COMPOSE INPUT: A DEMONSTRATION OF TEXT INPUT AND VALIDATION WITH ANDROID COMPOSE

**T. SARANI**

**S.DIVYA**

**P.PRIYADHARSHINI**

**P.KAVIYA**

## 1. INTRODUCTION

## *1.1 Overview*

The app is a sample project that demonstrates how to use the Android Compose UI toolkit to build a survey app. The app allows the user to answer a series of questions. It showcases some of the key features of the Compose UI toolkit, data management, and user interactions.

**Project Workflow:**

* Users register into the application.

* After registration, user logins into the application.
* User enters into the main page
* From Admin Side he can login to the app and can view all the data.

*1.2 Purpose*

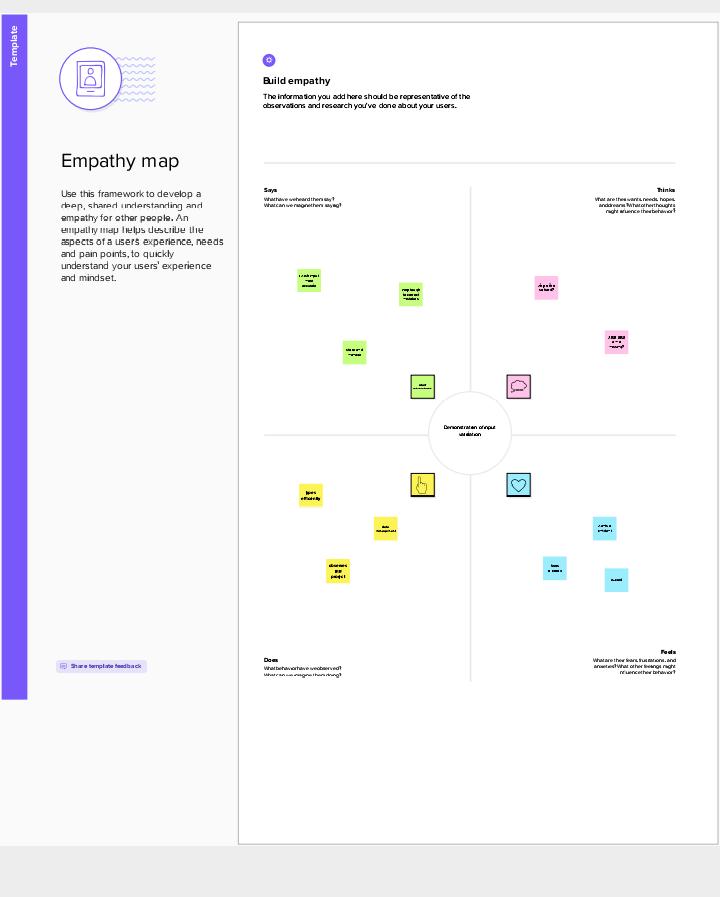
The purpose of a demonstration of text input and validation with Android Compose is to showcase how developers can use the Compose framework to create user interfaces for text input forms and fields, and to demonstrate how to validate user input to ensure it meets certain criteria.

Text input and validation are common tasks in mobile app development, and Compose offers a modern, declarative way to create UI components for handling text input. With Compose, developers can define text input fields with customizable attributes, such as text size, color, and placeholder text, and can add validation logic to ensure that user input meets specific criteria, such as minimum or maximum length, required format, or presence of certain characters.

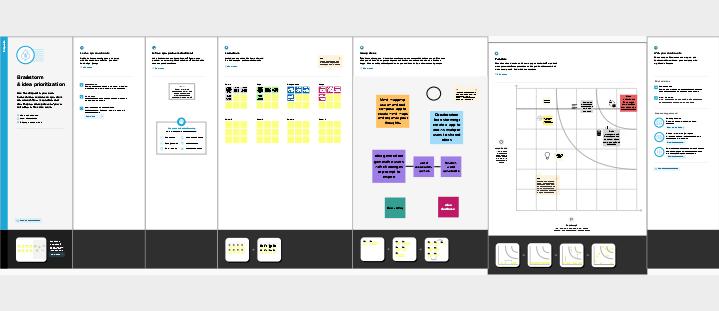
A demonstration of text input and validation with Android Compose can showcase how to create a form with multiple input fields, how to handle user input events, how to perform validation checks, and how to display error messages or feedback to the user. By demonstrating these concepts, developers can learn how to use Compose to create robust, user-friendly forms that provide a seamless user experience while also ensuring data accuracy and completeness.

**2. Purpose Definition & Design Thinking**

*2.1 Empathy map*

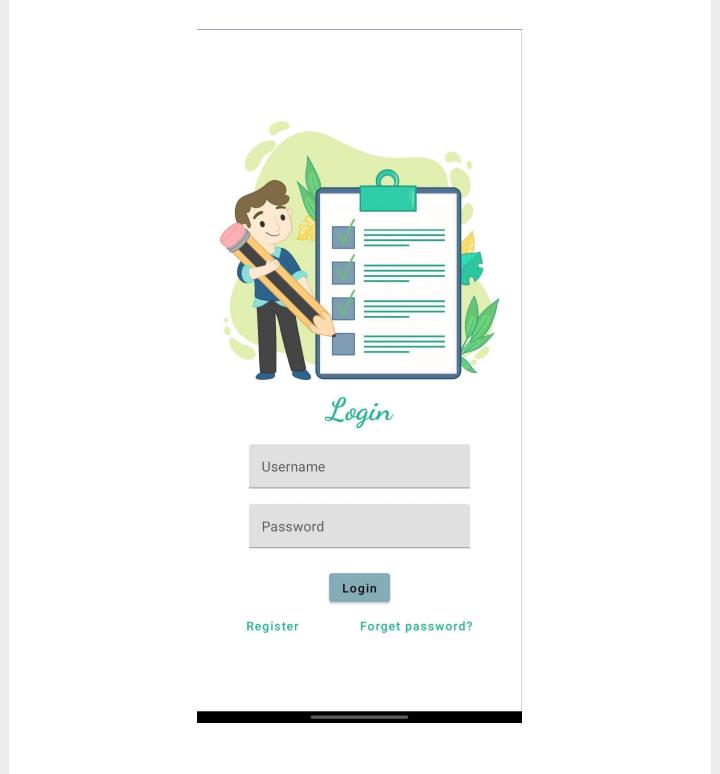


*2.2 Ideation & Brainstorming Map*

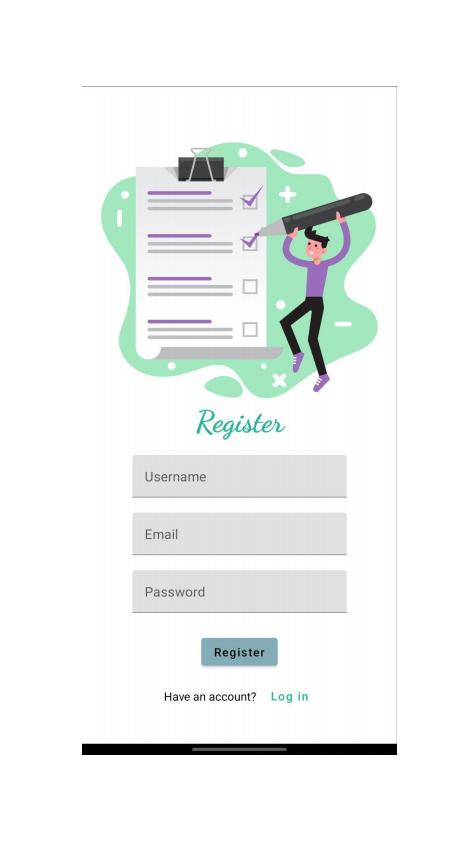


**3. RESULT**

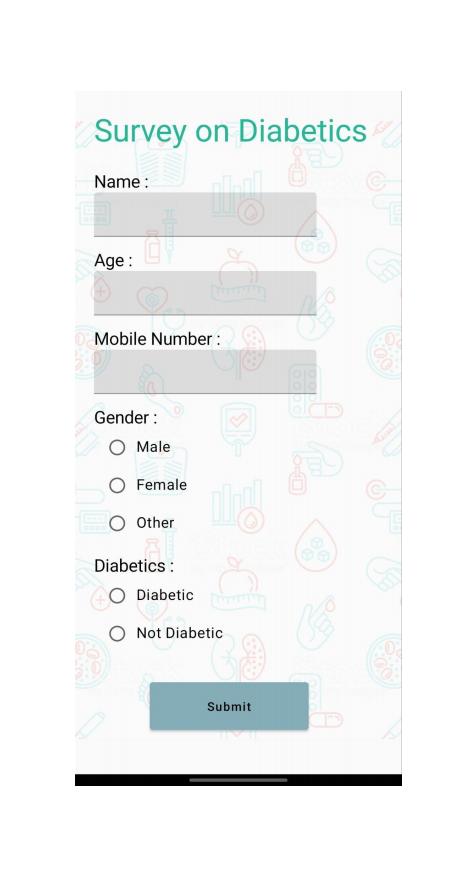
Login page:



Register page:



Main page:



**4 ADVANTAGES AND DISADVANTAGES**

*Advantages:*

Simplified syntax: Compose offers a simplified syntax for defining UI elements, including text inputs, that makes it easier to create and maintain code.

Real-time preview: Compose offers a real-time preview of your UI as you write your code, allowing you to see changes as you make them.

Automatic validation: Compose provides built-in support for input validation, making it easy to ensure that user input meets your application's requirements.

Improved performance: Compose uses a more efficient rendering system, resulting in faster UI rendering and improved performance.

*Disadvantages:*

Learning curve: Compose is a new technology, so there is a learning curve for developers who are new to it.

Limited documentation: Compose is still in its early stages, so there may be limited documentation and resources available.

Compatibility: Compose is only supported on newer versions of Android, so older devices may not be able to run applications built with Compose.

Complexity: While Compose simplifies many aspects of UI development, more complex UI elements can still be challenging to create.

**5 APPLICATIONS**

Login screens: User authentication is a common feature in many Android apps, and login screens typically require users to enter their email or username and password. Using the TextInputField component with validation can help ensure that users enter valid login credentials before attempting to log in.

Registration screens: Registration screens often require users to enter their name, email, password, and other personal information. Using the TextInputField component with validation can help ensure that the information entered is accurate and complete.

Search screens: Search screens allow users to enter keywords or phrases to find information within an app. using the TextInputField component with validation can help ensure that the search query is valid and meets certain criteria, such as a minimum or maximum length..

Payment screens: Payment screens require users to enter their payment information, such as credit card numbers and expiration dates. Using the TextInputField component with validation can help ensure that the payment information entered is accurate and complete, and meets certain criteria, such as a valid credit card number.

Profile editing screens: Profile editing screens allow users to update their personal information, such as their name, email, and profile picture. Using the TextInputField component with validation can help ensure that the information entered is accurate and complete, and meets certain criteria, such as a valid email address.

**6 Conclusion**

In conclusion, Android Compose provides a powerful and flexible way to handle text input and validation in your app. With Compose, you can easily create custom input fields and apply validation logic to ensure that user input is correct and consistent. You can also take advantage of Compose's state management and event handling features to update your UI in real-time as the user interacts with your app.

When implementing text input and validation in your app with Compose, it's important to consider the specific needs of your app and your users. You should carefully design your input fields and validation logic to ensure that they are intuitive and easy to use. You should also thoroughly test your implementation to catch any bugs or issues before releasing your app to users.

**7 FUTURE SCOPES**

Custom input fields: Compose allow developers to create custom input fields that are tailored to the specific needs of the application. This can include fields for email addresses, phone numbers, dates, and more. Developers can use Compose's built-in components to create these custom input fields and add validation logic to ensure that the user input meets the required format.

Input formatting: With Compose, it is easy to format user input as they type. For example, developers can use Compose's built-in TextFormatter class to format phone numbers as the user types them. This can improve the user experience and reduce errors caused by incorrect formatting.

Autocomplete and suggestions: Compose makes it easy to implement autocomplete and suggestion functionality in text input fields. Developers can use Compose's built-in Autocomplete and Suggest classes to provide suggestions based on the user's input.

Accessibility: Compose provides built-in support for accessibility, which can be used to make text input fields more accessible to users with disabilities. This includes features such as screen reader support, high contrast mode, and more.

**8 APPENDIX**

1. Source code

Creating the database classes:

Step 1:

**Create user data class**

package com.example.surveyapplication

import androidx.room.ColumnInfo

import androidx.room.Entity

import androidx.room.PrimaryKey

@Entity(tableName = "user\_table")

data class User(

@PrimaryKey(autoGenerate = true) val id: Int?,

@ColumnInfo(name = "first\_name") val firstName: String?,

@ColumnInfo(name = "last\_name") val lastName: String?,

@ColumnInfo(name = "email") val email: String?,

@ColumnInfo(name = "password") val password: String?,

)

Step 2:

**Create an UserDao interface**

package com.example.surveyapplication

import androidx.room.\*

@Dao

interface UserDao {

@Query("SELECT \* FROM user\_table WHERE email = :email")

suspend fun getUserByEmail(email: String): User?

@Insert(onConflict = OnConflictStrategy.REPLACE)

suspend fun insertUser(user: User)

@Update

suspend fun updateUser(user: User)

@Delete

suspend fun deleteUser(user: User)

}

Step 3:

**Create an UserDatabase class**

package com.example.surveyapplication

import android.content.Context

import androidx.room.Database

import androidx.room.Room

import androidx.room.RoomDatabase

@Database(entities = [User::class], version = 1)

abstract class UserDatabase : RoomDatabase() {

abstract fun userDao(): UserDao

companion object {

@Volatile

private var instance: UserDatabase? = null

fun getDatabase(context: Context): UserDatabase {

return instance ?: synchronized(this) {

val newInstance = Room.databaseBuilder(

context.applicationContext,

UserDatabase::class.java,

"user\_database"

).build()

instance = newInstance

newInstance

}

}

}

}

Step 4:

**Create an UserDatabaseHelper**

package com.example.surveyapplication

import android.annotation.SuppressLint

import android.content.ContentValues

import android.content.Context

import android.database.Cursor

import android.database.sqlite.SQLiteDatabase

import android.database.sqlite.SQLiteOpenHelper

class UserDatabaseHelper(context: Context) :

SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) {

companion object {

private const val DATABASE\_VERSION = 1

private const val DATABASE\_NAME = "UserDatabase.db"

private const val TABLE\_NAME = "user\_table"

private const val COLUMN\_ID = "id"

private const val COLUMN\_FIRST\_NAME = "first\_name"

private const val COLUMN\_LAST\_NAME = "last\_name"

private const val COLUMN\_EMAIL = "email"

private const val COLUMN\_PASSWORD = "password"

}

override fun onCreate(db: SQLiteDatabase?) {

val createTable = "CREATE TABLE $TABLE\_NAME (" +

"$COLUMN\_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +

"$COLUMN\_FIRST\_NAME TEXT, " +

"$COLUMN\_LAST\_NAME TEXT, " +

"$COLUMN\_EMAIL TEXT, " +

"$COLUMN\_PASSWORD TEXT" +

")"

db?.execSQL(createTable)

}

override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {

db?.execSQL("DROP TABLE IF EXISTS $TABLE\_NAME")

onCreate(db)

}

fun insertUser(user: User) {

val db = writableDatabase

val values = ContentValues()

values.put(COLUMN\_FIRST\_NAME, user.firstName)

values.put(COLUMN\_LAST\_NAME, user.lastName)

values.put(COLUMN\_EMAIL, user.email)

values.put(COLUMN\_PASSWORD, user.password)

db.insert(TABLE\_NAME, null, values)

db.close()

}

@SuppressLint("Range")

fun getUserByUsername(username: String): User? {

val db = readableDatabase

val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_FIRST\_NAME = ?", arrayOf(username))

var user: User? = null

if (cursor.moveToFirst()) {

user = User(

id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)),

lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)),

email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)),

password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)),

)

}

cursor.close()

db.close()

return user

}

@SuppressLint("Range")

fun getUserById(id: Int): User? {

val db = readableDatabase

val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_ID = ?", arrayOf(id.toString()))

var user: User? = null

if (cursor.moveToFirst()) {

user = User(

id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)),

lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)),

email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)),

password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)),

)

}

cursor.close()

db.close()

return user

}

@SuppressLint("Range")

fun getAllUsers(): List<User> {

val users = mutableListOf<User>()

val db = readableDatabase

val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME", null)

if (cursor.moveToFirst()) {

do {

val user = User(

id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)),

lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)),

email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)),

password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)),

)

users.add(user)

} while (cursor.moveToNext())

}

cursor.close()

db.close()

return users

}

}

Database 2

Step 1:

**Create Survey data class**

package com.example.surveyapplication

import androidx.room.ColumnInfo

import androidx.room.Entity

import androidx.room.PrimaryKey

@Entity(tableName = "survey\_table")

data class Survey(

@PrimaryKey(autoGenerate = true) val id: Int?,

@ColumnInfo(name = "name") val name: String?,

@ColumnInfo(name = "age") val age: String?,

@ColumnInfo(name = "mobile\_number") val mobileNumber: String?,

@ColumnInfo(name = "gender") val gender: String?,

@ColumnInfo(name = "diabetics") val diabetics: String?,

)

Step 2:

**Create SurveyDao interface**

package com.example.surveyapplication

import androidx.room.\*

@Dao

interface SurveyDao {

@Query("SELECT \* FROM survey\_table WHERE age = :age")

suspend fun getUserByAge(age: String): Survey?

@Insert(onConflict = OnConflictStrategy.REPLACE)

suspend fun insertSurvey(survey: Survey)

@Update

suspend fun updateSurvey(survey: Survey)

@Delete

suspend fun deleteSurvey(survey: Survey)

}

Step 3:

**Create SurveyDatabase class**

package com.example.surveyapplication

import android.content.Context

import androidx.room.Database

import androidx.room.Room

import androidx.room.RoomDatabase

@Database(entities = [Survey::class], version = 1)

abstract class SurveyDatabase : RoomDatabase() {

abstract fun surveyDao(): SurveyDao

companion object {

@Volatile

private var instance: SurveyDatabase? = null

fun getDatabase(context: Context): SurveyDatabase {

return instance ?: synchronized(this) {

val newInstance = Room.databaseBuilder(

context.applicationContext,

SurveyDatabase::class.java,

"user\_database"

).build()

instance = newInstance

newInstance

}

}

}

}

Step 4:

**Create SurveyDatabaseHelper class**

package com.example.surveyapplication

import android.annotation.SuppressLint

import android.content.ContentValues

import android.content.Context

import android.database.Cursor

import android.database.sqlite.SQLiteDatabase

import android.database.sqlite.SQLiteOpenHelper

class SurveyDatabaseHelper(context: Context) :

SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) {

companion object {

private const val DATABASE\_VERSION = 1

private const val DATABASE\_NAME = "SurveyDatabase.db"

private const val TABLE\_NAME = "survey\_table"

private const val COLUMN\_ID = "id"

private const val COLUMN\_NAME = "name"

private const val COLUMN\_AGE = "age"

private const val COLUMN\_MOBILE\_NUMBER= "mobile\_number"

private const val COLUMN\_GENDER = "gender"

private const val COLUMN\_DIABETICS = "diabetics"

}

override fun onCreate(db: SQLiteDatabase?) {

val createTable = "CREATE TABLE $TABLE\_NAME (" +

"$COLUMN\_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +

"$COLUMN\_NAME TEXT, " +

"$COLUMN\_AGE TEXT, " +

"$COLUMN\_MOBILE\_NUMBER TEXT, " +

"$COLUMN\_GENDER TEXT," +

"$COLUMN\_DIABETICS TEXT" +

")"

db?.execSQL(createTable)

}

override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {

db?.execSQL("DROP TABLE IF EXISTS $TABLE\_NAME")

onCreate(db)

}

fun insertSurvey(survey: Survey) {

val db = writableDatabase

val values = ContentValues()

values.put(COLUMN\_NAME, survey.name)

values.put(COLUMN\_AGE, survey.age)

values.put(COLUMN\_MOBILE\_NUMBER, survey.mobileNumber)

values.put(COLUMN\_GENDER, survey.gender)

values.put(COLUMN\_DIABETICS, survey.diabetics)

db.insert(TABLE\_NAME, null, values)

db.close()

}

@SuppressLint("Range")

fun getSurveyByAge(age: String): Survey? {

val db = readableDatabase

val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_AGE = ?", arrayOf(age))

var survey: Survey? = null

if (cursor.moveToFirst()) {

survey = Survey(

id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

name = cursor.getString(cursor.getColumnIndex(COLUMN\_NAME)),

age = cursor.getString(cursor.getColumnIndex(COLUMN\_AGE)),

mobileNumber = cursor.getString(cursor.getColumnIndex(COLUMN\_MOBILE\_NUMBER)),

gender = cursor.getString(cursor.getColumnIndex(COLUMN\_GENDER)),

diabetics = cursor.getString(cursor.getColumnIndex(COLUMN\_DIABETICS)),

)

}

cursor.close()

db.close()

return survey

}

@SuppressLint("Range")

fun getSurveyById(id: Int): Survey? {

val db = readableDatabase

val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_ID = ?", arrayOf(id.toString()))

var survey: Survey? = null

if (cursor.moveToFirst()) {

survey = Survey(

id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

name = cursor.getString(cursor.getColumnIndex(COLUMN\_NAME)),

age = cursor.getString(cursor.getColumnIndex(COLUMN\_AGE)),

mobileNumber = cursor.getString(cursor.getColumnIndex(COLUMN\_MOBILE\_NUMBER)),

gender = cursor.getString(cursor.getColumnIndex(COLUMN\_GENDER)),

diabetics = cursor.getString(cursor.getColumnIndex(COLUMN\_DIABETICS)),

)

}

cursor.close()

db.close()

return survey

}

@SuppressLint("Range")

fun getAllSurveys(): List<Survey> {

val surveys = mutableListOf<Survey>()

val db = readableDatabase

val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME", null)

if (cursor.moveToFirst()) {

do {

val survey = Survey(

cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

cursor.getString(cursor.getColumnIndex(COLUMN\_NAME)),

cursor.getString(cursor.getColumnIndex(COLUMN\_AGE)),

cursor.getString(cursor.getColumnIndex(COLUMN\_MOBILE\_NUMBER)),

cursor.getString(cursor.getColumnIndex(COLUMN\_GENDER)),

cursor.getString(cursor.getColumnIndex(COLUMN\_DIABETICS))

)

surveys.add(survey)

} while (cursor.moveToNext())

}

cursor.close()

db.close()

return surveys

}

}

Building application UI and connecting to database

Step 1:

Creating LoginActivity.kt with database

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.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)

}

Step 2:

Creating RegisterActivity.kt with database

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.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)

}

Step 3:

Creating MainActivity.kt file

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

)

}

}

}

}

Step 4:

Creating AdminActivity.kt file

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}")

}

}

}

}

}

}

Modifying AndroidManifest.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>