Time Box 7 – Login Fragment

Oversigt

OpgaveNavn	Implementering af login fragment		
Implementering af krav	Implementering af kravene APP-1, APP-2 og APP-3		
Udført af	Marc	**Dato**	05-11-2021
Timebox	7	**Område**	Mobilapp

Contents

Introduktion

Dette dokument beskriver implementeringen af mobilappens login funktionalitet, hvordan den er opbygget, hvad funktionerne gør og hvilke bruger elementer der er taget med.

Analyse

Der skal være to edit text's:

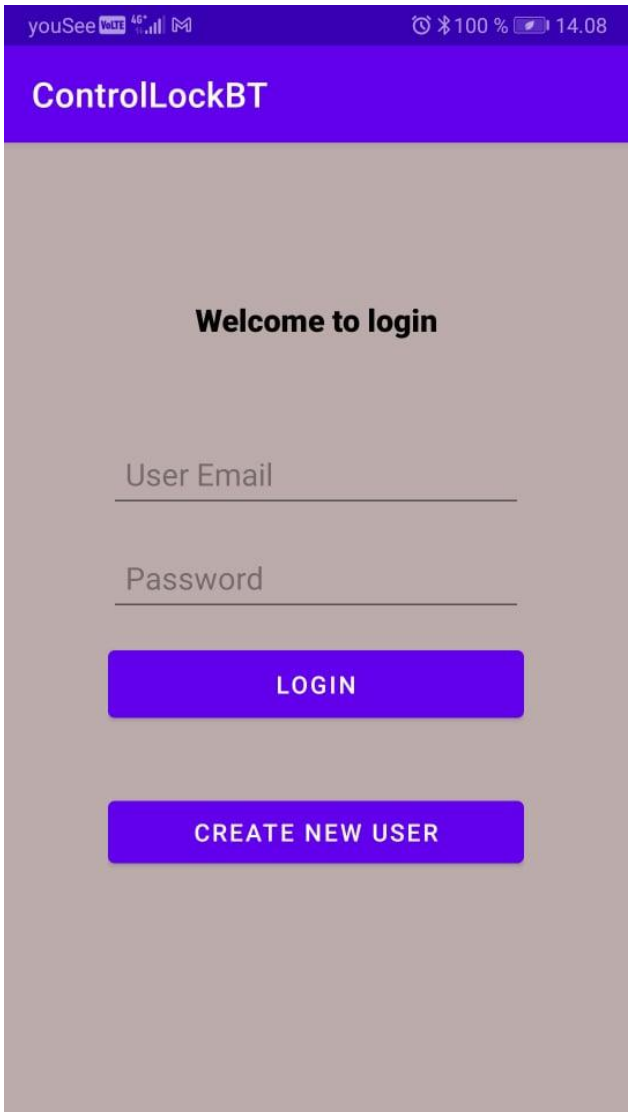
- Username, som er brugerens email
- Password, som er brugerens kodeord

Der skal være to knapper:

- Log in, som skal sende informationerne: user email og password til rest api for at tjekke om bruger eksistere, hvis ja skal user email og token fra rest api sendes med til næste fragment.
- Create new user, som sender brugeren til hjemmesiden "www.control-center.xyz/create_user/"

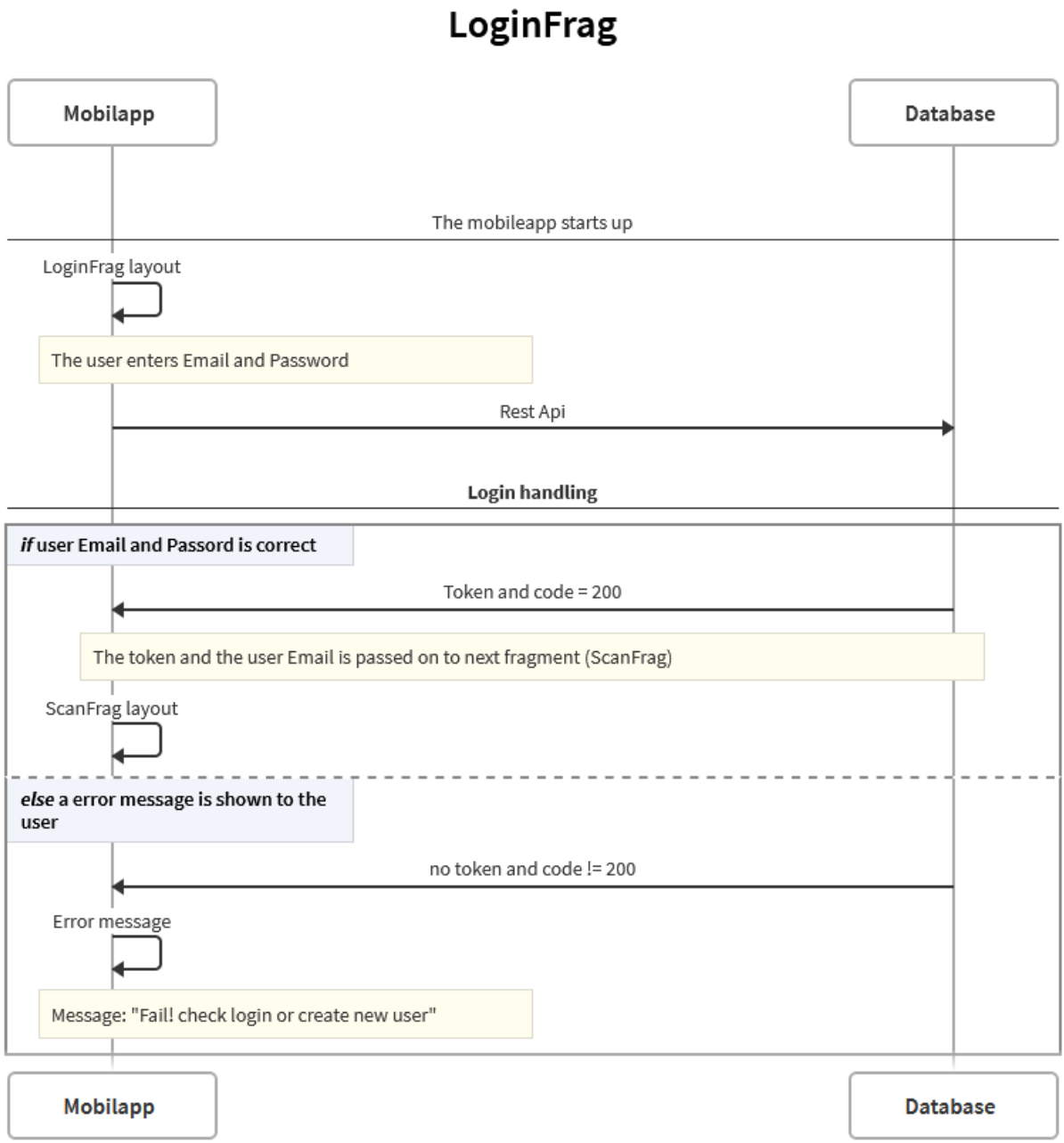
Design

Layout for LoginFrag er som på Figur 1, hvor der er en velkomsttekst, to felter, en til brugerens email og en til kodeord, og to knapper en til Login og en til create new user.



Figur 1 Login viewet

Fragmentets forløb ser ud således som på Figur 2



Figur 2 Swimlane for LoginFrag

Implementering

Implementeringen af "Create New User" knappen er som på Figur 3, hvor først findes viewet (dette gøres altid når der skal interageres med layoutet), når der trykkes på knappen, får brugeren en "Toast" besked og funktionen "openWebPage" aktiveres.

```
50          // Find button for create user
51          val btnCreateUser = view.findViewById<Button>(R.id.btnCreateUser);
52          // Create new user
53          btnCreateUser.setOnClickListener {  it: View!
54              showToast( msg: "Redirect to Create User URL")
55              // Redirect to Create new user page in browser
56              openWebPage( url: "http://www.control-center.xyz/create_user/")
57          }
```

Figur 3 Create User button

Funktionen openWebPage er som på Figur 4, som sender brugeren hen til "www.control-center.xyz/create_user/".

```
31      //  Function for opening web page
32      fun openWebPage(url: String) {
33          val webpage: Uri = Uri.parse(url)
34          val intent = Intent(Intent.ACTION_VIEW, webpage)
35          startActivity(intent)
36      }
```

Figur 4 Funktionen til at åbne en webpage

Knappen "Login" er som på Figur 5 og Figur 6, hvor Figur 5 henter de indtastede informationer fra brugeren, sætter rest api op og pakker beskeden der skal sendes til databasen.

```
61          // Login
62          btnLogin.setOnClickListener {  it: View!
63              // Get inputs
64              val viewEmail = view.findViewById<EditText>(R.id.editTextTextEmailAddress)
65              val viewPassword = view.findViewById<EditText>(R.id.editTextTextPassword)
66              // Login message
67              showToast( msg: "Login Clicked")
68              //retrofit repo
69              val repository = Repository()
70              //retrofit modelFac
71              val viewModelFactory = LoginFragModelFactory(repository)
72              //model extension
73              viewModel = ViewModelProvider( owner: this, viewModelFactory).get(LoginFragViewModel::class.java)
74              val myPost = PostLogin(viewEmail.text.toString(), viewPassword.text.toString(), auth_token = "")
75              //push POST to restAPI
76              viewModel.pushPost(myPost)
77              // Error message view
78              val errormsg = view.findViewById<TextView>(R.id.errorText)
```

Figur 5 Login button del 1

På Figur 6 sættes en ”observer” op som lytter på responset fra Rest api og tjekker efter koden, hvis koden er 200 har brugeren logget ind korrekt.

```
79          // Check response
80          viewModel.myResponse.observe(viewLifecycleOwner, { response ->
81              // check if login successful
82              if(response.code() == 200) {
83
84                  // save token for onDestroy
85                  tokenmodel.setToken(response.body()?.auth_token.toString())
86                  val test = tokenmodel.getToken()
87                  Log.d( tag: "ViewModel", test.toString())
88                  // Create bundle
89                  val bundle = Bundle()
90                  bundle.putString("UserEmail", viewEmail.text.toString())
91                  bundle.putString("Token", response.body()?.auth_token.toString())
92                  // Redirect to ScanFrag
93                  Navigation.findNavController(view).navigate(R.id.action_loginFrag_to_scanFrag, bundle)
94              }
95              else{
96                  // Failed to login send error message
97                  errormsg.visibility = View.VISIBLE
98              }
99          })
100     }
```

Figur 6 Login button del 2

Til sidst i koden returneres view.

Verifikation

Det antages at man har:

1. Adgang til en bruger
2. Har nyeste app
3. Har startet appen

Tabel 1: Tests til verifikation af opgave

Test	Test Steps	Pass-betingelser	Resultat
Login korrekt	1. indtast bruger informationerne 2. klik på login	1. Man bliver sendt videre	Bestået
Login forkert	1. indtast forkerte bruger informationer 2. klik på login	1. Man bliver ikke sendt videre, men får beskeden ”Fail! check login or create new user”	Bestået

Konklusion

Det er nu muligt at logge ind på mobilapp med en eksisterende bruger, der er implementeret en håndtering af forkert indtastede information.