**APPENDIX**

User.kt

|  |
| --- |
|  |
|  |
|  | Import android.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?, |
|  |  |
|  | ) |
|  |  |

UserDae.kt

|  |
| --- |
|  |
|  |  |
|  | Import android.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) |
|  | } |

UserDatabase.kt

|  |
| --- |
|  |
|  | 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 |
|  | } |
|  | } |
|  | } |
|  | } |

UserDatabase.kt

|  |
| --- |
|  |
|  | 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 |
|  | } |
|  | } |
|  | } |
|  | } |

UserDatabaseHelper.kt

|  |
| --- |
|  |
|  | 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" |
|  | }  UserDatabaseDao.kt   |  | | --- | | import androidx.room.\* | |  |  | |  | @Dao | |  | interface EmailDao { | |  |  | |  | @Query("SELECT \* FROM email\_table WHERE subject= :subject") | |  | suspend fun getOrderBySubject(subject: String): Email? | |  |  | |  | @Insert(onConflict = OnConflictStrategy.REPLACE) | |  | suspend fun insertEmail(email: Email) | |  |  | |  | @Update | |  | suspend fun updateEmail(email: Email) | |  |  | |  | @Delete | |  | suspend fun deleteEmail(email: Email) | |  | } | |
|  |  |
|  | 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 |
|  | } |
|  |  |
|  | } |

Email.kt

|  |
| --- |
|  |
|  | Import android.room.ColumnInfo  import androidx.room.Entity |
|  | import androidx.room.PrimaryKey |
|  |  |
|  | @Entity(tableName = "email\_table") |
|  | data class Email( |
|  | @PrimaryKey(autoGenerate = true) val id: Int?, |
|  | @ColumnInfo(name = "receiver\_mail") val recevierMail: String?, |
|  | @ColumnInfo(name = "subject") val subject: String?, |
|  | @ColumnInfo(name = "body") val body: String?, |
|  | ) |

EmailDao.kt

|  |
| --- |
|  |
|  |  |
|  | Import android.room.\*  @Dao |
|  | interface EmailDao { |
|  |  |
|  | @Query("SELECT \* FROM email\_table WHERE subject= :subject") |
|  | suspend fun getOrderBySubject(subject: String): Email? |
|  |  |
|  | @Insert(onConflict = OnConflictStrategy.REPLACE) |
|  | suspend fun insertEmail(email: Email) |
|  |  |
|  | @Update |
|  | suspend fun updateEmail(email: Email) |
|  |  |
|  | @Delete |
|  | suspend fun deleteEmail(email: Email) |
|  | } |

EmailDatabase.kt

|  |
| --- |
|  |
|  | Import android.content.Context  import androidx.room.Database |
|  | import androidx.room.Room |
|  | import androidx.room.RoomDatabase |
|  |  |
|  | @Database(entities = [Email::class], version = 1) |
|  | abstract class EmailDatabase : RoomDatabase() { |
|  |  |
|  | abstract fun emailDao(): EmailDao |
|  |  |
|  | companion object { |
|  |  |
|  | @Volatile |
|  | private var instance: EmailDatabase? = null |
|  |  |
|  | fun getDatabase(context: Context): EmailDatabase { |
|  | return instance ?: synchronized(this) { |
|  | val newInstance = Room.databaseBuilder( |
|  | context.applicationContext, |
|  | EmailDatabase::class.java, |
|  | "email\_database" |
|  | ).build() |
|  | instance = newInstance |
|  | newInstance |
|  | } |
|  | } |
|  | } |
|  | } |

UserDatabaseHelper.kt

|  |
| --- |
|  |
|  | 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 |
|  | } |
|  |  |
|  | } |

LoginActivity.kt

|  |
| --- |
|  |
|  | 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.emailapplication.ui.theme.EmailApplicationTheme |
|  |  |
|  | 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.email\_login), contentDescription = "" |
|  | ) |
|  |  |
|  |  |
|  | Text( |
|  | fontSize = 36.sp, |
|  | fontWeight = FontWeight.ExtraBold, |
|  | fontFamily = FontFamily.Cursive, |
|  | 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() |
|  | } |
|  |  |
|  | } else { |
|  | error = "Please fill all fields" |
|  | } |
|  | }, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFd3e5ef)), |
|  | modifier = Modifier.padding(top = 16.dp) |
|  | ) { |
|  | Text(text = "Login") |
|  | } |
|  | Row { |
|  | TextButton(onClick = {context.startActivity( |
|  | Intent( |
|  | context, |
|  | RegisterActivity::class.java |
|  | ) |
|  | )} |
|  | ) |
|  | { Text(color = Color(0xFF31539a),text = "Sign up") } |
|  | TextButton(onClick = { |
|  | }) |
|  |  |
|  | { |
|  | Spacer(modifier = Modifier.width(60.dp)) |
|  | Text(color = Color(0xFF31539a),text = "Forget password?") |
|  | } |
|  | } |
|  | } |
|  | } |
|  | private fun startMainPage(context: Context) { |
|  | val intent = Intent(context, MainActivity::class.java) |
|  | ContextCompat.startActivity(context, intent, null) |
|  | } |

RegisterActivity.kt

|  |
| --- |
|  |
|  |
|  | 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.emailapplication.ui.theme.EmailApplicationTheme |
|  |  |
|  | 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.email\_signup), contentDescription = "", |
|  | modifier = Modifier.height(300.dp) |
|  | ) |
|  | Text( |
|  | fontSize = 36.sp, |
|  | fontWeight = FontWeight.ExtraBold, |
|  | fontFamily = FontFamily.Cursive, |
|  | 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" |
|  |  |
|  | context.startActivity( |
|  | Intent( |
|  | context, |
|  | LoginActivity::class.java |
|  | ) |
|  | ) |
|  |  |
|  | } else { |
|  | error = "Please fill all fields" |
|  | } |
|  | }, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFd3e5ef)), |
|  | 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(0xFF31539a),text = "Log in") |
|  | } |
|  | } |
|  | } |
|  | } |
|  | private fun startLoginActivity(context: Context) { |
|  | val intent = Intent(context, LoginActivity::class.java) |
|  | ContextCompat.startActivity(context, intent, null) |
|  | } |
|  |  |

MainActivity.kt

|  |
| --- |
|  |
|  | 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.Composable |
|  | 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.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 androidx.core.content.ContextCompat.startActivity |
|  | import com.example.emailapplication.ui.theme.EmailApplicationTheme |
|  |  |
|  | class MainActivity : ComponentActivity() { |
|  | override fun onCreate(savedInstanceState: Bundle?) { |
|  | super.onCreate(savedInstanceState) |
|  | setContent { |
|  |  |
|  | Surface( |
|  | modifier = Modifier.fillMaxSize().background(Color.White), |
|  | ) { |
|  | Email(this) |
|  | } |
|  |  |
|  | } |
|  | } |
|  | } |
|  |  |
|  | @Composable |
|  | fun Email(context: Context) { |
|  | Text( |
|  | text = "Home Screen", |
|  | modifier = Modifier.padding(top = 74.dp, start = 100.dp, bottom = 24.dp), |
|  | color = Color.Black, |
|  | fontWeight = FontWeight.Bold, |
|  | fontSize = 32.sp |
|  | ) |
|  |  |
|  | Column( |
|  | horizontalAlignment = Alignment.CenterHorizontally, |
|  | verticalArrangement = Arrangement.Center |
|  | ) { |
|  | Image( |
|  | painterResource(id = R.drawable.home\_screen), contentDescription = "" |
|  | ) |
|  |  |
|  |  |
|  | Button(onClick = { |
|  | context.startActivity( |
|  | Intent( |
|  | context, |
|  | SendMailActivity::class.java |
|  | ) |
|  | ) |
|  | }, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFadbef4)) |
|  | ) { |
|  | Text( |
|  | text = "Send Email", |
|  | modifier = Modifier.padding(10.dp), |
|  | color = Color.Black, |
|  | fontSize = 15.sp |
|  | ) |
|  | } |
|  |  |
|  | Spacer(modifier = Modifier.height(20.dp)) |
|  |  |
|  | Button(onClick = { |
|  | context.startActivity( |
|  | Intent( |
|  | context, |
|  | ViewMailActivity::class.java |
|  | ) |
|  | ) |
|  | }, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFadbef4)) |
|  | ) { |
|  | Text( |
|  | text = "View Emails", |
|  | modifier = Modifier.padding(10.dp), |
|  | color = Color.Black, |
|  | fontSize = 15.sp |
|  | ) |
|  | } |
|  |  |
|  |  |
|  | } |
|  | } |

SendMailActivity.kt

|  |
| --- |
|  |
|  | import android.annotation.SuppressLint  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.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.platform.LocalContext |
|  | import androidx.compose.ui.text.TextStyle |
|  | import androidx.compose.ui.text.font.FontWeight |
|  | 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.emailapplication.ui.theme.EmailApplicationTheme |
|  |  |
|  | class SendMailActivity : ComponentActivity() { |
|  | private lateinit var databaseHelper: EmailDatabaseHelper |
|  | @SuppressLint("UnusedMaterialScaffoldPaddingParameter") |
|  | override fun onCreate(savedInstanceState: Bundle?) { |
|  | super.onCreate(savedInstanceState) |
|  | databaseHelper = EmailDatabaseHelper(this) |
|  | setContent { |
|  |  |
|  | Scaffold( |
|  |  |
|  | topBar = { |
|  |  |
|  |  |
|  | TopAppBar(backgroundColor = Color(0xFFadbef4), modifier = Modifier.height(80.dp), |
|  |  |
|  |  |
|  | title = { |
|  |  |
|  |  |
|  | Text( |
|  |  |
|  |  |
|  | text = "Send Mail", |
|  | fontSize = 32.sp, |
|  | color = Color.Black, |
|  |  |
|  |  |
|  |  |
|  | modifier = Modifier.fillMaxWidth(), |
|  |  |
|  |  |
|  |  |
|  | textAlign = TextAlign.Center, |
|  | ) |
|  | } |
|  | ) |
|  | } |
|  | ) { |
|  |  |
|  |  |
|  | openEmailer(this,databaseHelper) |
|  | } |
|  | } |
|  | } |
|  | } |
|  | @Composable |
|  | fun openEmailer(context: Context, databaseHelper: EmailDatabaseHelper) { |
|  |  |
|  |  |
|  |  |
|  | var recevierMail by remember {mutableStateOf("") } |
|  | var subject by remember {mutableStateOf("") } |
|  | var body by remember {mutableStateOf("") } |
|  | var error by remember { mutableStateOf("") } |
|  |  |
|  |  |
|  |  |
|  | val ctx = LocalContext.current |
|  |  |
|  |  |
|  | Column( |
|  |  |
|  |  |
|  |  |
|  | modifier = Modifier |
|  | .fillMaxSize() |
|  | .padding(top = 55.dp, bottom = 25.dp, start = 25.dp, end = 25.dp), |
|  | horizontalAlignment = Alignment.Start |
|  | ) { |
|  |  |
|  |  |
|  |  |
|  | Text(text = "Receiver Email-Id", |
|  | fontWeight = FontWeight.Bold, |
|  | fontSize = 16.sp) |
|  | TextField( |
|  |  |
|  |  |
|  | value = recevierMail, |
|  |  |
|  |  |
|  |  |
|  | onValueChange = { recevierMail = it }, |
|  |  |
|  |  |
|  | label = { Text(text = "Email address") }, |
|  | placeholder = { Text(text = "abc@gmail.com") }, |
|  |  |
|  |  |
|  |  |
|  | modifier = Modifier |
|  | .padding(16.dp) |
|  | .fillMaxWidth(), |
|  |  |
|  |  |
|  |  |
|  | textStyle = TextStyle(color = Color.Black, fontSize = 15.sp), |
|  |  |
|  |  |
|  |  |
|  | singleLine = true, |
|  | ) |
|  |  |
|  | Spacer(modifier = Modifier.height(10.dp)) |
|  |  |
|  | Text(text = "Mail Subject", |
|  | fontWeight = FontWeight.Bold, |
|  | fontSize = 16.sp) |
|  |  |
|  | TextField( |
|  |  |
|  |  |
|  | value = subject, |
|  |  |
|  |  |
|  |  |
|  | onValueChange = { subject = it }, |
|  |  |
|  |  |
|  | placeholder = { Text(text = "Subject") }, |
|  |  |
|  |  |
|  |  |
|  | modifier = Modifier |
|  | .padding(16.dp) |
|  | .fillMaxWidth(), |
|  |  |
|  |  |
|  |  |
|  | textStyle = TextStyle(color = Color.Black, fontSize = 15.sp), |
|  |  |
|  |  |
|  |  |
|  | singleLine = true, |
|  | ) |
|  |  |
|  |  |
|  | Spacer(modifier = Modifier.height(10.dp)) |
|  |  |
|  | Text(text = "Mail Body", |
|  | fontWeight = FontWeight.Bold, |
|  | fontSize = 16.sp) |
|  |  |
|  | TextField( |
|  |  |
|  |  |
|  | value = body, |
|  |  |
|  |  |
|  |  |
|  | onValueChange = { body = it }, |
|  |  |
|  |  |
|  | placeholder = { Text(text = "Body") }, |
|  |  |
|  |  |
|  |  |
|  | modifier = Modifier |
|  | .padding(16.dp) |
|  | .fillMaxWidth(), |
|  |  |
|  |  |
|  |  |
|  | textStyle = TextStyle(color = Color.Black, fontSize = 15.sp), |
|  |  |
|  |  |
|  |  |
|  | singleLine = true, |
|  | ) |
|  |  |
|  |  |
|  | Spacer(modifier = Modifier.height(20.dp)) |
|  |  |
|  |  |
|  |  |
|  | Button(onClick = { |
|  |  |
|  | if( recevierMail.isNotEmpty() && subject.isNotEmpty() && body.isNotEmpty()) { |
|  | val email = Email( |
|  | id = null, |
|  | recevierMail = recevierMail, |
|  | subject = subject, |
|  | body = body |
|  |  |
|  | ) |
|  | databaseHelper.insertEmail(email) |
|  | error = "Mail Saved" |
|  | } else { |
|  | error = "Please fill all fields" |
|  | } |
|  |  |
|  |  |
|  |  |
|  | val i = Intent(Intent.ACTION\_SEND) |
|  |  |
|  |  |
|  |  |
|  | val emailAddress = arrayOf(recevierMail) |
|  | i.putExtra(Intent.EXTRA\_EMAIL,emailAddress) |
|  | i.putExtra(Intent.EXTRA\_SUBJECT,subject) |
|  | i.putExtra(Intent.EXTRA\_TEXT,body) |
|  |  |
|  |  |
|  |  |
|  | i.setType("message/rfc822") |
|  |  |
|  |  |
|  | ctx.startActivity(Intent.createChooser(i,"Choose an Email client : ")) |
|  |  |
|  | }, |
|  | colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFd3e5ef)) |
|  | ) { |
|  |  |
|  | Text( |
|  |  |
|  |  |
|  | text = "Send Email", |
|  | modifier = Modifier.padding(10.dp), |
|  | color = Color.Black, |
|  | fontSize = 15.sp |
|  | ) |
|  | } |
|  | } |
|  | } |

ViewMailActivity.kt

|  |
| --- |
|  |
|  |
|  | Import android.annotation.SupperssLint  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.layout.R |
|  | import androidx.compose.foundation.lazy.LazyColumn |
|  | import androidx.compose.foundation.lazy.LazyRow |
|  | import androidx.compose.foundation.lazy.items |
|  | import androidx.compose.material.\* |
|  | 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.text.font.FontWeight |
|  | 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.emailapplication.ui.theme.EmailApplicationTheme |
|  |  |
|  | class ViewMailActivity : ComponentActivity() { |
|  | private lateinit var emailDatabaseHelper: EmailDatabaseHelper |
|  | @SuppressLint("UnusedMaterialScaffoldPaddingParameter") |
|  | override fun onCreate(savedInstanceState: Bundle?) { |
|  | super.onCreate(savedInstanceState) |
|  | emailDatabaseHelper = EmailDatabaseHelper(this) |
|  | setContent { |
|  |  |
|  | Scaffold( |
|  |  |
|  | topBar = { |
|  |  |
|  |  |
|  | TopAppBar(backgroundColor = Color(0xFFadbef4), modifier = Modifier.height(80.dp), |
|  |  |
|  |  |
|  | title = { |
|  |  |
|  |  |
|  | Text( |
|  |  |
|  |  |
|  | text = "View Mails", |
|  | fontSize = 32.sp, |
|  | color = Color.Black, |
|  |  |
|  |  |
|  |  |
|  | modifier = Modifier.fillMaxWidth(), |
|  |  |
|  |  |
|  |  |
|  | textAlign = TextAlign.Center, |
|  | ) |
|  | } |
|  | ) |
|  | } |
|  | ) { |
|  | val data = emailDatabaseHelper.getAllEmails(); |
|  | Log.d("swathi", data.toString()) |
|  | val email = emailDatabaseHelper.getAllEmails() |
|  | ListListScopeSample(email) |
|  | } |
|  | } |
|  | } |
|  | } |
|  | @Composable |
|  | fun ListListScopeSample(email: List<Email>) { |
|  | LazyRow( |
|  | modifier = Modifier |
|  | .fillMaxSize(), |
|  | horizontalArrangement = Arrangement.SpaceBetween |
|  | ) { |
|  | item { |
|  |  |
|  | LazyColumn { |
|  | items(email) { email -> |
|  | Column( |
|  | modifier = Modifier.padding( |
|  | top = 16.dp, |
|  | start = 48.dp, |
|  | bottom = 20.dp |
|  | ) |
|  | ) { |
|  | Text("Receiver\_Mail: ${email.recevierMail}", fontWeight = FontWeight.Bold) |
|  | Text("Subject: ${email.subject}") |
|  | Text("Body: ${email.body}") |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | } |
|  | } |
|  |  |

Andoridmanifest.xml

|  |
| --- |
|  |
|  | <xml version=”1.0” encoding=”jtf-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.EmailApplication" |
|  | tools:targetApi="31" > |
|  | <activity |
|  | android:name=".RegisterActivity" |
|  | android:exported="false" |
|  | android:label="@string/title\_activity\_register" |
|  | android:theme="@style/Theme.EmailApplication" /> |
|  | <activity |
|  | android:name=".MainActivity" |
|  | android:exported="false" |
|  | android:label="MainActivity" |
|  | android:theme="@style/Theme.EmailApplication" /> |
|  | <activity |
|  | android:name=".ViewMailActivity" |
|  | android:exported="false" |
|  | android:label="@string/title\_activity\_view\_mail" |
|  | android:theme="@style/Theme.EmailApplication" /> |
|  | <activity |
|  | android:name=".SendMailActivity" |
|  | android:exported="false" |
|  | android:label="@string/title\_activity\_send\_mail" |
|  | android:theme="@style/Theme.EmailApplication" /> |
|  | <activity |
|  | android:name=".LoginActivity" |
|  | android:exported="true" |
|  | android:label="@string/app\_name" |
|  | android:theme="@style/Theme.EmailApplication" > |
|  | <intent-filter> |
|  | <action android:name="android.intent.action.MAIN" /> |
|  |  |
|  | <category android:name="android.intent.category.LAUNCHER" /> |
|  | </intent-filter> |
|  | </activity> |
|  | </application> |
|  |  |
|  | </manifest> |