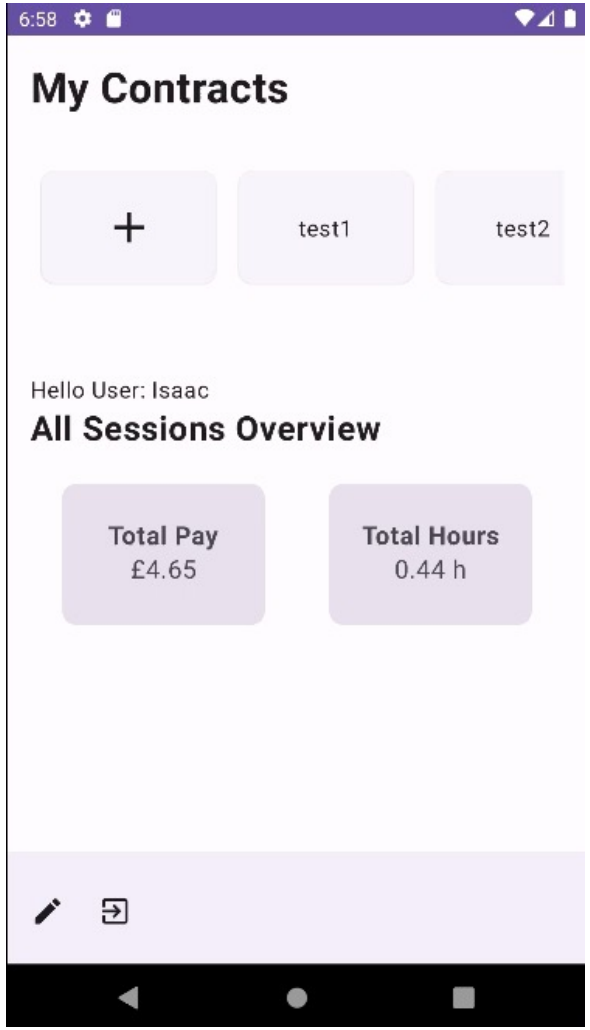
Contractor detail Screen

- Used Jetpack Compose
  - ContractorScreen.kt
- Uses The Compose Navigator
  - navRouter.kt
- What can you Do on this Screen?
  - Create a new Contractor session
  - Delete a contractor session
  - Navigate back to the home screen



Notification and Services

- Make use of a bound Service which allows us to tie the service to an activity and fully manage its instance.
  - Was used to create the notification, edit the state with intent actions and stream the timer information bidirectional to the notification and the application
  - Allowed us to keep the timer running even when the application is closed or in the background.



Future Work

- Authentication
  - Other Auth Providers – Google, Facebook, Github
    - Firebase Handles them but needs to integrate and enable it.
- Contractor
  - Edit The Hourly Rate – right now it's fixed to £10.50
  - Share Your Sessions with Other Contractor Users
- Users
  - Be able to add other users and see when they are online with the Realtime database from Firebase.


Thanks For Listening

References

https://developer.android.com/codelabs/basic-android-kotlin-compose-viewmodel-and-state#4 – Basic Android Kotlin Compose ViewModel and State, by Google, accessed 2 May 2023

https://developer.android.com/guide/components/bound-services - Bound Services, , by Google, accessed 8 May 2023

https://youtu.be/4gUeyNkGE3g - Jetpack Compose Navigation for Beginners - Android Studio Tutorial Youtube video, by Philipp Lackner, accessed 3 May 2023

https://youtu.be/goFpG25uoc8 - Type safe, multi-module best practices with Navigation Compose Youtube video, by Google, accessed 5 May 2023

https://developer.android.com/guide/navigation/navigation-type-safety - navigation type safety Youtube video, by Google, accessed 5 May 2023

https://github.com/stevdza-san/Stopwatch-Android-App - reference for stopwatch lib, by stevdza-san on GitHub, accessed 8 May 2023

https://developer.android.com/training/dependency-injection/hilt-android - Dependency injection with Hilt, , by Google, accessed 10 May 2023