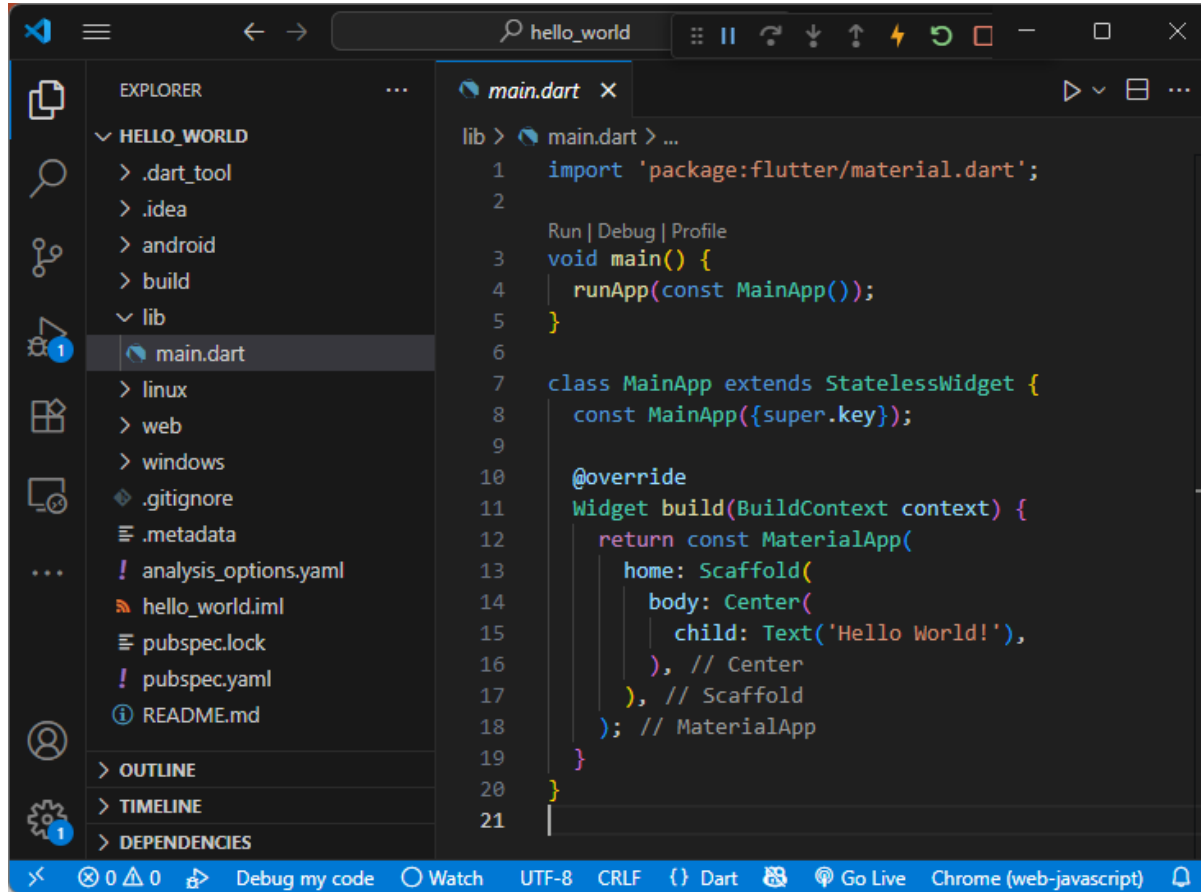


# Hello World in Flutter



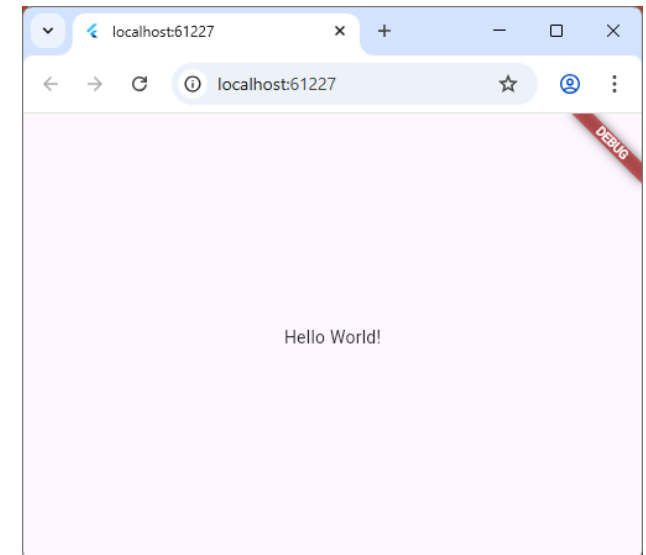
On Android



```
lib > main.dart > ...
1  import 'package:flutter/material.dart';
2
3  Run | Debug | Profile
4  void main() {
5    runApp(const MainApp());
6  }
7
8  class MainApp extends StatelessWidget {
9    const MainApp({super.key});
10
11    @override
12    Widget build(BuildContext context) {
13      return const MaterialApp(
14        home: Scaffold(
15          body: Center(
16            child: Text('Hello World!'),
17          ), // Center
18        ), // Scaffold
19      ); // MaterialApp
20    }
21  }
```



On Chrome

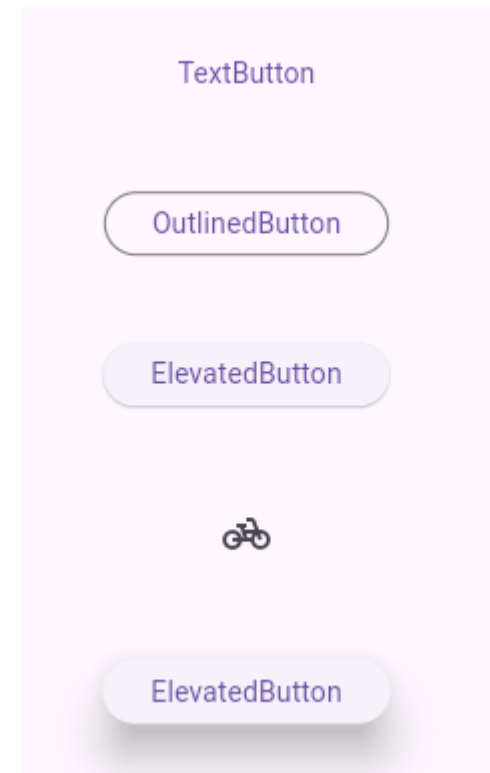


# Button types



```
// more button types:
TextButton(onPressed: onPressed, child: Text("TextButton")),
OutlinedButton(
  onPressed: onPressed,
  child: Text("OutlinedButton"),
), // OutlinedButton
ElevatedButton(
  onPressed: onPressed,
  child: Text("ElevatedButton"),
), // ElevatedButton
IconButton(
  onPressed: onPressed,
  icon: Icon(Icons.pedal_bike),
), // IconButton

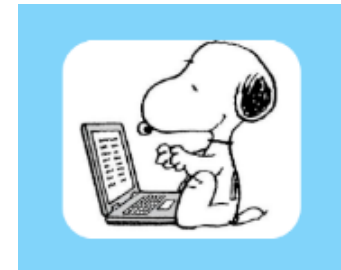
// solution for last exercise: Elevated button with big elevation
ElevatedButton(
  style: ElevatedButton.styleFrom(elevation: 20),
  onPressed: onPressed,
  child: Text("ElevatedButton"),
), // ElevatedButton
```



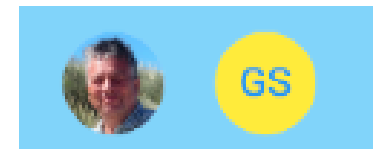
# Images



```
ClipRect(  
  borderRadius: BorderRadius.circular(20),  
  child: Image.asset("assets/images/snoopy_laptop.jpg",  
    width: 150)) // Image.asset // ClipRect
```



```
const Row(  
  children: [  
    SizedBox(width: 20),  
    CircleAvatar(  
      foregroundImage: AssetImage("assets/images/GS.jpg")), // CircleAvatar  
    SizedBox(width: 20),  
    CircleAvatar(  
      foregroundColor: Colors.blue,  
      backgroundColor: Colors.yellow,  
      child: Text("GS")), // CircleAvatar  
  ],  
), // Row
```








# Sliders, Checkboxes, Switches

```
Slider(  
  value: sliderValue,  
  min: -10,  
  max: 10,  
  // label is only shown during movement of the  
  // slider and when divisions is defined !  
  label: "$sliderValue",  
  divisions: 10,  
  onChanged: (value) {  
    setState(() {  
      sliderValue = value;  
    });  
  },  
) // Slider
```



```
Checkbox(  
  tristate: true,  
  activeColor: Colors.amber,  
  checkColor: Colors.green,  
  splashRadius: 50, // effect size when pressed  
  value: checkboxValue,  
  onChanged: (value) {  
    setState(() {  
      checkboxValue = value;  
      print("checkboxValue is $checkboxValue");  
    });  
  },  
) // Checkbox
```

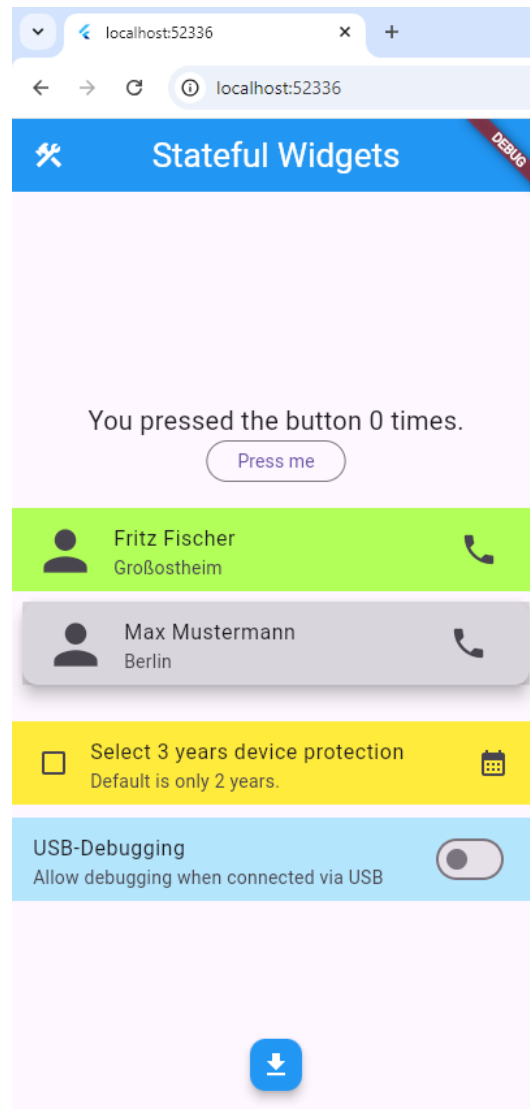
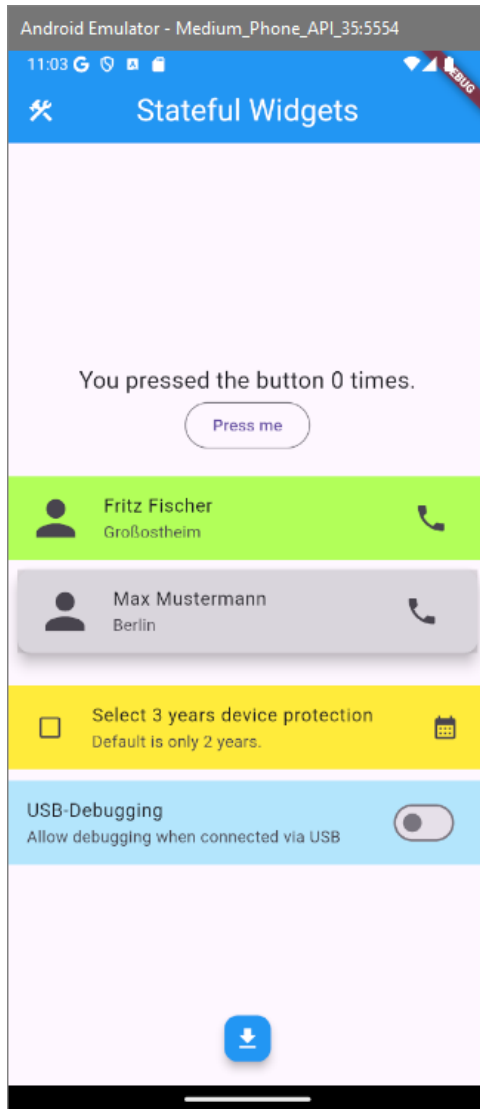
	true	
checkboxValue: false	false	
	null	

```
Row(  
  mainAxisAlignment: MainAxisAlignment.center,  
  children: [  
    Switch(  
      value: switchValue,  
      activeColor: Colors.green,  
      inactiveThumbColor: Colors.red,  
      onChanged: (value) {  
        setState(() {  
          switchValue = value;  
        });  
      },  
    ), // Switch  
    const SizedBox(width: 20),  
    Text(  
      switchValue ? "ON" : "OFF",  
      style: const TextStyle(fontSize: 22),  
    ), // Text  
  ],  
) // Row
```

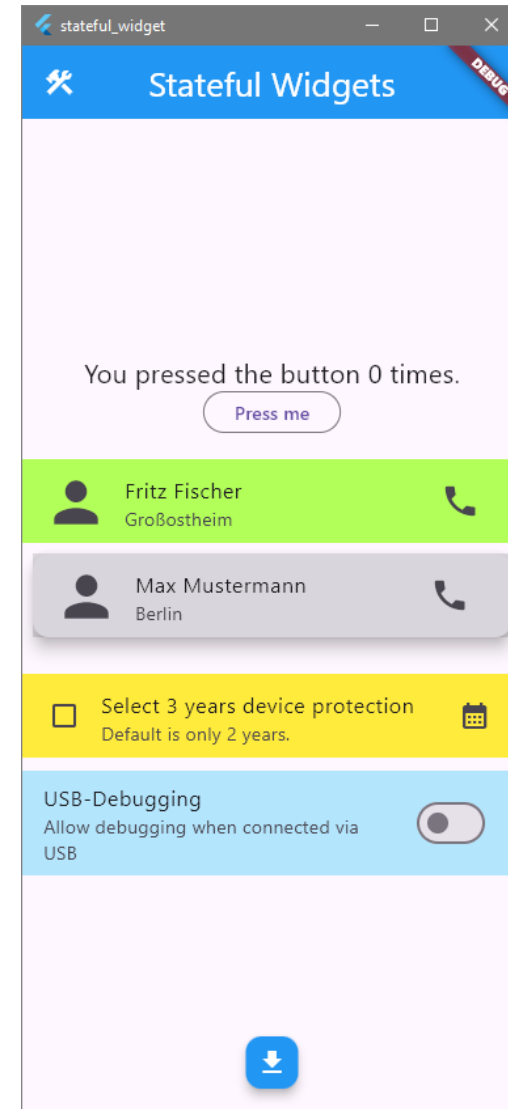




# Same UI on different devices



As Windows app:





# Dart – the programming language used in Flutter

- Built-In types in Dart (int, double, String, bool, dynamic)
- Nullable types and “sound null safety”
- Methods with positional or named parameters or both
- Classes with getters, setters, private members and derived classes
- Named and const constructors
- List and Map as generic types
- Asynchronous code execution with `async` / `await`



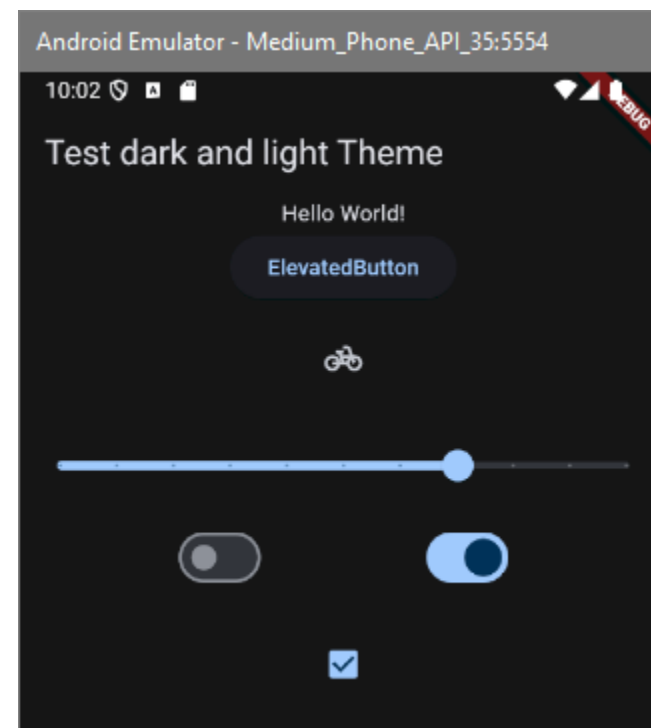
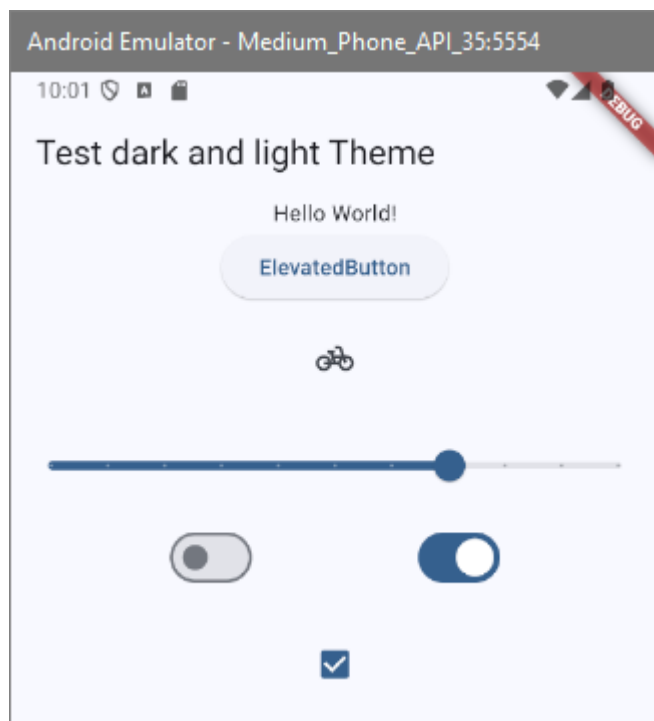
# Unit Tests and Test Coverage

The screenshot shows an IDE interface with the following components:

- TESTING Panel:** Displays test results. A filter bar at the top says "Filter (e.g. text, !exclude, @tag)". Below it, a summary shows "6/6" tests passed in "2.9s". The tests are grouped as follows:
  - test\location\_provider\_test.dart: 2/2 passed: 22ms
    - Aschaffenburg is a known location: 20ms
    - XXX is an unknown location: 2.0ms
  - test\weather\_in\_dart\_test.dart: 0/4 passed
    - no forecast when location is not set
    - no forecast when location was modified
    - Sydney time is before Aschaffenburg time
    - New York time is after Aschaffenburg time
  - test\weather\_provider\_test.dart: 4/4 passed: 534ms
- Source Code Editor:** Shows the `weather_provider.dart` file. The `fetchHourlyForecast` method is visible, starting with `Future<String> fetchHourlyForecast() async {`. The code includes HTTP requests, JSON decoding, and list manipulations. Line numbers 8 through 53 are visible on the left margin.
- Status Bar:** At the bottom, it shows "File 97% Covered" (highlighted with a red box), "Ln 51, Col 1", "Spaces: 2", "UTF-8", "CRLF", and "Dart".



# Light and Dark Theme

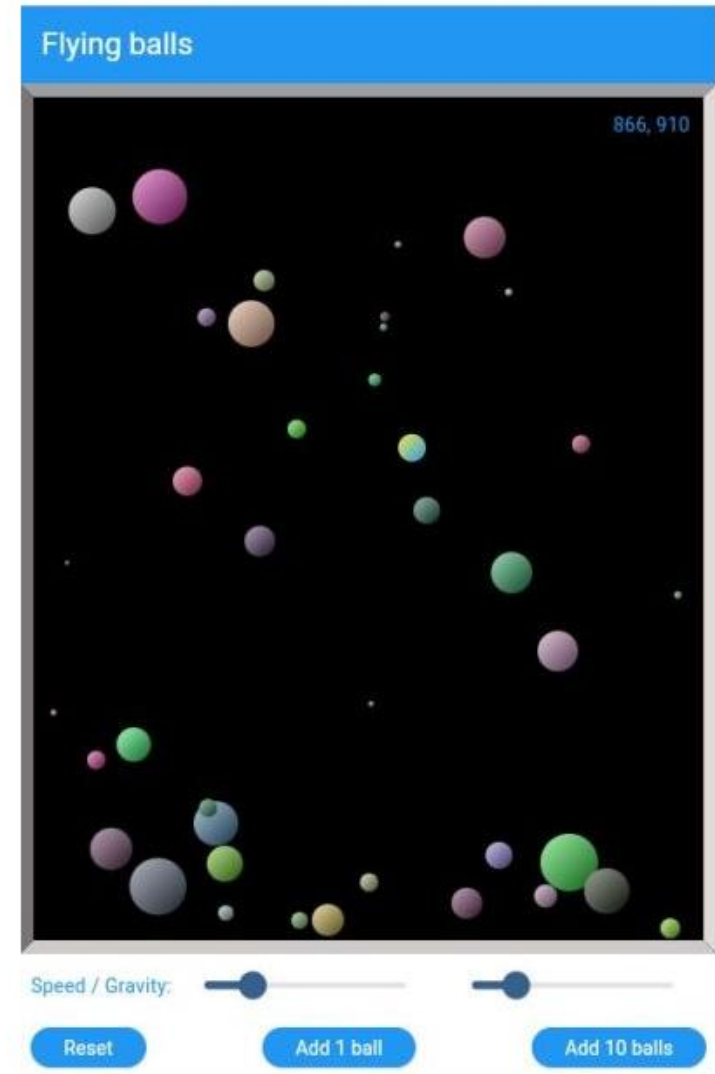




# Flying balls

- Stack widget to position other widgets
- Timers for animation
- Show or hide UI elements after user input
- Gradient colors
- Gesture detection

[https://fdg2425.github.io/web/flying\\_balls/](https://fdg2425.github.io/web/flying_balls/)



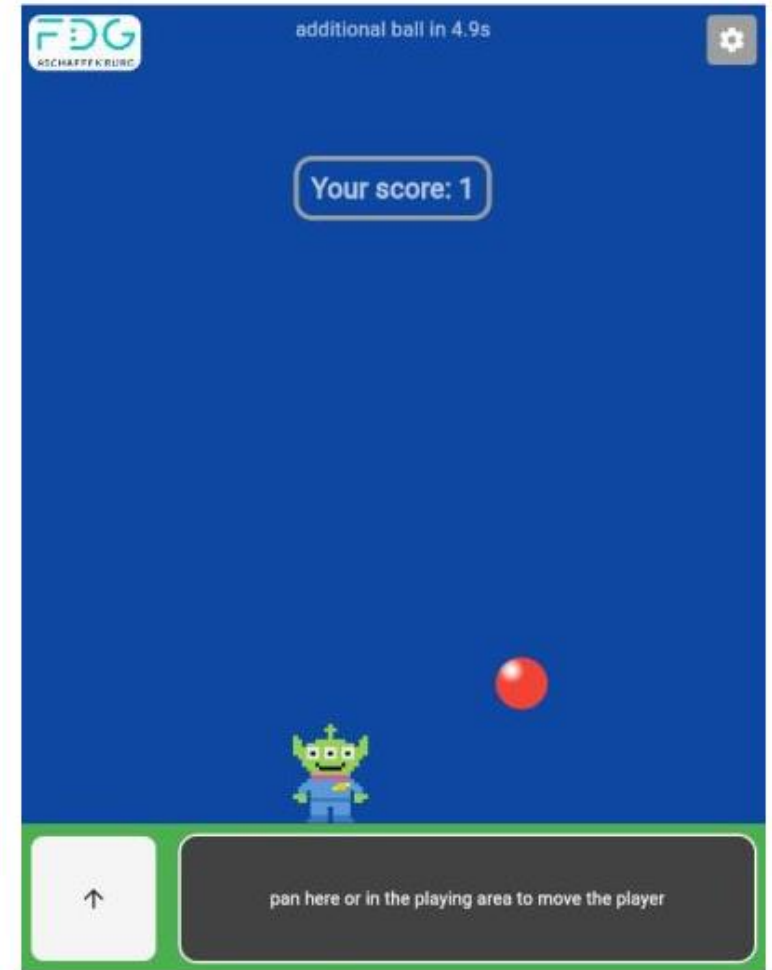
# Optional: Bubble Trouble

Mitch Koko implemented basic Flutter versions of several games like Tetris, Chess & Flappy Bird, see corresponding tutorials [on YouTube](#).

We enhanced his implementation of Bubble Trouble during our last Flutter training at FDG.

In case students are interested, we can have a look at that.

[https://fdg2425.github.io/webtest/bubble/naalnorth\\_branch\\_gs\\_ideas/](https://fdg2425.github.io/webtest/bubble/naalnorth_branch_gs_ideas/)



# Lux Meter

Make use of external flutter packages:

- **light** to read lux values from front camera
- **syncfusion\_flutter\_gauges** for the gauge
- **fl\_chart** to display line charts
- **shared\_preferences** to make settings persistent

[https://fdg2425.github.io/web/fdg\\_lux\\_meter/](https://fdg2425.github.io/web/fdg_lux_meter/)



# Open Meteo UI

Goal at the end of the training is to implement a full-fledged weather app without advertising.

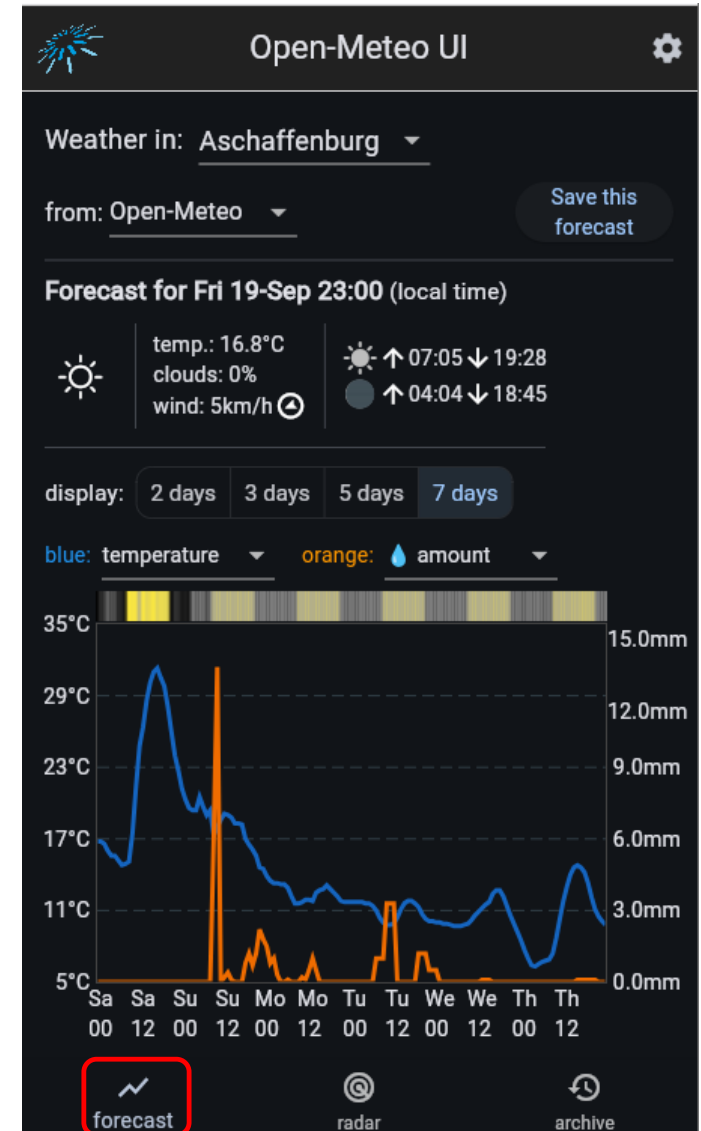
We will use weather data from [Open-Meteo](https://open-meteo.com/).

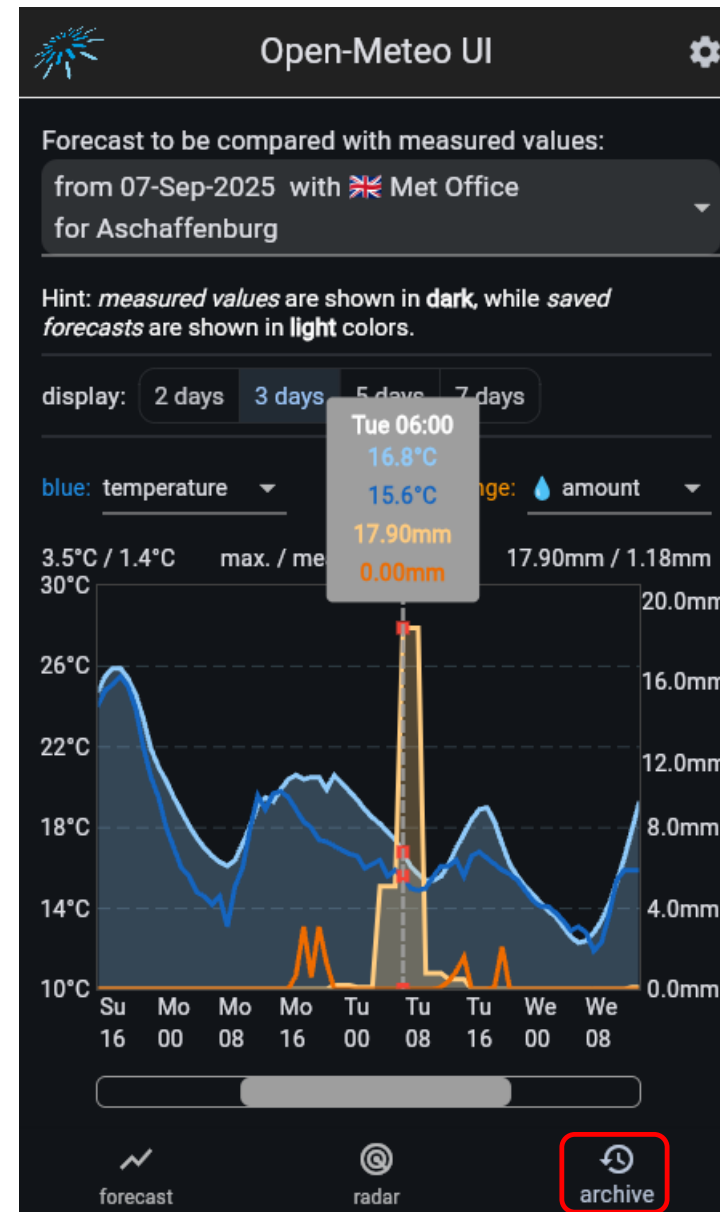
The app allows to compare forecasts later with real measurements.

You can switch app language from  to .

The UI is a bit crowded, improvement ideas are welcome.

[https://guenterschmitt.github.io/web\\_test/open\\_meteo\\_ui/](https://guenterschmitt.github.io/web_test/open_meteo_ui/)







## Parallel to Flutter you will learn to use Git & GitHub

- Create repositories
- Stage and commit
- Fetch, pull and push
- Branch
- Fork
- Use GitHub Pages to publish Flutter Web Apps



# Appendix: What says ChatGPT

ChatGPT ▾

why should I learn Flutter as a Software Design student

## 1. Cross-Platform Development with One Codebase

- Flutter lets you build apps for **Android, iOS, web, desktop, and even embedded devices** using one language (Dart) and one framework.
- This means you can focus more on **design and user experience** instead of managing separate projects for each platform.

## 2. Rapid Prototyping & UI Experimentation

- Flutter is very **design-centric**: its widget system makes it easy to experiment with UI/UX ideas quickly.
- Hot Reload lets you instantly see changes, which is great for **iterating on design concepts** during projects or user testing.

## 4. Strong Job & Freelance Opportunities

- Flutter is one of the **most in-demand frameworks** for mobile app development worldwide.
- Startups and even big companies like Google, BMW, and Alibaba use it. That means your skill will be **relevant in industry**.

✅ **In short:** Flutter helps you bring your software design ideas to life across platforms, rapidly test user experiences, and gain skills that are highly valuable in the job market.



# Hardware Requirements

ChatGPT ▾

What are the hardware requirements to learn flutter

## ◆ Minimum Hardware Requirements for Learning Flutter

### General (Windows, macOS, Linux)

- **Processor:** Intel i3 (or equivalent) – but i5/i7 or Apple M1/M2 is much better for speed.
- **RAM:** At least 8 GB (16 GB is highly recommended if you want to run emulators smoothly).
- **Storage:** ~10 GB free for Flutter SDK, Android Studio/Xcode, and emulators (SSD strongly recommended).
- **Graphics:** A GPU with OpenGL 3.0 support helps with rendering (most modern laptops already have this).

### For Android Development

- You need **Android Studio** (or VS Code with Android SDK).
- Running an Android Emulator is heavy → needs **at least 8 GB RAM + SSD**.
  - Alternative: connect a physical Android phone (much faster).

### For Web & Desktop

- Web apps: Runs in Chrome or Edge (lightweight).
- Desktop apps: Works on Windows, macOS, or Linux—just need the OS SDKs installed.

## ◆ Suggested Setup for a Smooth Experience

- **CPU:** Intel i5 / Ryzen 5 or better, OR Apple M1/M2 chip.
- **RAM:** 16 GB.
- **Storage:** 256+ GB SSD.
- **OS:**
  - Windows 10/11
  - macOS Monterey or later (if you also want iOS dev)
  - Linux (Ubuntu, Fedora, etc., works fine for Android/web)