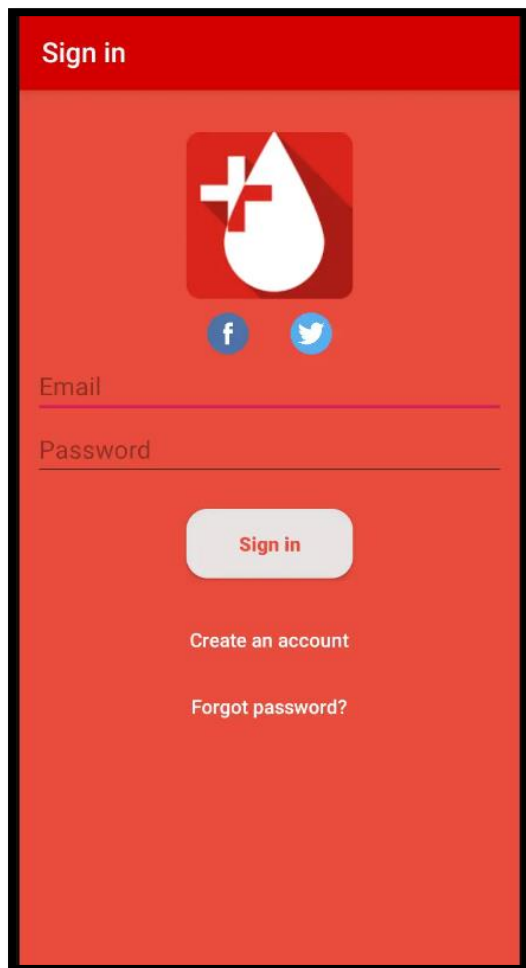


Week 2 – Android Application Development with Kotlin

Name: Suyash Patil

Registration Number: 20BIT0257

Screenshots:



Source Code:

//Suyash Patil 20BIT0257

```
import android.app.DatePickerDialog
import android.content.Context
import android.os.Bundle
import android.telephony.SmsManager
import android.widget.DatePicker
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.rememberLauncherForActivityResult
import androidx.activity.compose.setContent
import androidx.activity.result.contract.ActivityResultContracts
import androidx.compose.foundation.BorderStroke
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.border
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.selection.selectable
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.outlined.Email
import androidx.compose.material.icons.outlined.LocationOn
import androidx.compose.material.icons.outlined.Person
import androidx.compose.material.icons.outlined.Phone
import androidx.compose.material3.AlertDialog
```

```
import androidx.compose.material3.Button
import androidx.compose.material3.ButtonDefaults
import androidx.compose.material3.DropdownMenuItem
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.ExposedDropdownMenuBox
import androidx.compose.material3.ExposedDropdownMenuDefaults
import androidx.compose.material3.Icon
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.RadioButton
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.material3.TextField
import androidx.compose.material3.TextFieldDefaults
import androidx.compose.material3.TopAppBar
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.TextFieldValue
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import androidx.navigation.compose.NavHost
```

```

import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import java.util.*

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Surface(
                modifier = Modifier.fillMaxSize(),
                color = MaterialTheme.colorScheme.background
            ) {
                val getpermission = rememberLauncherForActivityResult(
                    ActivityResultContracts.RequestPermission()
                ) { isGranted ->
                    if (isGranted) {
                        //permission accepted Do Something
                    } else {
                        // Permission not accepted show message
                    }
                }
                SideEffect {
                    getpermission.launch(android.Manifest.permission.SEND_SMS)
                }
                App(applicationContext)
            }
        }
    }
}

```

```

@Composable
fun App(context: Context)
{
    val navController = rememberNavController()

    NavHost(
        navController = navController,
        startDestination = "home"
    ) {
        composable("home") {
            HomePage(navController = navController)
        }
        composable("submit/{name}/{email}/{mobile}/{gen}/{blg}/{location}") {
            backStackEntry ->
            val name = backStackEntry.arguments?.getString("name")
            val email = backStackEntry.arguments?.getString("email")
            val phone = backStackEntry.arguments?.getString("mobile")
            val gender = backStackEntry.arguments?.getString("gen")
            val bloodgrp = backStackEntry.arguments?.getString("blg")
            val loc = backStackEntry.arguments?.getString("location")

            NextPage(context,name=name,email=email,mobile=phone,gen=gender,blg=bloodgrp,location=loc)
        }
    }
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun HomePage(navController: NavController)
{
    Scaffold(

```

```

        topBar = { TopBar("Registration") },
        content = {pad -> FormFill(pad,navController) },
    )
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun TopBar(abc:String?) {
    TopAppBar(
        title = {
            Row(
                verticalAlignment = Alignment.CenterVertically,
                horizontalArrangement = Arrangement.Center,
                modifier = Modifier
                    .fillMaxSize()
                    .fillMaxWidth()
                    .padding(end = 20.dp)
                    .background(color = Color.White)
            ) {
                Text(
                    ""+abc,
                    color = Color.Red,
                    textAlign = TextAlign.Center,
                    fontSize = 25.sp,
                    fontWeight = FontWeight.Bold
                )
            }
        }
    )
}

```

```

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun FormFill(h: PaddingValues,navController: NavController) {
    Column(
        modifier = Modifier
            .fillMaxWidth()
            .fillMaxSize()
            .background(color = Color.White)
            .padding(25.dp),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.SpaceEvenly
    )
    {
        val name = remember { mutableStateOf(TextFieldValue("")) }
        val email = remember { mutableStateOf(TextFieldValue("")) }
        val mobile = remember { mutableStateOf(TextFieldValue("")) }
        var gen by remember { mutableStateOf("") }
        var blg by remember { mutableStateOf("") }
        val location = remember { mutableStateOf(TextFieldValue("")) }
        val mYear: Int
        val mMonth: Int
        val mDay: Int
        val mCalendar = android.icu.util.Calendar.getInstance()
        mYear = mCalendar.get(android.icu.util.Calendar.YEAR)
        mMonth = mCalendar.get(android.icu.util.Calendar.MONTH)
        mDay = mCalendar.get(android.icu.util.Calendar.DAY_OF_MONTH)
        val dobdate = remember { mutableStateOf("") }
        val lastdate = remember { mutableStateOf("") }
        val mContext = LocalContext.current
    }
}

```

```

val dob = DatePickerDialog(
    mContext,
    { _: DatePicker, mYear: Int, mMonth: Int, mDayOfMonth: Int ->
        dobdate.value = "$mDayOfMonth/${mMonth + 1}/${mYear}"
    }, mYear, mMonth, mDay
)

val lastime = DatePickerDialog(
    mContext,
    { _: DatePicker, mYear: Int, mMonth: Int, mDayOfMonth: Int ->
        lastdate.value = "$mDayOfMonth/${mMonth + 1}/${mYear}"
    }, mYear, mMonth, mDay
)

Image(
    painterResource(id = R.drawable.bloodbank),
    contentDescription = "logo",
    modifier = Modifier
        .size(width = 90.dp, height = 90.dp)
        .padding(top = 30.dp)
)

Text(
    "Every Blood Donor is a Hero",
    color = Color.Red,
    textAlign = TextAlign.Center,
    fontWeight = FontWeight.Bold,
    fontSize = 10.sp
)

TextField(
    value = name.value,
    onValueChange = {

```



```

        name.value = it
    },
    colors = TextFieldDefaults.textFieldColors(containerColor = Color.White),
    label = { Text("Enter Full Name") },
    modifier = Modifier.fillMaxWidth().border(BorderStroke(2.dp, Color.Black)),
    leadingIcon = { Icon(Icons.Outlined.Person, contentDescription = null) }
)
TextField(
    value = email.value,
    onChange = {
        email.value = it
    },
    colors = TextFieldDefaults.textFieldColors(containerColor = Color.White),
    label = { Text("Enter Email ID") },
    modifier = Modifier.fillMaxWidth().border(BorderStroke(2.dp, Color.Black)),
    keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Email),
    leadingIcon = { Icon(Icons.Outlined.Email, contentDescription = null) }
)
TextField(
    value = mobile.value,
    onChange = {
        mobile.value = it
    },
    colors = TextFieldDefaults.textFieldColors(containerColor = Color.White),
    label = { Text("Enter Mobile Number") },
    modifier = Modifier.fillMaxWidth().border(BorderStroke(2.dp, Color.Black)),
    keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Phone),
    leadingIcon = { Icon(Icons.Outlined.Phone, contentDescription = null) }
)

```

```

TextField(
    value = location.value,
    onChange = {
        location.value = it
    },
    colors = TextFieldDefaults.textFieldColors(containerColor = Color.White),
    label = { Text("Enter Location / City") },
    modifier = Modifier.fillMaxWidth().border(BorderStroke(2.dp, Color.Black)),
    shape = RoundedCornerShape(5.dp),
    leadingIcon = { Icon(Icons.Outlined.LocationOn, contentDescription = null) }
)
blg = bloodGroup()
Text(
    "Select the Gender",
    color = Color.Black,
    fontWeight = FontWeight.Bold,
    fontSize = 16.sp
)
gen = gender()
Button(
    onClick = { dob.show() },
    modifier = Modifier.fillMaxWidth(),
    colors = ButtonDefaults.buttonColors(Color(0xFFFC2141)),
    shape = RoundedCornerShape(5.dp),
)
{
    Text(
        text = "DoB "+ dobdate.value,
        color = Color.White,
    )
}

```

```

        textAlign = TextAlign.Center,
        fontWeight = FontWeight.Bold
    )
}
Button(
    onClick = { lastime.show() },
    modifier = Modifier.fillMaxWidth(),
    colors = ButtonDefaults.buttonColors(Color(0xFFFC2141)),
    shape = RoundedCornerShape(5.dp)
)
{
    Text(
        text = "Last Blood Donation Date "+ lastdate.value,
        color = Color.White,
        textAlign = TextAlign.Center,
        fontWeight = FontWeight.Bold
    )
}
Button(
    onClick = {

navController.navigate("submit/${name.value.text}/${email.value.text}/${mobile.value.text}/${gen}/${blg}/${location.value.text}")
    },
    modifier = Modifier.fillMaxWidth(),
    colors = ButtonDefaults.buttonColors(Color(0xFF634A4D)),
    shape = RoundedCornerShape(5.dp)
)
{
    Text(

```

```

        text = "Submit Registration Details",
        color = Color.White,
        textAlign = TextAlign.Center,
        fontWeight = FontWeight.Bold
    )
}
}
}

@Composable
fun gender():String {
    val radioOptions = listOf("Male", "Female", "Other")
    val (selectedOption, onOptionSelected) = remember { mutableStateOf(radioOptions[2]) }

    Column {
        Row {
            radioOptions.forEach { text ->
                Row(
                    Modifier.selectable(selected = (text == selectedOption),
                        onClick = { onOptionSelected(text) }
                    )
                ){
                    RadioButton(
                        selected = (text == selectedOption),
                        onClick = {
                            onOptionSelected(text)
                        },
                    )
                    Text(
                        text = text,
                        modifier = Modifier.padding(top = 15.dp)
                    )
                }
            }
        }
    }
}

```

```

        )
    }
}
}
}
return selectedOption
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun bloodGroup():String {
    var isexpanded by remember { mutableStateOf(false) }
    var bloodgrp by remember { mutableStateOf("") }
    val bloodgroups = listOf("A+", "A-", "B+", "B-", "AB+", "AB-", "O+", "O-")
    ExposedDropdownMenuBox(expanded = isexpanded,
        onExpandedChange = { isexpanded = it })
    {
        TextField(
            value = bloodgrp,
            onValueChange = {},
            readOnly = true,
            trailingIcon = { ExposedDropdownMenuDefaults.TrailingIcon(expanded = isexpanded)
        },

        colors = ExposedDropdownMenuDefaults.textFieldColors(containerColor =
Color.White),
        modifier = Modifier.fillMaxWidth()
            .menuAnchor()
            .border(BorderStroke(2.dp, Color.Black)),
        placeholder = { Text(text = "Select Blood Group") },
    )
    ExposedDropdownMenu(

```

```

        expanded = isexpanded,
        onDismissRequest = { isexpanded = false },
        modifier = Modifier.background(color = Color.White)
    )
}

bloodgroups.forEach { bloodgroups ->
    DropdownMenuItem(
        text = { Text(text = bloodgroups) },
        onClick = {
            bloodgrp = bloodgroups
            isexpanded = false
        })
    }
}

return bloodgrp
}

@Composable
@OptIn(ExperimentalMaterial3Api::class)
fun NextPage(context: Context, name: String?, email: String?, mobile: String?, gen: String?,
    blg: String?, location: String?)
{
    Scaffold(
        topBar = { TopBar("Confirm Details") },
        content = { pad -> NextPageContent(pad, context, name, email, mobile, gen, blg, location) },
    )
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable

```

```
fun NextPageContent(paddingvalues:PaddingValues, context: Context, name:String?,
email:String?, mobile:String?, gen:String?, blg:String?, location:String?) {
```

```
    TopAppBar(
```

```
        title = {
```

```
            Row(
```

```
                verticalAlignment = Alignment.CenterVertically,
```

```
                horizontalArrangement = Arrangement.Center,
```

```
                modifier = Modifier
```

```
                    .fillMaxSize()
```

```
                    .fillMaxWidth()
```

```
                    .padding(end = 20.dp)
```

```
                    .background(color = Color.White)
```

```
            ) {
```

```
                Text(
```

```
                    "",
```

```
                    color = Color.Red,
```

```
                    textAlign = TextAlign.Center,
```

```
                    fontSize = 25.sp,
```

```
                    fontWeight = FontWeight.Bold
```

```
                )
```

```
            }
```

```
        }
```

```
    )
```

```
    Column(
```

```
        horizontalAlignment = Alignment.CenterHorizontally,
```

```
        verticalArrangement = Arrangement.Top,
```

```
        modifier = Modifier
```

```
            .fillMaxSize()
```

```
            .fillMaxWidth()
```

```
            .background(color = Color.White)
```

```

        .padding(20.dp)
    ) {
        val details = listOf("Name", "Email", "Mobile", "Gender", "Blood Group", "Location")
        val params = listOf(name, email, mobile, gen, blg, location)
        val openDialog = remember { mutableStateOf(false) }
        Spacer(modifier = Modifier.padding(top = 40.dp))
        LazyColumn(modifier = Modifier.padding(5.dp))
        {
            items(params) { paramval ->
                (
                    Button(
                        onClick = {},
                        modifier = Modifier
                            .width(300.dp)
                            .height(50.dp),
                        colors = ButtonDefaults.buttonColors(Color(0xFFFC2141)),
                        shape = RoundedCornerShape(10),
                    )
                {
                    Text(
                        text = "" + details[params.indexOf(paramval)] + ": " + paramval,
                        fontWeight = FontWeight.Bold,
                        fontSize = 17.sp
                    )
                })
            Spacer(modifier = Modifier.padding(20.dp))
        }
        item {
            Button(

```



```

onClick = { openDialog.value = true },
modifier = Modifier
    .width(300.dp)
    .height(50.dp),
colors = ButtonDefaults.buttonColors(Color(0xFF634A4D)),
shape = RoundedCornerShape(10),
) {
    Text(text = "Confirm the Above Details")
}
if (openDialog.value) {
    AlertDialog(
        onDismissRequest = { openDialog.value = false },
        title = { Text(text = "Alert") },
        text = { Text("Want to send confirmation message to this Number ?") },
        dismissButton = {
            Button(
                onClick = {
                    openDialog.value = false
                    try {
                        val smsManager: SmsManager = SmsManager.getDefault()
                        smsManager.sendTextMessage(
                            "+91" + mobile,
                            null,
                            "Thanks for Your Information - BloodBank Application",
                            null,
                            null
                        )
                        Toast.makeText(context, "Message Sent", Toast.LENGTH_LONG)
                            .show()
                    }
                }
            )
        }
    )
}

```

```
        } catch (e: Exception) {

            Toast.makeText(

                context,

                "Error: " + e.message,

                Toast.LENGTH_LONG

            ).show()

        }

    },

    modifier = Modifier

        .width(100.dp)

        .height(45.dp),

    colors = ButtonDefaults.buttonColors(Color(0xFF268D2A)),

    shape = RoundedCornerShape(10)

)

{

    Text("Yes")

}

},

confirmButton = {

    Button(

        onClick = { openDialog.value = false },

        modifier = Modifier

            .width(100.dp)

            .height(45.dp),

        colors = ButtonDefaults.buttonColors(Color(0xFF268D2A)),

        shape = RoundedCornerShape(10)

    )

    {

        Text("Cancel")

    }

}
```

```
        }
    })
}
}
}
}
}
```

Link:

https://drive.google.com/drive/folders/1o14KYFp1TOliDMoT6-_BGnNpsNKEJQRJ?usp=share_link