Lab 2: Ionic/Angular Testcase for Android

YT0798 Cross Development, Thomas More

Gilles Van pellicom

November 19, 2025

Contents

Notes on the Process

Reproducability

- **Development & Compilation environment:**
  - **Platform:** macOS Tahoe 26.1
  - **Hardware:** M3 Pro, 36 GB LPDDR5
  - **IDE:** IntelliJ 2025.2.4
  - **Emulator:** Android Studio 2025.2.1 patch 1 ”Otter”
- **Testing environment:**
  - **Platform:** Android 15 (kernel 6.1.124android14-11-gf044a08b82f5-ab13293802)
  - **Hardware:** MediaTek Dimensity 7300 Pro 5G, 8 GB LPDDR5

Chapter 1

Testcase

1.1 Requirements and Fulfillment

The test case is an Ionic port of the Lab 1 calculator testcase. It meets the following criteria:

- **Standard Ionic components:** Implemented using `<ion-grid>`, `<ion-row>`, and `<ion-button>` for styling, inputs, etc.
- **Custom component using canvas:** Renders expression/result on a high-DPI-aware `<canvas>`.
- **Dialog for selecting an accent color:** Settings dialog includes `<input type="color">` for accent color selection.
- **File I/O for settings storage:** Selected accent color stored persistently using `@capacitor/preferences`.
- **Non-default Capacitor plugin:** Utilised `@capacitor/haptics` for haptic feedback on button presses.
- **Two pages with shared state:** `CalculatorService` and `SettingsService` are shared between pages `tab1`, `tab2` and settings modal. These services are located under `/src/app/services`

1.2 Portability Considerations

Since the project was built in Ionic, the project was built with portability from the ground up. On my development machine the project required no special considerations except for the installation of the correct android/iOS toolchains for portability and successful cross-compilation. Some highlights of using Ionic for portability:

- **Capacitor plugins:** Utilized capacitor plugins for access to OS function in an OS-agnostic manner.
- **OS-agnostic color picker:** Leveraged Ionic's `<input type="color">` to handle color selection, allowing Ionic to manage platform-specific details seamlessly.

Chapter 2

Toolchain Setup and Workflow

2.1 Prerequisites

Ionic requires Node.js. Install using macOS package manager brew:

```
brew install node
```

Install ionic CLI:

```
npm install -g @ionic/cli
```

Verify installation:

```
ionic --version
```

2.2 Project Initialization

Create an Ionic Angular project in the current folder:

```
ionic start . --type=angular
```

Follow prompts. Select default behavior and `tabs` starter project.

2.3 Running the Project in Browser

Start the development server to verify the project works:

```
ionic serve
```

The `tabs` app will open in the default browser

2.4 Running the Project on Device

2.4.1 Android

Run the following command to bundle the project:

```
ionic build
```

The location of Android studio must be in the PATH for the project to sync with Android Studio:

```
export CAPACITOR_ANDROID_STUDIO_PATH="/Users/gvp/Applications/Android Studio.app"
source ~/.zshrc
```

Now we can sync with Android Studio to emulate the app:

```
npx cap sync android
npx cap open android
```

Once in Android Studio, select an emulator to test the app. If a physical device is required, e.g. to test features such as haptics, an Android debug-enabled device can paired like so:

Make sure Android Debug Bridge (ADB) is installed. Navigate to "Android Wireless Debugging" in the device settings and find the device's IP and port. Take note of the pairing code.

```
adb pair 192.168.0.227:44499
```

Continue by entering the pairing code when prompted to complete the pair. The device will now show up as a target in Android Studio. Recompile the project with the new device as target to test the app.

2.4.2 iOS

The basic process remain the same compared to Android. Instead of Android Studio we use Xcode, which is streamlined for use on macOS. No additional setup should be required.

```
ionic build
npx cap sync ios
npx cap open ios
```

Now the app can be tested for iOS

2.5 Outcome

Aside from the localhost-served website, a working native-Android build of the testcase was achieved utilising Ionic/Angular, and was tested on emulator and on device via ADB.

A Screenshot of the successful deployment on device is provided in the `screenshots` folder.