Griffith College

Repeat Assignment

BSCH-MD
Mobile Development

Valentin Gnidyi

3054010

Stage 4

Mobile Development Project

Android Based Mobile Application

“Student Support”

Documentation

The developed mobile application is a platform that helps students to connect between each other and support with study modules.
The developed mobile application shows to logged person in the system they current location and profiles of other students with details of what modules they are in studying progress at the moment, what not started (to possibly ask support in future), and programs that other students are finished together with location where they are comfortable to meet.

Main idea is that students are voluntarily giving contact details to have contact from other students that might need support with modules that have been completed.

For such a project was used Google Firestore as a database where information about users is stored. Alongside for user authentication was user Firebase Auth system. For having location of the current user, the GPS location sensor was used.

Functionalities of the developed system are:

- Login as a User Student
- Registration of User Student
- Creating students` profile
- See on the Map other student profiles

To developed such system handy tools are Class Diagram, Sequence Diagram and Use Case diagram , which helps developer to be focused and properly clarified what is need to be developed and how is better to implement.

The Use Case Diagram of developed project is:

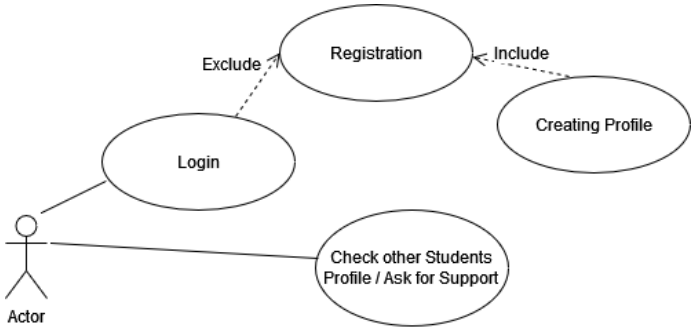

Figure 1: Student Support Application Use Case Diagram

Which gives explanation how user going to interact with the developed system. User as Student can login to system, if not have a profile – register in a system and automatically create a profile.

The Sequence Diagram helps to clarify how the information flows inside the system and to see how the system takes and process information.

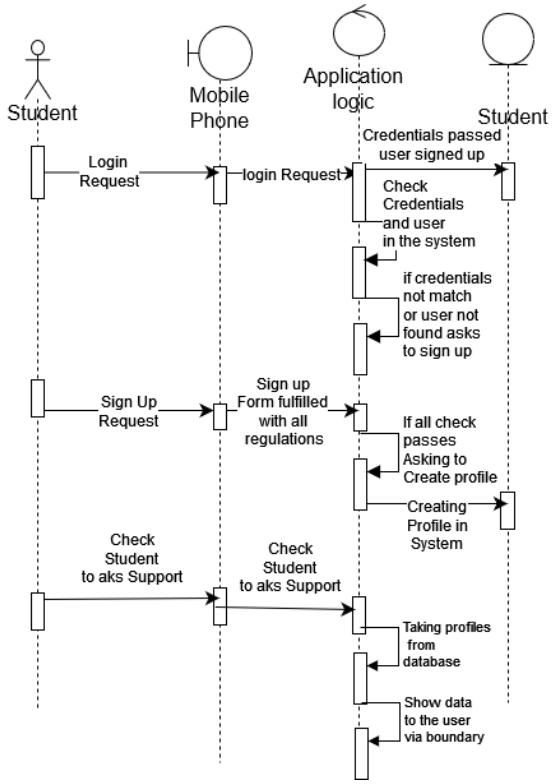

Figure 2: Student Support Sequence Diagram

Last but not least is Class Diagram, which identifies how structured classes and objects in developed system. There are only 2 classes such as Module and Student, with consideration that students can have as much modules as possible and many modules can be related to multiple students. All attributes of such classes and functionality that related to them are presented on Figure 3.

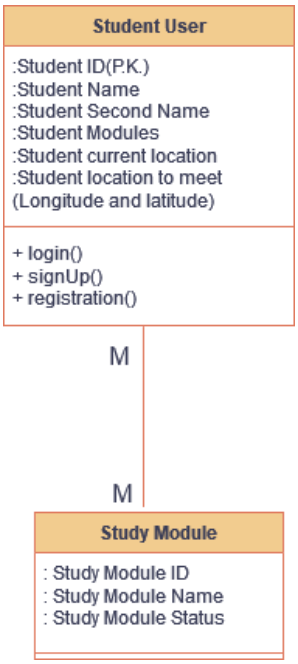

Figure 3. Student Support Class Diagram

The design of Student Support Mobile application is based on principals of being simple and intuitive understanding. There are 3 screens in the developed application: Student Login Screen,
The Wireframes of the mobile application alongside with actual screens are presented below.

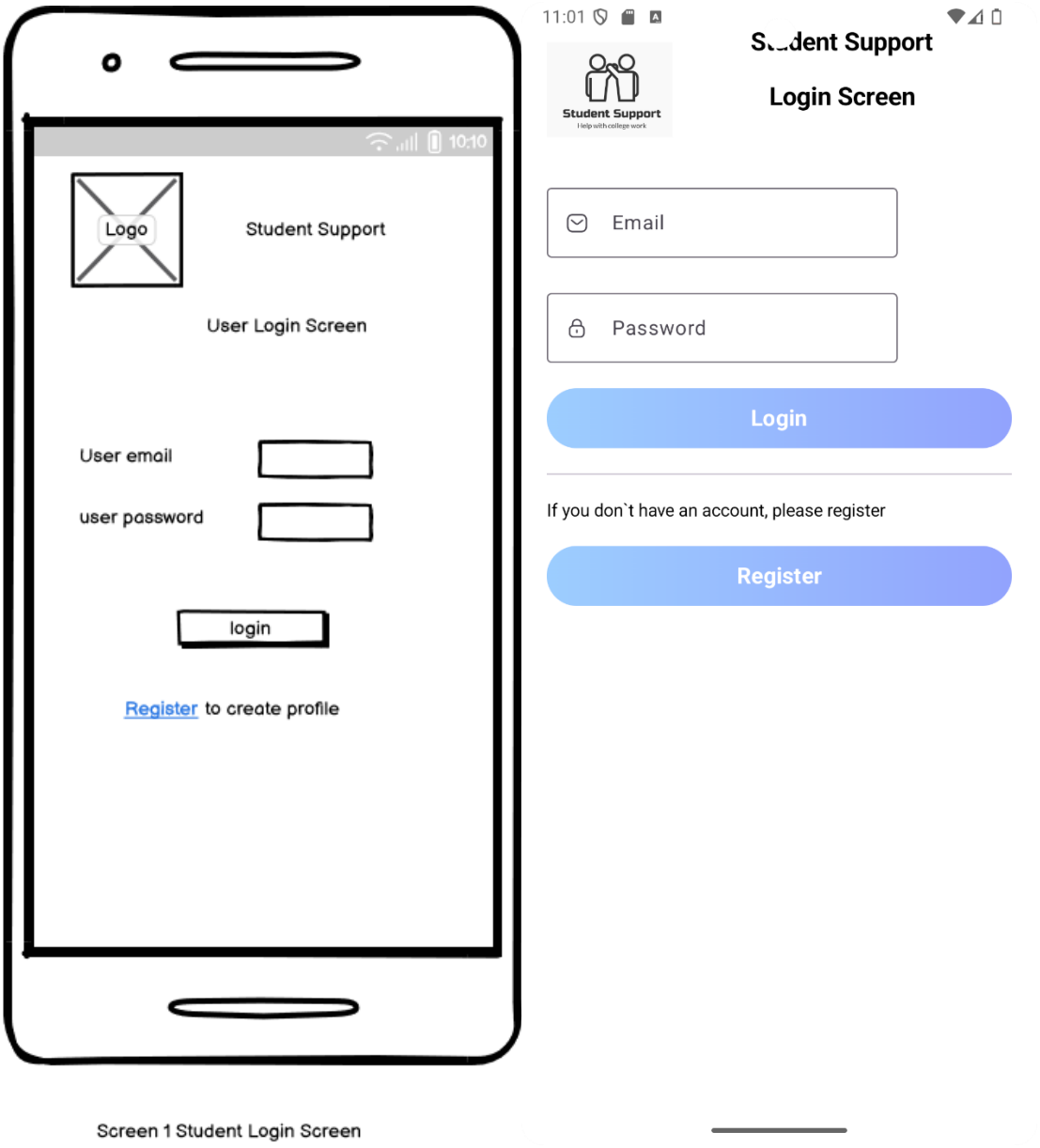

Figure 4: Student Support Screen Wireframe together with actual developed screen

Screen 5: Student create profile Wireframe and Actual Developped Screen

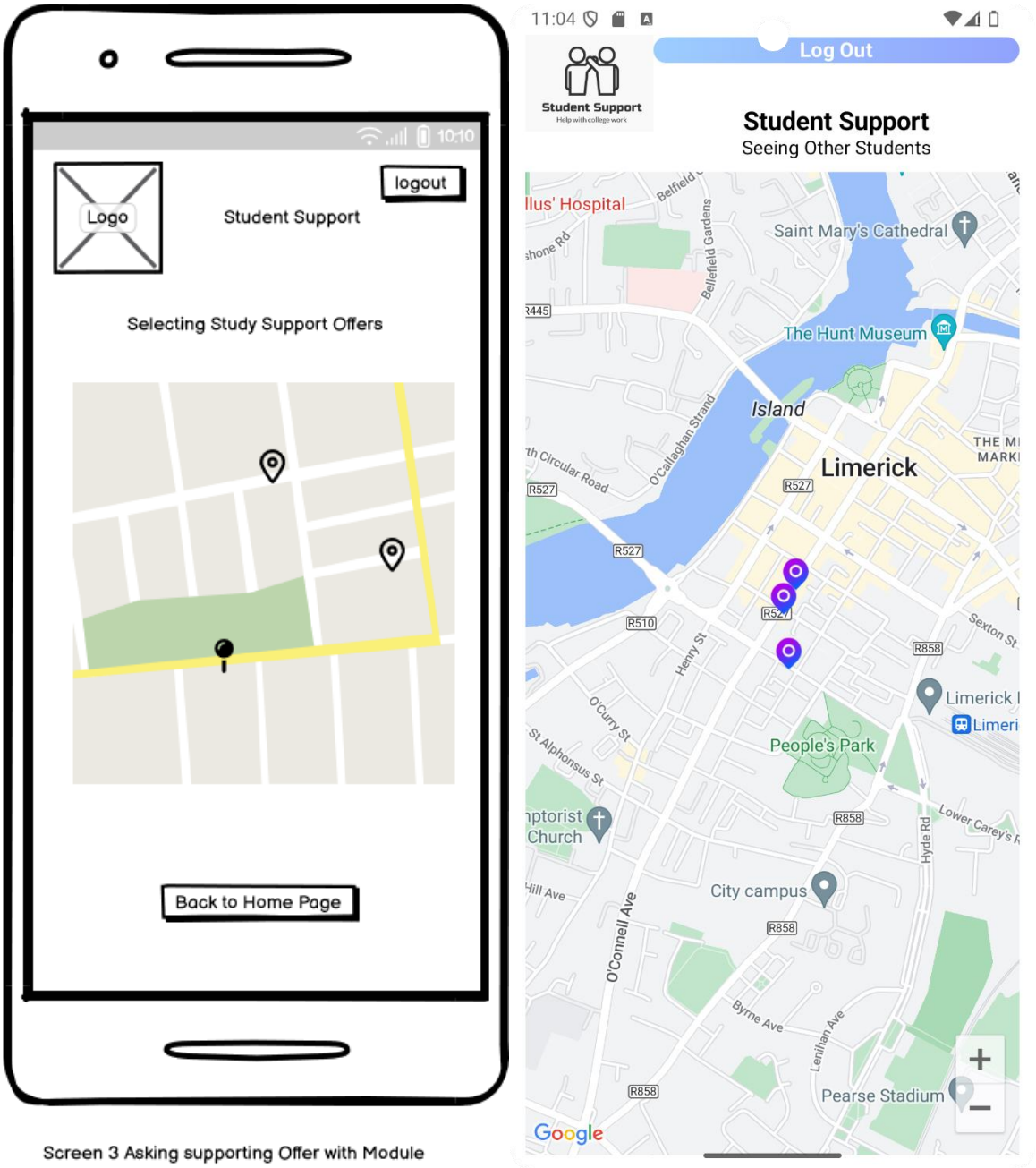

Figure 6: Wireframe and Actual Developed Screen of Seeing other Students and Asking for Support screen

In relation of the code was user MVVM architecture system, where all possible data and system logic are processed and being stored in the View Model of the mobile application.

The structure of the mobile app is represented on Figure 7. Where all related classes and files are separated via using proper development package structure.

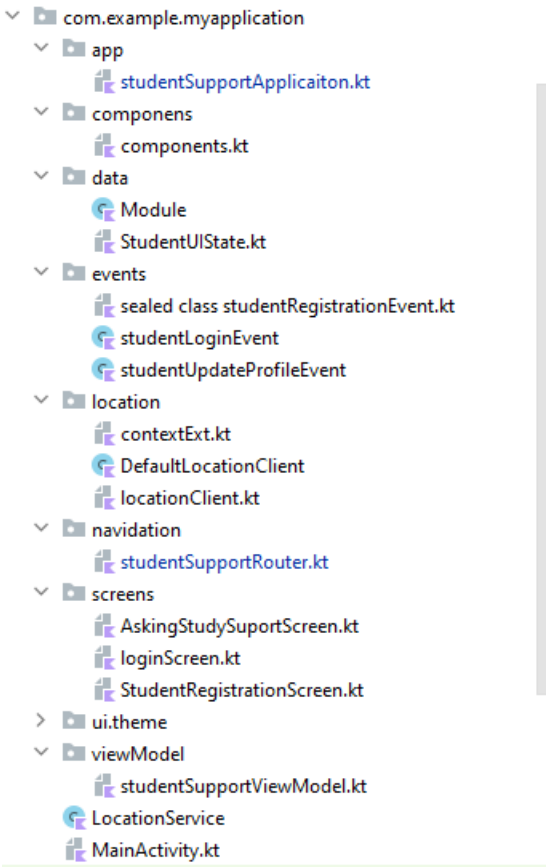

Figure 7: Code Development structure in Student Support Mobile Application

Developing Screens and Components was by using the Jetpack Compose framework. Snippets of code are given below.

```kotlin
@OptIn(ExperimentalMaterial3Api::class)  ≛ val1un41k
@Composable
fun MyTextFieldComponent(
    labelValue: String, painterResource: Painter,
    onTextChanged: (String) -> Unit,
    errorStatus: Boolean = false
){
    val textValue = remember{
        mutableStateOf( value: "")
    }
    val localFocusManager = LocalFocusManager.current

    OutlinedTextField(
        label = {Text(text = labelValue)},
        colors = TextFieldDefaults.outlinedTextFieldColors(
            focusedBorderColor = Primary,
            focusedLabelColor = Primary,
            cursorColor = Primary,
            ),
        keyboardOptions = KeyboardOptions(imeAction = ImeAction.Next),
        singleLine = true,
        maxLines = 1,
        value = textValue.value,
        onValueChange = {
            textValue.value = it
            onTextChanged(it)
        },
        leadingIcon = {
            Icon(painter = painterResource, contentDescription = "")
        },
```

Figure 8: Snippet of code related to the Custom Components elements.

```kotlin
@Composable  ≛ val1un41k+1 *
fun LoginScreen(studentSupportViewModel: StudentSupportViewModel = viewModel()){
    //take from Firebase list of modules
    LaunchedEffect(key1=true){
        studentSupportViewModel.getModules()
        studentSupportViewModel.takeDataForAllStudentsFromFirebase()
    }

    Surface(
        Modifier
            .fillMaxSize()
            .background(Color.White)
            .padding(20.dp)
    ){
        Column( modifier = Modifier.fillMaxSize()){

            HeadingTextComponentWithoutLogout(value = "Login Screen")
            Spacer(modifier = Modifier.height(20.dp))

            MyTextFieldComponent(labelValue = "Email",
                painterResource(R.drawable.message),
                onTextChanged = {
                studentSupportViewModel.loginEvent((studentLoginEvent.studentLoginEmailChanged(it)))
                })
            Spacer(modifier = Modifier.height(20.dp))

            MyTextFieldComponent(labelValue = "Password",
                painterResource(R.drawable.lock) ,
                onTextChanged = {
                    studentSupportViewModel.loginEvent(studentLoginEvent.studentLoginPasswordChanged(it))}
```

Figure 9: Snippet of Code Related to the User Login Screen.

The code that is provided by link on gitfub

https://github.com/val1un41k/StudentSupport.git
can be describe in a way that Student opening app creating an intent to use to collect location that will be used in next screen. The user need to accept the permission that will be asking to have an app working.

On Login Screen (Figure 9): User is able to insert the information in text fields of Login and Password. After it press login button – system checking in database such user and if founds – allows the access to the Asking Support screen where user can see other profiles. By pressing Register button – user will be sent to the Register / Create profile Screen.

On Register Profile Screen: User need to fulfill related form that can give information to the stystem for placing it in the Class Student into firebase. This can be proceeded by pressing Register button.

```
@Composable  val1un41k +1
fun StudentRegistrationScreen(studentSupportViewModel: StudentSupportViewModel = viewModel()) {

    Surface(
        modifier = Modifier
            .fillMaxSize()
            .background(Color.White)
            .padding(28.dp)
    ) {
        Column(modifier = Modifier.fillMaxSize()) {
            LazyColumn() {
                item {
                    HeadingTextComponentWithoutLogout(value = "Student Registration Screen")
                    Spacer(modifier = Modifier.height(20.dp))
                    Text(text = "First Name")
                    MyTextFieldComponent(labelValue = studentSupportViewModel.registrationUIState.value.studentName,
                        painterResource(R.drawable.message),
                        onTextChanged = {
                            studentSupportViewModel.onEvent(
                                studentRegistrationEvent.StudentNameChanged(
                                    it
                                )
                            )
                        })
                    Spacer(modifier = Modifier.height(20.dp))

                    Text(text = "Second Name")
                    MyTextFieldComponent(labelValue = studentSupportViewModel.registrationUIState.value.studentSurname,
                        painterResource(R.drawable.message),
                        onTextChanged = {
                            studentSupportViewModel.onEvent(
                                studentRegistrationEvent.StudentSurnameChanged(
```

Figure 10: Snippet of code from Student Registration screeen.
the Event ementes are responsible to stnaport the data from the screen to the ViewModel for keep then and after create the Class Student with atributes and send this data to firebase.

On Asking Suppport Screen building witheger with Google Map API custom map where location of students with Logitude and Latitude will be represented by Student atribute location to meet. Also the pin on the location, when will be cliecked, will sow information window with contact details of such student with modules that they were facing together with their status. And user can call or send message or email via the given contact details.

That`s All related to the functionality of the developped mobile applciation.
In conclusion system can have a high benefit on additional iterations in relation to add features such

as requests for modules that are having finished status, notifications for created fequests.
Such a features was in a mind to be placed, however consider timing constrains were taken as not realistic.