A demonstration of text input and validation with android compose

Team Leader: UTHESH KUMAR S

Team members: UDHAYAMAHA S

THANDAVAN K

INTRODUCTION:

OVERVIEW:

Input validation is the process of checking whether the input data provided to a software application or system is valid, correct, and appropriate for processing. This is an essential step in software development to ensure the reliability, security, and robustness of the application.

Input validation typically involves checking the format, type, range, and size of input data against predefined rules and constraints. For example, validating an email address could involve checking that it has a valid format (e.g., contains an @ symbol and a domain name), whereas validating a numeric input could involve checking that it falls within a specific range.

Validating input data can prevent a wide range of problems, such as buffer overflow attacks, injection attacks, data corruption, and data loss. By ensuring that input data is valid, software applications can operate as expected and provide accurate results to users.

PURPOSE:

Input validation is a crucial aspect of software development, and it serves various purposes, including:

Ensuring data accuracy: Input validation helps to ensure that the data entered into a software application is accurate and complete. This ensures that the application operates as intended and produces accurate results.

Preventing security vulnerabilities: Input validation helps to prevent security vulnerabilities such as SQL injection attacks, buffer overflow attacks, and cross-site scripting attacks. By validating input data, developers can prevent malicious actors from exploiting vulnerabilities and gaining unauthorized access to the system.

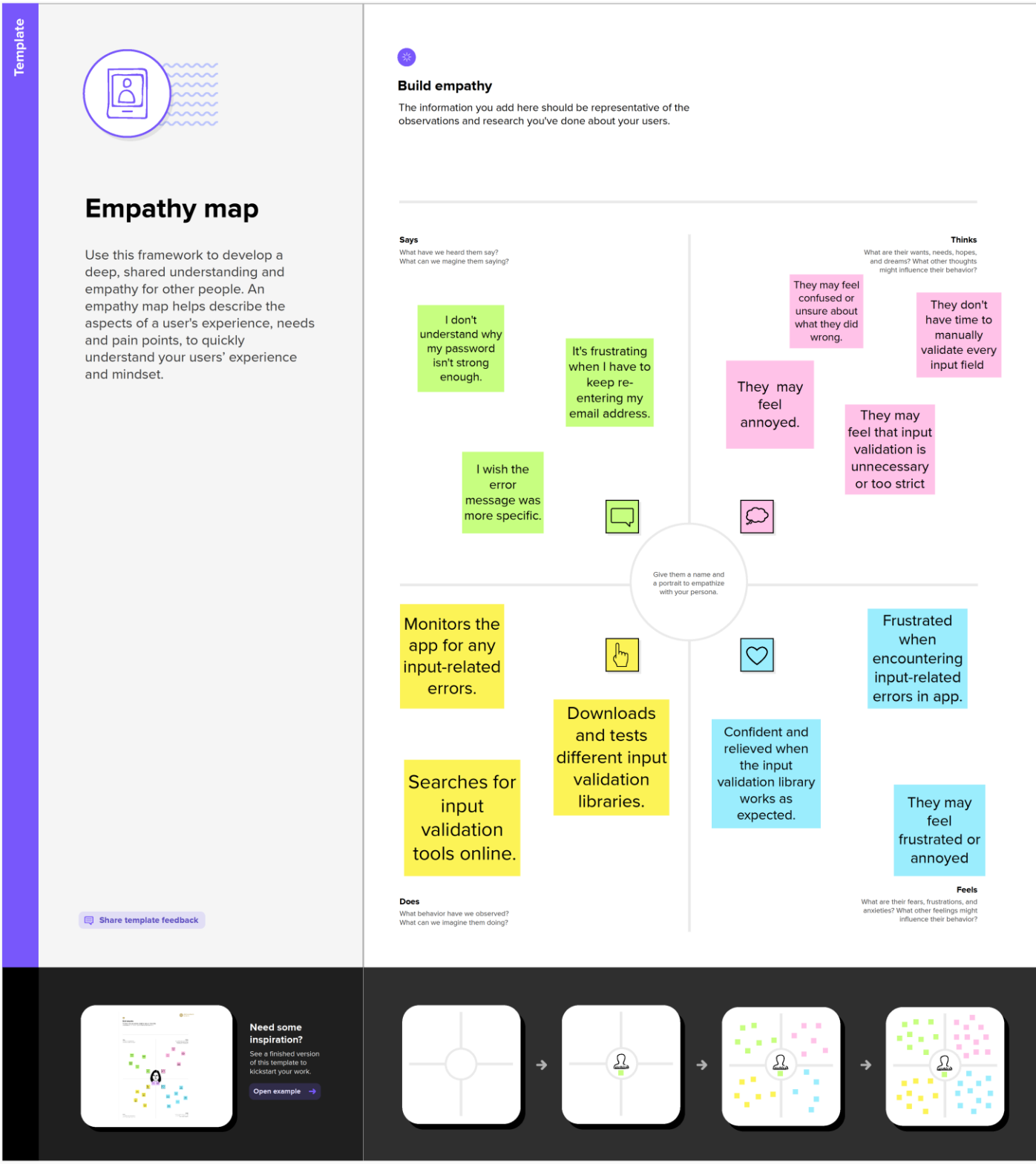
Improving user experience: Validating input data can improve the user experience by providing immediate feedback to users if they enter invalid data. This feedback can guide the user in correcting the input and reduces frustration and errors caused by incorrect input.

Reducing data processing errors: Input validation helps to reduce data processing errors by preventing the processing of incomplete or incorrect data. This helps to improve the accuracy and reliability of the data processed by the application.

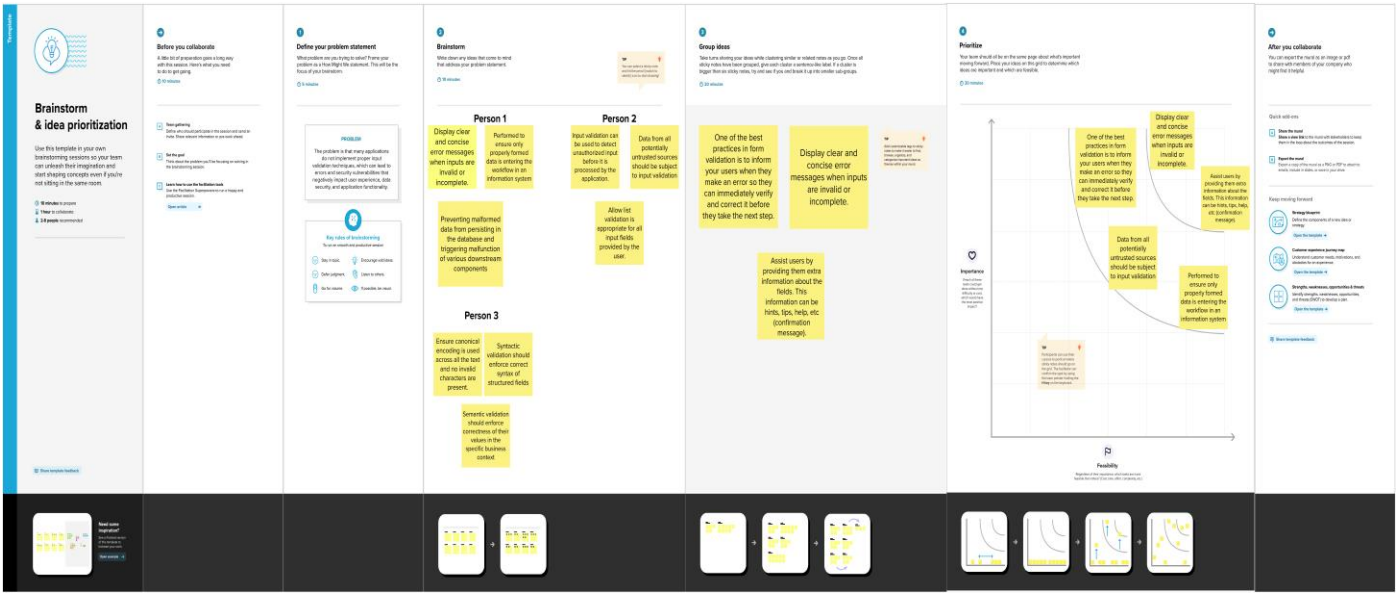
Ensuring compliance with regulations: Input validation is essential in ensuring that software applications comply with regulations, such as the General Data Protection Regulation (GDPR) and the Health Insurance Portability and Accountability Act (HIPAA). These regulations require that personal data is processed securely and accurately, and input validation helps to ensure compliance with these requirements.

PROBLEM DEFINITION AND DESIGN THINKING

EMPATHY MAP:

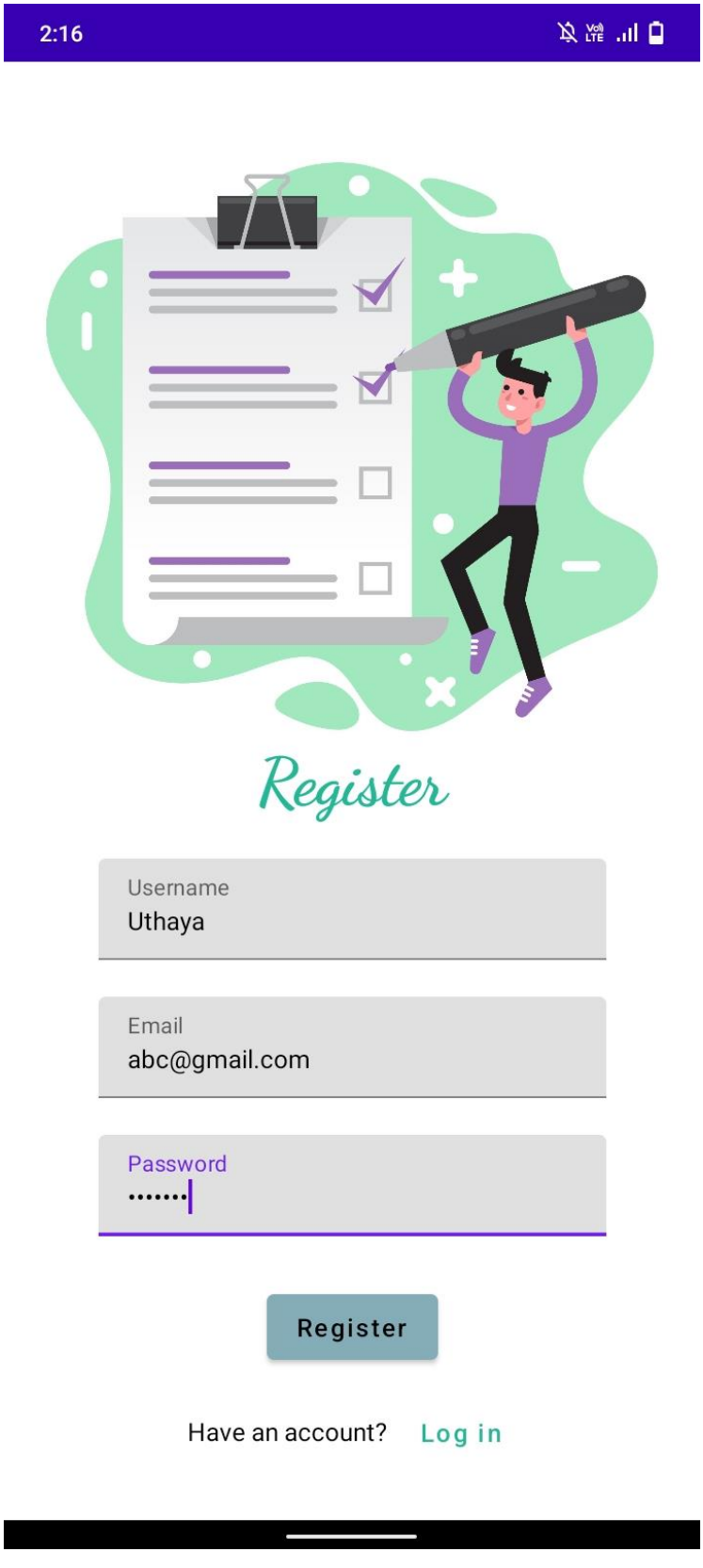


IDEATION & BRAINSTORMING MAP:



RESULT

Register Page



Login Page

2:17

Login

Username
Uthaya

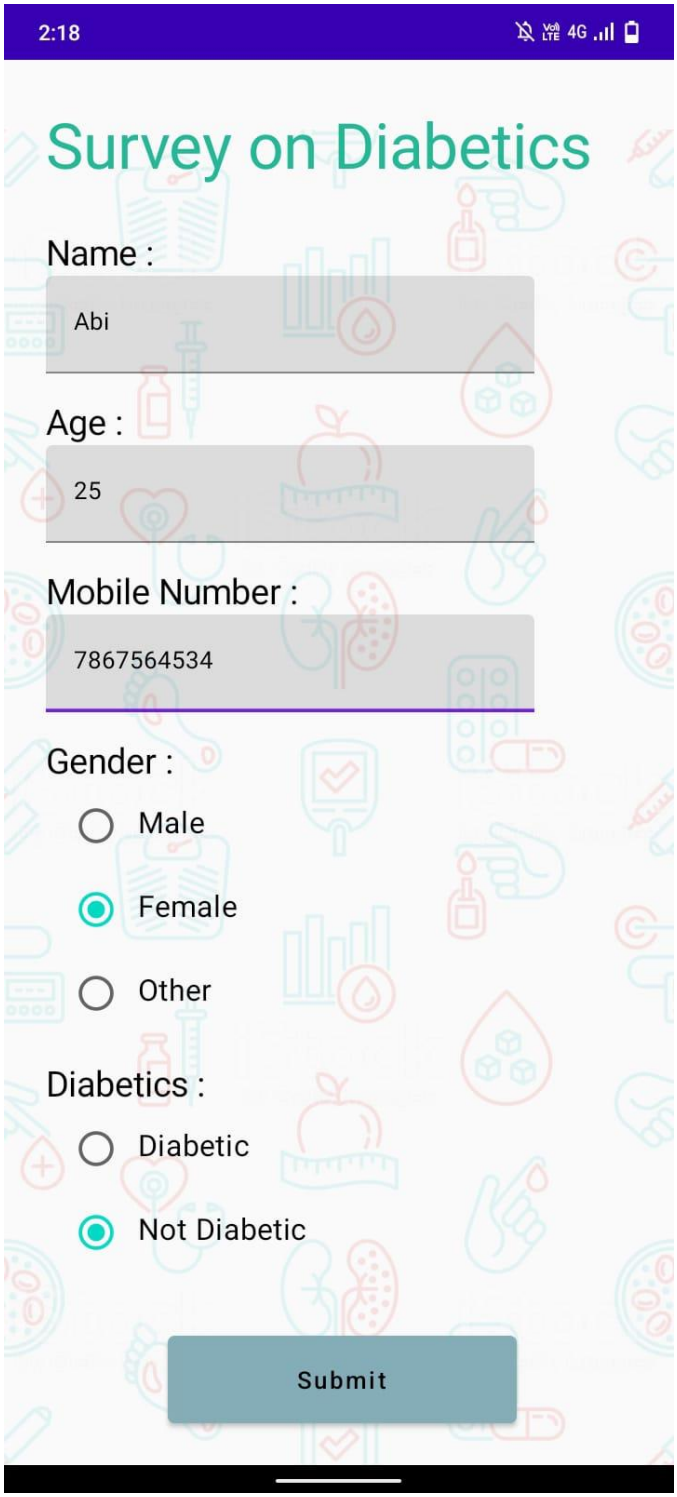
Password
•••••••••

Login

Register

Forget password?

Entry Page


2:18
Survey on Diabetics
Name :
Abi
Age :
25
Mobile Number :
7867564534
Gender :
Male
Female
Other
Diabetics :
Diabetic
Not Diabetic
Submit

ADVANTAGES & DISADVANTAGES

ADVANTAGES:

Increased data accuracy: Input validation helps to ensure that the data entered is accurate, complete, and consistent. This can improve the quality of the data and prevent errors that can occur due to user input.

Improved security: Input validation can help to prevent malicious attacks such as SQL injection and cross-site scripting by validating the input data and rejecting any inputs that contain potentially harmful code.

Better user experience: By validating input data, an app can provide real-time feedback to users on whether their input is valid or not. This can prevent frustration and improve the user experience.

Reduced development time: An input validation app can help to reduce the development time by providing pre-built validation rules that can be easily integrated into the app.

Consistency: Input validation ensures that all data entered into the app conforms to a specific format, making it easier to process and analyse the data consistently.

Cost-effective: Input validation can help to reduce costs associated with fixing errors and addressing security breaches caused by faulty input data.

Overall, an input validation app can help to improve the quality, security, and user experience of an application while reducing development time and costs.

DISADVANTAGES:

Overly restrictive: If the input validation rules are too strict, it can result in legitimate data being rejected, leading to frustration for users and potentially affecting the functionality of the app.

Complex implementation: Implementing input validation can be complex, especially in cases where the data being validated is highly variable or unpredictable. This can result in longer development times and potentially higher costs.

Difficulty with legacy systems: In some cases, it may be difficult to implement input validation in legacy systems that were not originally designed with input validation in mind. This can result in additional development costs and time.

Increased risk of false positives: Input validation can result in false positives, where legitimate data is rejected due to overly restrictive validation rules. This can be frustrating for users and can potentially impact the functionality of the app.

May not catch all errors: While input validation can help to catch many errors, it may not catch all errors, especially if the validation rules are not comprehensive or if users find ways to bypass the validation.

Maintenance: Input validation rules may need to be updated over time as new data types are added or as new security threats emerge. This can result in ongoing maintenance costs for the app.

APPLICATIONS

An input validation app can be used in a variety of applications, including:

Web forms: Input validation can be used to validate data entered into web forms, such as contact forms, registration forms, and checkout forms, to ensure that the data is accurate and complete.

E-commerce: Input validation can be used to validate credit card numbers, expiration dates, and CVV codes in e-commerce applications to prevent fraud and ensure that transactions are secure.

Healthcare: Input validation can be used in healthcare applications to ensure that patient data is entered accurately and consistently across different systems, helping to improve patient care and reduce errors.

Banking: Input validation can be used in banking applications to validate account numbers, routing numbers, and other financial data to ensure that transactions are accurate and secure.

Mobile apps: Input validation can be used in mobile apps to validate user input, such as login credentials, search queries, and form data, to ensure that the app functions correctly and user data is secure.

Gaming: Input validation can be used in gaming applications to validate user input, such as button presses or joystick movements, to ensure that the game functions correctly and prevent cheating.

CONCLUSION

In conclusion, an input validation app can provide several advantages, including increased data accuracy, improved security, better user experience, reduced development time, consistency, and cost-effectiveness. However, there are also potential disadvantages to consider, such as overly restrictive validation rules, complex implementation, difficulty with legacy systems, increased risk of false positives, incomplete error catching, and ongoing maintenance costs.

Despite these potential challenges, input validation is a critical component of many different types of applications, including web forms, e-commerce, healthcare, banking, mobile apps, and gaming. By implementing input validation in a way that balances data accuracy, security, and user experience, developers can help to ensure that their applications are effective, efficient, and user-friendly.

FUTURE SCOPE

The future scope of input validation app is likely to expand as technology continues to evolve and new applications are developed.

Some of the potential areas of growth and innovation for input validation apps include:

Integration with AI and machine learning: As mentioned earlier, future input validation apps are likely to incorporate more sophisticated algorithms and machine learning techniques to more accurately identify valid input data and prevent false positives. This integration could enable input validation apps to learn from past inputs and patterns to improve the accuracy and efficiency of data validation.

Increased customization: Future input validation apps may provide more customizable validation rules, allowing developers to tailor input validation to specific use cases and data types.

Integration with blockchain technology: Blockchain technology could be used to ensure the security and immutability of validated data, providing an additional layer of security for sensitive data.

Integration with IoT devices: With the proliferation of IoT devices, input validation apps may be used to validate data from a wide range of sensors and devices, helping to ensure the accuracy and consistency of data collected from these devices.

U**se in fields such as artificial intelligence and data analytics**: Input validation apps may be used in fields such as artificial intelligence and data analytics to ensure the accuracy and consistency of data used for training models and making decisions based on data.

APPENDIX

SOURCE CODE:

Build.gradle(Survey Application)

```
buildscript {
    ext {
        compose_ui_version = '1.2.0'
    }
}// Top-level build file where you can add configuration options common to
all sub-projects/modules.
plugins {
    id 'com.android.application' version '7.3.1' apply false
    id 'com.android.library' version '7.3.1' apply false
    id 'org.jetbrains.kotlin.android' version '1.7.0' apply false
}
```

Build.gradle(:app)

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
}

android {
    namespace 'com.example.surveyapplication'
    compileSdk 33

    defaultConfig {
        applicationId "com.example.surveyapplication"
        minSdk 30
        targetSdk 33
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        vectorDrawables {
            useSupportLibrary true
        }
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-
optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
    buildFeatures {
        compose true
    }
    composeOptions {
        kotlinCompilerExtensionVersion '1.2.0'
    }
    packagingOptions {
        resources {
            excludes += '/META-INF/{AL2.0,LGPL2.1}'
        }
    }
}

dependencies {

    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'
    implementation 'androidx.activity:activity-compose:1.3.1'
    implementation "androidx.compose.ui:ui:$compose_ui_version"
    implementation "androidx.compose.ui:ui-tooling-
preview:$compose_ui_version"
    implementation 'androidx.compose.material:material:1.2.0'
    implementation 'androidx.room:room-common:2.5.0'
```

```
    implementation 'androidx.room:room-ktx:2.5.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
    androidTestImplementation "androidx.compose.ui:ui-test-
junit4:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-
tooling:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-test-
manifest:$compose_ui_version"
}
```

AdminActivity.kt

```
package com.example.surveyapplication

import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.LazyRow
import androidx.compose.foundation.lazy.items
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.surveyapplication.ui.theme.SurveyApplicationTheme

class AdminActivity : ComponentActivity() {
    private lateinit var databaseHelper: SurveyDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = SurveyDatabaseHelper(this)
        setContent {
            val data = databaseHelper.getAllSurveys();
            Log.d("swathi", data.toString())
            val survey = databaseHelper.getAllSurveys()
            ListListScopeSample(survey)
        }
    }
}
@Composable
fun ListListScopeSample(survey: List<Survey>) {

    Image(
        painterResource(id = R.drawable.background), contentDescription =
"",
        alpha =0.1F,
        contentScale = ContentScale.FillHeight,
        modifier = Modifier.padding(top = 40.dp)
    )

    Text(
        text = "Survey Details",
        modifier = Modifier.padding(top = 24.dp, start = 106.dp, bottom =
24.dp),
        fontSize = 30.sp,
        color = Color(0xFF25b897)
    )
    Spacer(modifier = Modifier.height(30.dp))
    LazyRow(
        modifier = Modifier
            .fillMaxSize()
```

```
            .padding(top = 80.dp),

        horizontalArrangement = Arrangement.SpaceBetween
    ) {
        item {

            LazyColumn {
                items(survey) { survey ->
                    Column(
                        modifier = Modifier.padding(
                            top = 16.dp,
                            start = 48.dp,
                            bottom = 20.dp
                        )
                    ) {
                        Text("Name: ${survey.name}")
                        Text("Age: ${survey.age}")
                        Text("Mobile_Number: ${survey.mobileNumber}")
                        Text("Gender: ${survey.gender}")
                        Text("Diabetics: ${survey.diabetics}")
                    }
                }
            }
        }
    }
}
```

LoginActivity.kt

```
package com.example.surveyapplication

import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.surveyapplication.ui.theme.SurveyApplicationTheme

class LoginActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {

                LoginScreen(this, databaseHelper)

        }
    }
}

@Composable
fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) {

    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }

    Column(
        modifier = Modifier.fillMaxSize().background(Color.White),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {

        Image(painterResource(id = R.drawable.survey_login),
contentDescription = "")

        Text(
            fontSize = 36.sp,
            fontWeight = FontWeight.ExtraBold,
            fontFamily = FontFamily.Cursive,
```

```
        color = Color(0xFF25b897),
        text = "Login"
    )
    Spacer(modifier = Modifier.height(10.dp))

    TextField(
        value = username,
        onValueChange = { username = it },
        label = { Text("Username") },
        modifier = Modifier
            .padding(10.dp)
            .width(280.dp)
    )

    TextField(
        value = password,
        onValueChange = { password = it },
        label = { Text("Password") },
        visualTransformation = PasswordVisualTransformation(),
        modifier = Modifier
            .padding(10.dp)
            .width(280.dp)
    )

    if (error.isNotEmpty()) {
        Text(
            text = error,
            color = MaterialTheme.colors.error,
            modifier = Modifier.padding(vertical = 16.dp)
        )
    }

    Button(
        onClick = {
            if (username.isNotEmpty() && password.isNotEmpty()) {
                val user = databaseHelper.getUserByUsername(username)
                if (user != null && user.password == password) {
                    error = "Successfully log in"
                    context.startActivity(
                        Intent(
                            context,
                            MainActivity::class.java
                        )
                    )
                    //onLoginSuccess()
                }
                if (user != null && user.password == "admin") {
                    error = "Successfully log in"
                    context.startActivity(
                        Intent(
                            context,
                            AdminActivity::class.java
                        )
                    )
                }
                else {
                    error =  "Invalid username or password"
                }

            } else {
                error = "Please fill all fields"
```

```
                }
            },
            colors = ButtonDefaults.buttonColors(backgroundColor =
Color(0xFF84adb8)),
            modifier = Modifier.padding(top = 16.dp)
        ) {
            Text(text = "Login")
        }
        Row {
            TextButton(onClick = {context.startActivity(
                Intent(
                    context,
                    RegisterActivity::class.java
                )
            )}
            )
            { Text(color = Color(0xFF25b897),text = "Register") }
            TextButton(onClick = {
            })

            {
                Spacer(modifier = Modifier.width(60.dp))
                Text(color = Color(0xFF25b897),text = "Forget password?")
            }
        }
    }
}
private fun startMainPage(context: Context) {
    val intent = Intent(context, MainActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```

RegisterActivity.kt

```
package com.example.surveyapplication

import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.core.content.ContextCompat
import com.example.surveyapplication.ui.theme.SurveyApplicationTheme

class RegisterActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {

                    RegistrationScreen(this,databaseHelper)

        }
    }
}

@Composable
fun RegistrationScreen(context: Context, databaseHelper:
UserDatabaseHelper) {

    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var email by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }

    Column(
        modifier = Modifier.fillMaxSize().background(Color.White),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {

        Image(painterResource(id = R.drawable.survey_signup),
contentDescription = "")

        Text(
            fontSize = 36.sp,
```

```
            fontWeight = FontWeight.ExtraBold,
            fontFamily = FontFamily.Cursive,
            color = Color(0xFF25b897),
            text = "Register"
        )

        Spacer(modifier = Modifier.height(10.dp))
        TextField(
            value = username,
            onValueChange = { username = it },
            label = { Text("Username") },
            modifier = Modifier
                .padding(10.dp)
                .width(280.dp)

        )

        TextField(
            value = email,
            onValueChange = { email = it },
            label = { Text("Email") },
            modifier = Modifier
                .padding(10.dp)
                .width(280.dp)
        )

        TextField(
            value = password,
            onValueChange = { password = it },
            label = { Text("Password") },
            visualTransformation = PasswordVisualTransformation(),
            modifier = Modifier
                .padding(10.dp)
                .width(280.dp)
        )


        if (error.isNotEmpty()) {
            Text(
                text = error,
                color = MaterialTheme.colors.error,
                modifier = Modifier.padding(vertical = 16.dp)
            )
        }

        Button(
            onClick = {
                if (username.isNotEmpty() && password.isNotEmpty() &&
email.isNotEmpty()) {
                    val user = User(
                        id = null,
                        firstName = username,
                        lastName = null,
                        email = email,
                        password = password
                    )
                    databaseHelper.insertUser(user)
                    error = "User registered successfully"
                    // Start LoginActivity using the current context
                    context.startActivity(
                        Intent(
```

```
                            context,
                            LoginActivity::class.java
                        )
                    )

                } else {
                    error = "Please fill all fields"
                }
            },
            colors = ButtonDefaults.buttonColors(backgroundColor =
Color(0xFF84adb8)),
            modifier = Modifier.padding(top = 16.dp),

        ) {
            Text(text = "Register")
        }
        Spacer(modifier = Modifier.width(10.dp))
        Spacer(modifier = Modifier.height(10.dp))

        Row() {
            Text(
                modifier = Modifier.padding(top = 14.dp), text = "Have an
account?"
            )
            TextButton(onClick = {
                context.startActivity(
                    Intent(
                        context,
                        LoginActivity::class.java
                    )
                )
            })

            {
                Spacer(modifier = Modifier.width(10.dp))
                Text( color = Color(0xFF25b897),text = "Log in")
            }
        }
    }
}
private fun startLoginActivity(context: Context) {
    val intent = Intent(context, LoginActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```

Manifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.SurveyApplication"
        tools:targetApi="31">
        <activity
            android:name=".RegisterActivity"
            android:exported="false"
            android:label="@string/title_activity_register"
            android:theme="@style/Theme.SurveyApplication" />
        <activity
            android:name=".MainActivity"
            android:exported="false"
            android:label="MainActivity"
            android:theme="@style/Theme.SurveyApplication" />
        <activity
            android:name=".AdminActivity"
            android:exported="false"
            android:label="@string/title_activity_admin"
            android:theme="@style/Theme.SurveyApplication" />
        <activity
            android:name=".LoginActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.SurveyApplication">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Main Activity:

```
package com.example.surveyapplication

import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.surveyapplication.ui.theme.SurveyApplicationTheme

class MainActivity : ComponentActivity() {
private lateinit var databaseHelper: SurveyDatabaseHelper
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
databaseHelper = SurveyDatabaseHelper(this)
setContent {
FormScreen(this, databaseHelper)
}
}
}
```

```
@Composable
fun FormScreen(context: Context, databaseHelper: SurveyDatabaseHelper) {

Image(
painterResource(id = R.drawable.background), contentDescription = "",
alpha =0.1F,
contentScale = ContentScale.FillHeight,
modifier = Modifier.padding(top = 40.dp)
)


// Define state for form fields
var name by remember { mutableStateOf("") }
var age by remember { mutableStateOf("") }
var mobileNumber by remember { mutableStateOf("") }
var genderOptions = listOf("Male", "Female", "Other")
var selectedGender by remember { mutableStateOf("") }
var error by remember { mutableStateOf("") }
var diabeticsOptions = listOf("Diabetic", "Not Diabetic")
var selectedDiabetics by remember { mutableStateOf("") }

Column(
modifier = Modifier.padding(24.dp),
horizontalAlignment = Alignment.Start,
verticalArrangement = Arrangement.SpaceEvenly
) {

Text(
fontSize = 36.sp,
textAlign = TextAlign.Center,
text = "Survey on Diabetics",
color = Color(0xFF25b897)
)

Spacer(modifier = Modifier.height(24.dp))
```

```
Text(text = "Name :", fontSize = 20.sp)
TextField(
value = name,
onValueChange = { name = it },
)

Spacer(modifier = Modifier.height(14.dp))

Text(text = "Age :", fontSize = 20.sp)
TextField(
value = age,
onValueChange = { age = it },
)

Spacer(modifier = Modifier.height(14.dp))

Text(text = "Mobile Number :", fontSize = 20.sp)
TextField(
value = mobileNumber,
onValueChange = { mobileNumber = it },
)

Spacer(modifier = Modifier.height(14.dp))

Text(text = "Gender :", fontSize = 20.sp)
RadioGroup(
options = genderOptions,
selectedOption = selectedGender,
onSelectedChange = { selectedGender = it }
)

Spacer(modifier = Modifier.height(14.dp))

Text(text = "Diabetics :", fontSize = 20.sp)
RadioGroup(
options = diabeticsOptions,
selectedOption = selectedDiabetics,
onSelectedChange = { selectedDiabetics = it }
```

```
)

Text(

text = error,

textAlign = TextAlign.Center,

modifier = Modifier.padding(bottom = 16.dp)

)

// Display Submit button

Button(

onClick = { if (name.isNotEmpty() && age.isNotEmpty() && mobileNumber.isNotEmpty()
&& genderOptions.isNotEmpty() && diabeticsOptions.isNotEmpty()) {

val survey = Survey(

id = null,

name = name,

age = age,

mobileNumber = mobileNumber,

gender = selectedGender,

diabetics = selectedDiabetics

)

databaseHelper.insertSurvey(survey)

error = "Survey Completed"

} else {

error = "Please fill all fields"

}

},

colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFF84adb8)),

modifier = Modifier.padding(start = 70.dp).size(height = 60.dp, width = 200.dp)

) {

Text(text = "Submit")

}

}

}

@Composable
```

```
fun RadioGroup(
options: List<String>,
selectedOption: String?,
onSelectedChange: (String) -> Unit
) {
Column {
options.forEach { option ->
Row(
Modifier
.fillMaxWidth()
.padding(horizontal = 5.dp)
) {
RadioButton(
selected = option == selectedOption,
onClick = { onSelectedChange(option) }
)
Text(
text = option,
style = MaterialTheme.typography.body1.merge(),
modifier = Modifier.padding(top = 10.dp),
fontSize = 17.sp
)
}
}
}
}
```