**Assignment 1 *Android Basics with Compose and BackupManager***

COMP2430-Mobile Computing Technologies

Department of Computer Science

Winter 2025

**Group 12**

**Group Members:-**

1. Sanskriti Dabhade (1225131)
2. Sweety Das (1238456)
3. Kirtan Sukhadiya (1222983)
4. Julia Petterle (1185004)

## **Introduction:-** This report documents the solutions to Assignment 1 of COMP2430, which involves building a business card app using Jetpack Compose, configuring full backups in an Android application, and using Android Debug Bridge (ADB) for backup testing. The tasks focus on understanding basic Android UI development and backup mechanisms in Android Studio.

## **Presentation Video:-**

[Business card app.mp4](https://drive.google.com/file/d/1oXCeHErx9Y0V7M3U7deUDGk_V3WTEpiO/view?usp=drive_link)

## **Task 1: Android Basics with Compose 1a) Business Card App Development**

## **Methods:**

The application follows a modular approach, where different UI components are separated into composable functions. Key methods include:

* MainActivity: Serves as the entry point of the application. It initializes the application theme and sets up the content using a Scaffold layout, which is a composable structure providing top-level layout functionality, including support for app bars, floating action buttons, and padding management. The enableEdgeToEdge() function is called to ensure that the content extends into the system bars, creating a more immersive experience.
* MainScreen: A composable function displaying the user's name, title, and logo. It uses a Box container and nested Column and Row layouts for vertical and horizontal alignment. The Box composable acts as a wrapper, providing flexibility in positioning its child elements. Inside the Box, the Column arranges the logo, name, and title vertically, ensuring that they are centered on the screen.
* ContactBox: This composable displays the contact details with icons for phone, share, and email. It utilizes a multi-level layout structure with nested Columns and Rows to align icons and text. Each contact detail is presented in a separate Row, with an icon on the left and the corresponding text on the right. The contentAlignment property of the outer Box ensures that the contact information is positioned at the bottom of the screen.
* Icon Functions: Each icon is rendered using Jetpack Compose's Icon component with prebuilt Material Icons. These functions are small, single-purpose composables enhancing code modularity and readability. They allow easy updates to individual icons without affecting other parts of the UI.
* Theming and Layout Management: The application applies a custom color scheme through the BusinessCardAppTheme, ensuring visual consistency. The primary background color is a light green shade (0xFFbfe3d0), complemented by a dark blue logo background (0xFF222b42) and green icon tint (0xFF268755). These colors were chosen to create a professional yet approachable aesthetic.

## **Testing:**

Testing of the Business Card App was conducted using:

* Jetpack Compose Preview: Ensuring UI components render correctly in Android Studio.
* Manual Testing on Emulator and Physical Device: Verifying responsiveness, alignment, and usability.
* Edge-to-Edge Display Verification: Checking the seamless extension of UI elements across screen edges.

## **Detailed Results:**

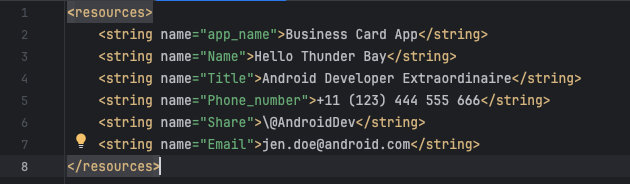
The Business Card App successfully renders a structured digital card with distinct sections for user details and contact information. The implementation of Jetpack Compose enhances performance, ensuring a smooth and responsive experience. The UI components are modular, making future modifications easier. The application adheres to modern Android development principles, leveraging Compose’s declarative UI approach.

The Scaffold component ensures a consistent layout, while Surface provides a themed background. The Row and Column structures effectively align text and icons, ensuring clarity in displaying contact information. The business card’s logo is contained within a Box with appropriate size constraints.

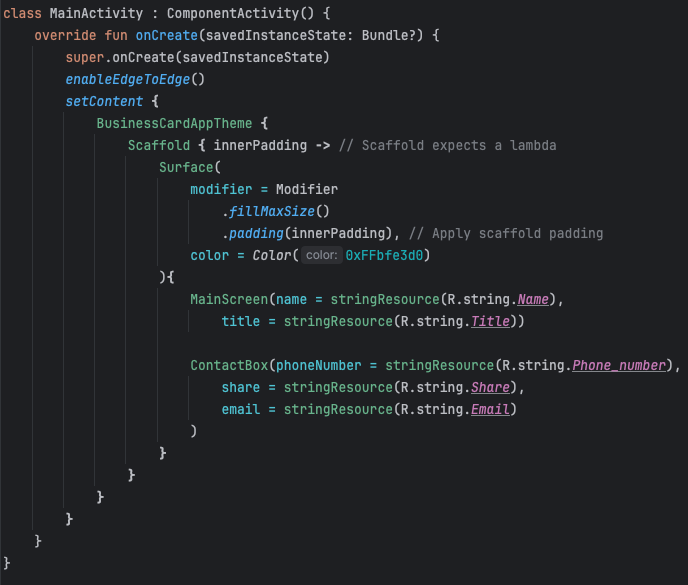
Overall, the Business Card App functions as intended, providing an elegant and structured display of user information using Jetpack Compose.

**Screenshot of the App Running:**

Modified code to display “Hello Thunder Bay”

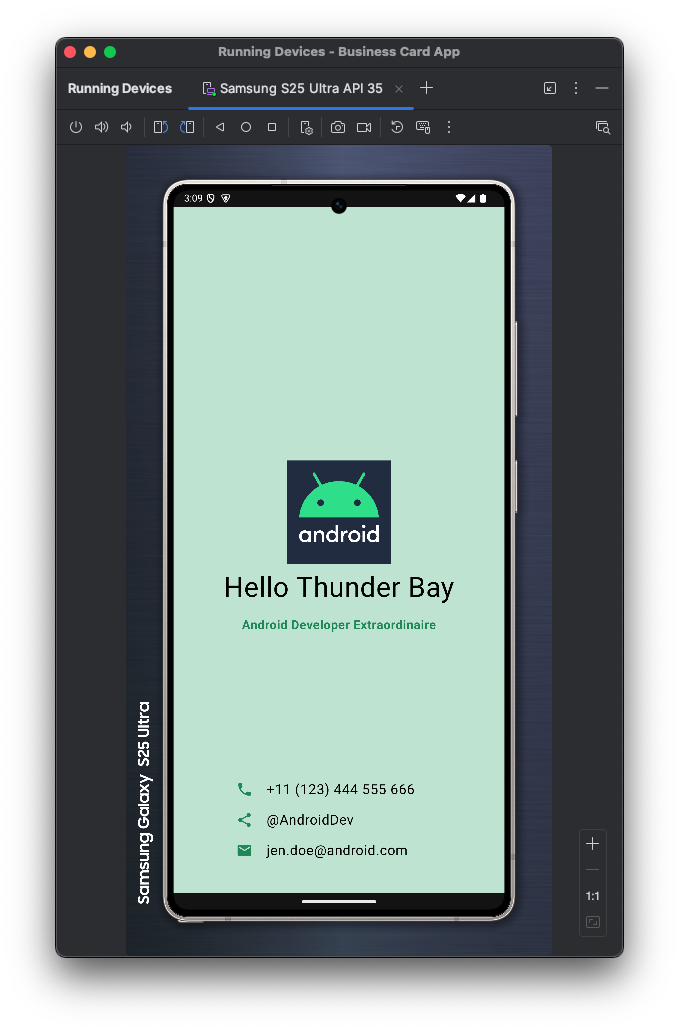


String.xml



MainActivity.kt

**Screenshot of the app with modified output:**



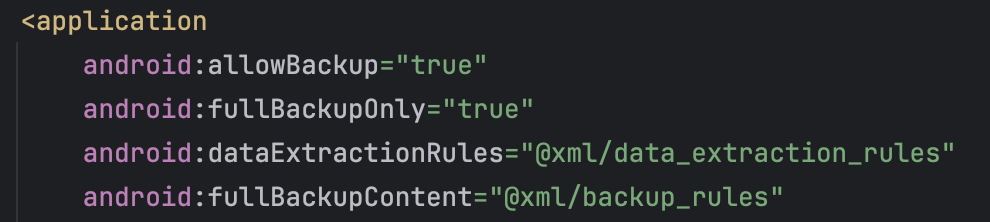
**Video Demonstration:**

[Question1.mov](https://drive.google.com/file/d/1Z-jrivzQM3cTtr8dVB7W_WaLF7Rcgd1_/view?usp=drive_link)

## **Task 2: Perform a full backup of all apps using Android's Backup Service**

**2a)**

Enabled full backups in AndroidManifest.xml file…



**Instructions to backup data**

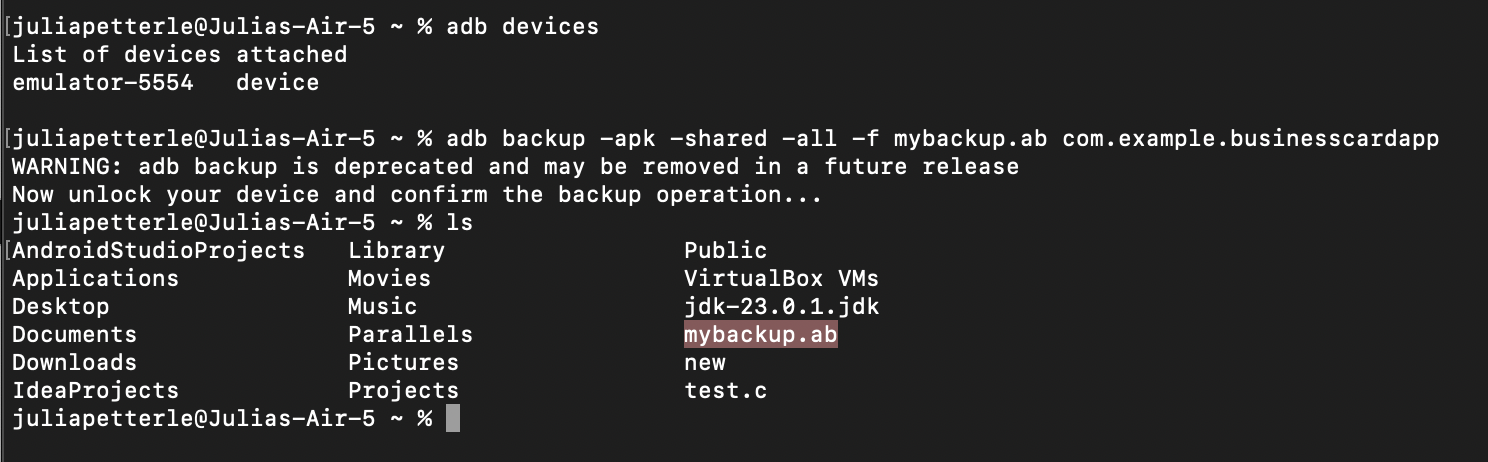
(Applicable for MacOS)

1. Connect your Android emulator or device. Ensure it's recognized by running the following command in the Terminal window:

adb devices

1. Execute the Backup Command:

adb backup -apk -shared -all -f mybackup.ab com.example.businesscardapp

1. Checking if the backup worked:
2. Use ADB to uninstall the app from your device:

adb uninstall com.example.businesscardapp

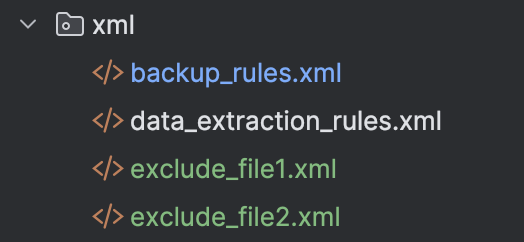
1. Reinstall the app from Android Studio by running it again.
2. Execute the Restore Command:

adb restore mybackup.ab

1. Check if the app is present on the device and can be launched. Use the app to make sure all expected functionalities are working.

**2b)**

Added two resource files to my project named exclude\_file1.xml and exclude\_file2.xml…



Configured the XMLs for the backup manager to include all files in the backup except for exclude\_file1.xml and exclude\_file2.xml…



**Instructions to check if both files were excluded**

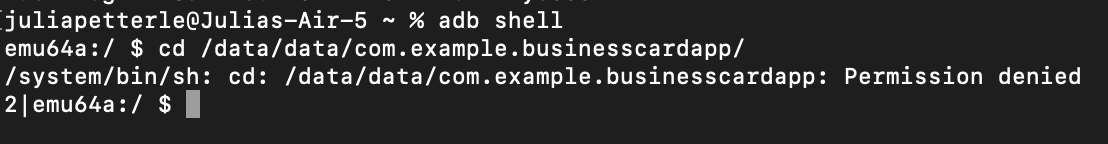
(First follow the steps from “Instructions to backup data”)

1. Access the App's Data Directory

./adb shell su cd /data/data/com.example.businesscardapp/files ls

* This command lists the contents of the files directory, where you can check for the absence of *exclude\_file1.xml* and *exclude\_file2.xml*.

We were not able to verify that both includes and excludes worked (or did not work) correctly because when accessing the app’s data using adb shell we got an error that said “Permission denied.” This is due to Android's security model, which restricts access to app-specific directories to only the app itself and the system. (SEE SCREENSHOT BELOW)



## **NOTE:-**

Due to the absence of an Android device and the limitations of the emulator, we were unable to demonstrate the backup that excluded the necessary files.