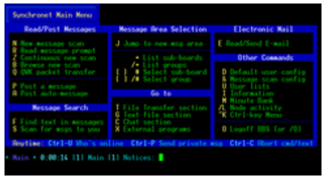
MOBILE DEVELOPMENT APP (REVIEWER)

Punch Card Programming:
The earliest applications were coded using punch cards, requiring meticulous manual work and limited functionality.

Mainframe Era:
Large, expensive mainframe computers dominated the scene, primarily used for scientific and government applications.

Limited User Interface:
Text-based interfaces and basic functionalities characterized early applications.



```
Synchronet Main Menu
  Read/Post Messages        Message Area Selection      Electronic Mail
N New message scan         J Jump to new msg area      E Read/Send E-mail
R Read message prompt            * List sub-boards       Other Commands
Z Continuous new scan           /* List groups         D Default user config
B Browse new scan          [ ]   # Select sub-board    S Message scan config
Q QWK packet transfer      { } /# Select group         U User lists
P Post a message                  Go to                I Information
A Post auto-message        T File Transfer section     M Minute Bank
    Message Search         G Text file section        /L Node activity
F Find text in messages    C Chat section             ^K Ctrl-key Menu
S Scan for msgs to you     X External programs         O Logoff BBS (or /O)
Anytime: Ctrl-U Who's online  Ctrl-P Send private msg  Ctrl-C Abort cmd/text
• Main • 0:00:16 [1] Main [1] Notices:
```

The Rise of Personal Computers (1970s - 1980s)

Birth of the PC

Graphical User Interfaces (GUIs):

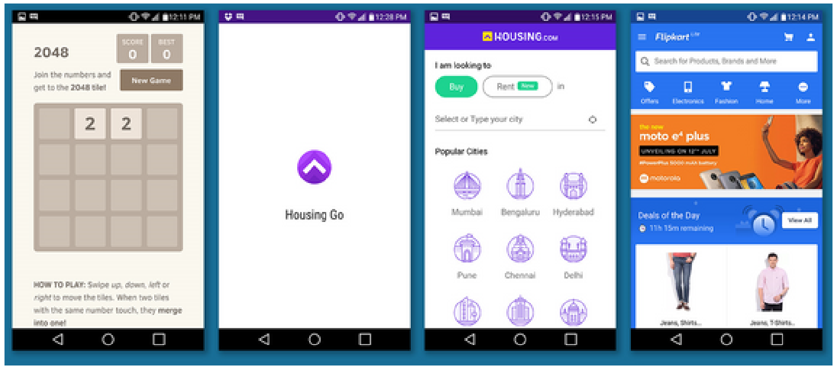
Rise of Software Developers

Birth of the PC: The invention of personal computers democratized access to computing, paving the way for more user-friendly applications.

Graphical User Interfaces (GUIs): The introduction of GUIs with icons and menus made applications more intuitive and accessible.

Rise of Software Developers: A new generation of programmers emerged, creating diverse applications for personal use and business needs.

The Networking Revolution (1990s - 2000s)



The Internet Boom: The rise of the internet revolutionized application development, enabling communication, collaboration, and information sharing on a global scale.

Client-Server Architecture: Applications split into client-side (user interface) and server-side (data processing) components, facilitating network-based functionality.

MOBILE DEVELOPMENT APP (REVIEWER)

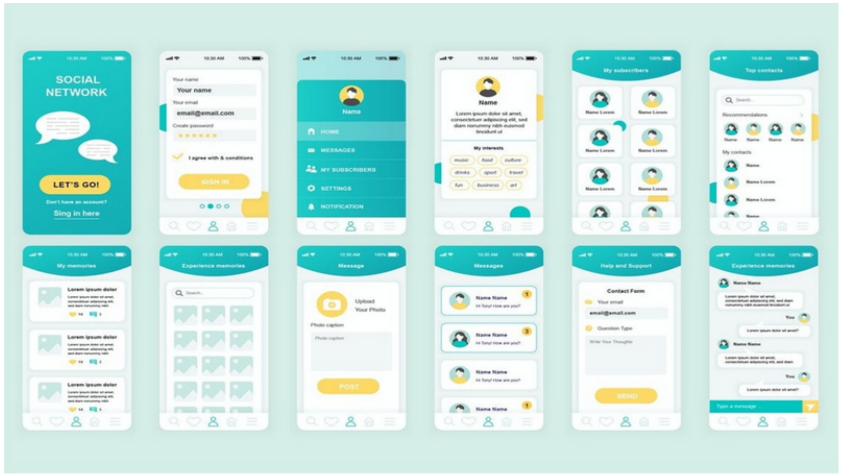
Web Applications: The emergence of web browsers and web applications transformed how users accessed and interacted with software.

The Mobile App Revolution (2010s - Present)

The Rise of Smartphones: The proliferation of smartphones and tablets created a new platform for innovative mobile applications.

Touch-Based Interfaces: Applications adapted to touch-based interfaces, requiring new design considerations and user interactions.

App Stores and Distribution: App stores like Google Play and Apple App Store simplified app discovery and distribution, fueling a thriving app ecosystem.



Core Principles of Mobile App Development

User-Centered Design: Understanding user needs and behavior (Personas, user journeys, accessibility)
Mobile-First Design: Tailoring the UI/UX for small screens and touch interaction
Performance and Efficiency: Optimizing resources, battery life, and network usage
Security and Privacy: Protecting user data and ensuring app integrity

Artificial Intelligence (AI) & Machine Learning (ML):

Personalized experiences: AI tailors the app to individual user preferences and needs.
Smart assistants: Chatbots provide 24/7 support and automate tasks.
Predictive recommendations: Apps anticipate user needs and suggest relevant content or actions.

Augmented Reality (AR) & Virtual Reality (VR):

AR overlays: Enhance the physical world with virtual information and interactive elements.
VR immersion: Create fully immersive experiences for gaming, education, and beyond.
Location-based experiences: AR/VR apps leverage GPS to deliver context-aware features.

MOBILE DEVELOPMENT APP (REVIEWER)

Internet of Things (IoT) Integration:

Smart home control: Manage lights, thermostats, and appliances from your phone.
Wearable data integration: Track fitness activity, health metrics, and personalize workouts.
Industrial automation: Monitor and control machinery remotely for increased efficiency.

Foldable Phone Optimization:

Flexible layouts: Design interfaces that adjust seamlessly to different screen sizes and orientations.
Multi-window functionality: Leverage the multitasking capabilities of foldable phones.
New interaction models: Explore innovative ways to interact with apps on foldable devices.

Beacon Technology

Proximity-based notifications: Send targeted offers and information to users based on their location.
Interactive experiences: Trigger actions like museum exhibits or store information based on a user's presence.
Indoor navigation: Provide guidance and wayfinding within buildings like airports or malls.

Cloud-Based Apps:

Reduced storage requirements: Store data and run complex processes in the cloud.
Real-time collaboration: Enable seamless teamwork and data sharing across devices.
Faster updates and deployments: Deliver new features and bug fixes instantly to all users.

5G Technology:

Ultra-fast speeds: Download content and stream data effortlessly.
Low latency: Enable real-time applications like video conferencing and AR/VR experiences.
Massive connectivity: Connect to billions of devices in the Internet of Things (IoT) ecosystem.

Voice User Interfaces (VUIs):

Hands-free interactions: Navigate apps, perform tasks, and access information using voice commands.
Accessibility for everyone: VUIs provide alternative input methods for users with disabilities.
Natural language processing: Apps understand and respond to complex sentences and requests.

What is Flutter?

Flutter is an open source framework developed and supported by Google. Frontend and full-stack developers use Flutter to build an application’s user interface (UI) for multiple platforms with a single codebase.
When **Flutter launched in 2018**, it mainly supported mobile app development. Flutter now supports application development on six platforms: iOS, Android, the web, Windows, MacOS, and Linux.

MOBILE DEVELOPMENT APP (REVIEWER)

How does Flutter help app development?

Flutter simplifies the process of creating consistent, appealing UIs for an application across the six platforms it supports.
Because Flutter is a cross-platform development framework,

Coding an application for one specific platform, such as iOS, is called native app development.

cross-platform app development is building an application for multiple platforms with a single codebase.

Native app development

Because developers code for a specific platform in native app development, they have full access to native device functionality.

if you want to launch an application on multiple platforms, native app development requires more code and more developers. In addition to these expenses, native app development can make it harder to launch on different platforms at the same time with a consistent user experience.

Cross-platform app development

Allows developers to use one programming language and one codebase to build an application for multiple platforms. If you're releasing an application for multiple platforms, cross-platform app development is less costly and time-consuming than native app development.
This process also lets developers create a more consistent experience for users across platforms.

The Advantages of Flutter

- Close-to-native performance. Flutter uses the programming language Dart and compiles into machine code. Host devices understand this code, which ensures a fast and effective performance.
- Fast, consistent, and customizable rendering. Instead of relying on platform-specific rendering tools, Flutter uses Google's open-source Skia graphic library to render UI. This provides users with consistent visuals no matter what platform they use to access an application.
- Developer-friendly tools. Google built Flutter with an emphasis on ease-of-use. With tools like hot reload, developers can preview what code changes will look like without losing state. Other tools like the widget inspector make it easy to visualize and solve issues with UI layouts.

MOBILE DEVELOPMENT APP (REVIEWER)

What is Dart?

Dart is a modern, object-oriented programming language developed by Google. It is specifically designed for building high-performance, cross-platform applications. Dart offers a clean syntax, strong typing, and a wide range of built-in libraries, making it a powerful tool for app development.

a versatile programming language with applications in various domains.

is the primary language used in Flutter, Google's UI toolkit for building natively compiled applications for mobile, web, and desktop from a single codebase.

Flutter's fast development, expressive UI, and excellent performance have made it a favorite choice among developers.

Web Development

Dart can also be used for web development, thanks to the Dart SDK's ability to compile Dart code to efficient JavaScript. Dart's strong typing, asynchronous programming support, and extensive libraries make it suitable for creating complex and performant web applications.

Server-Side Development

With frameworks like Aqueduct and Angel, Dart can be used for server-side development. These frameworks provide tools and libraries for building scalable and efficient backend systems, APIs, and web servers using Dart.

Internet of Things (IoT)

Dart's efficient and lightweight nature makes it well-suited for IoT development. Dart can be used to write code for microcontrollers, embedded systems, and IoT devices, enabling developers to create connected applications for the Internet of Things.

Flutter SDK Installation Guide

1. Download Flutter SDK
2. Once finished copy the file to your C: Drive
3. To test if the Flutter SDK has been successfully installed to your machine open flutter_console.bat
4. Run "flutter --version" to check the current version of your flutter

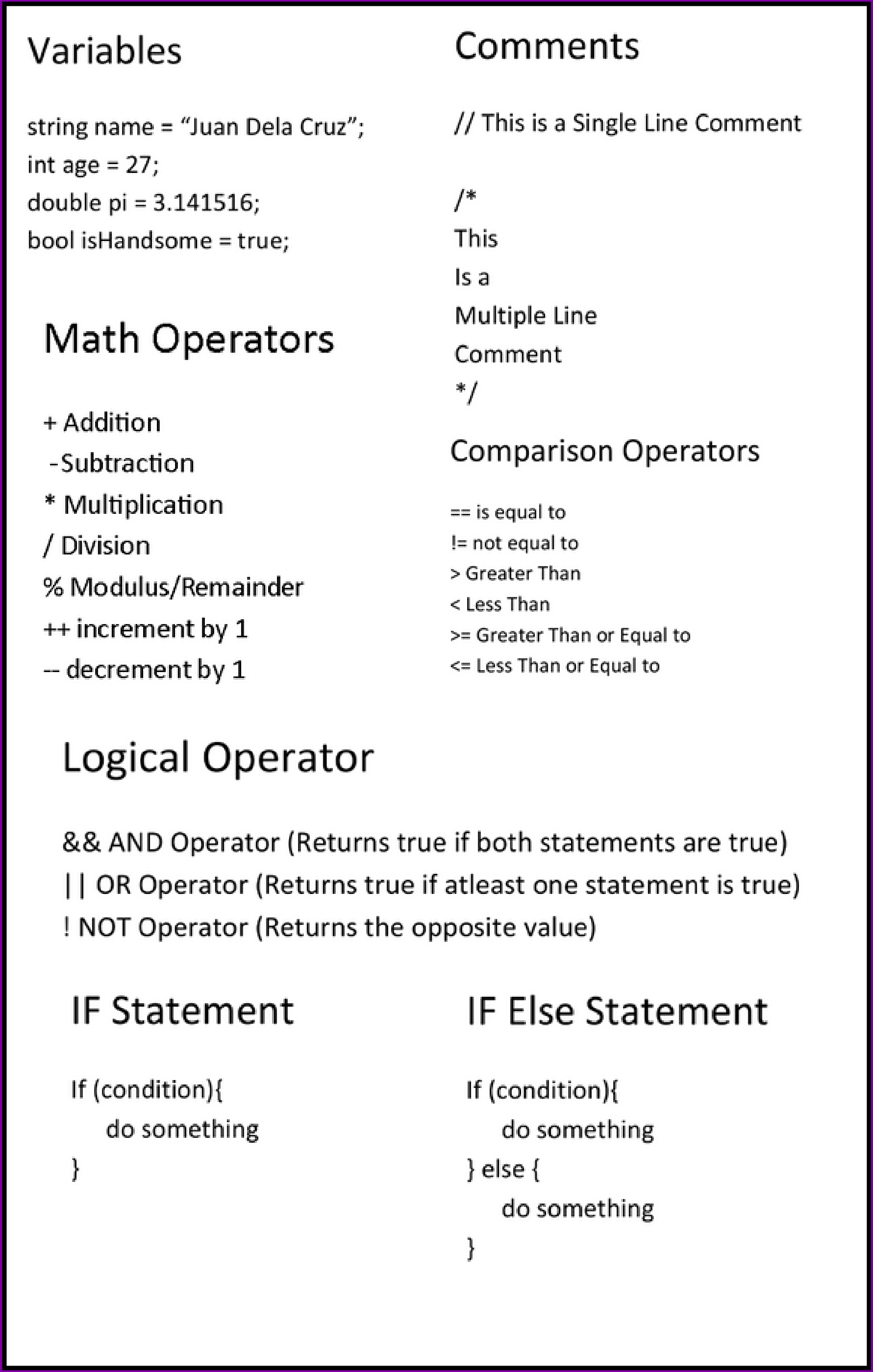
MOBILE DEVELOPMENT APP (REVIEWER)

To set the Flutter environment variable path

1. Click windows and search “Edit environment variable for your account”
2. Look for “PATH” under “User variables for..” click and hit edit
3. Click “new” and add the path towards flutter bin folder that might look like this “C:\flutter\flutter\bin”
4. Finish it off, and to test run CMD and run “flutter --version” command.

Android Studio Installation Guide

- Download latest version of Android Studio at https://developer.android.com/studio currently the latest version is Hedgehog.
- Install all the default settings and hit finish.
- Run Android Studio, and look for Plugins
- Search for Flutter and hit install, it will also promt to install Dart hit “Okay”
- Restart Android Studio and “Create New Flutter App” will be available.



Variables

```
string name = “Juan Dela Cruz”;
int age = 27;
double pi = 3.141516;
bool isHandsome = true;
```

Math Operators

+ Addition
-Subtraction
* Multiplication
/ Division
% Modulus/Remainder
++ increment by 1
-- decrement by 1

Comments

```
// This is a Single Line Comment

/*
This
Is a
Multiple Line
Comment
*/
```

Comparison Operators

== is equal to
!= not equal to
> Greater Than
< Less Than
>= Greater Than or Equal to
<= Less Than or Equal to

Logical Operator

&& AND Operator (Returns true if both statements are true)
|| OR Operator (Returns true if atleast one statement is true)
! NOT Operator (Returns the opposite value)

IF Statement

```
If (condition){
    do something
}
```

IF Else Statement

```
If (condition){
    do something
} else {
    do something
}
```

MOBILE DEVELOPMENT APP (REVIEWER)

ELSE IF Statement

```
if (condition){
    do something
} else if (another condition){
    do something
} else {
    do something
}
```

Switch Statement

```
Swith(expression){
    case value:
        do something
    break;
}
```

FOR Loop

```
For (initiation : condition : iteration){
    do something
}
```

WHILE Loop

```
While (condition){
    do something
}
```

```
// Basic Function
void greet(){
    print("Hello JP!");
}

//Function with Parameters
void greetMe(String name){
    print("Hello" + name);
}

//Function with Return Type
int add(int a, int b){
    int sum = a + b;
    return sum;
}
```

Data Structure

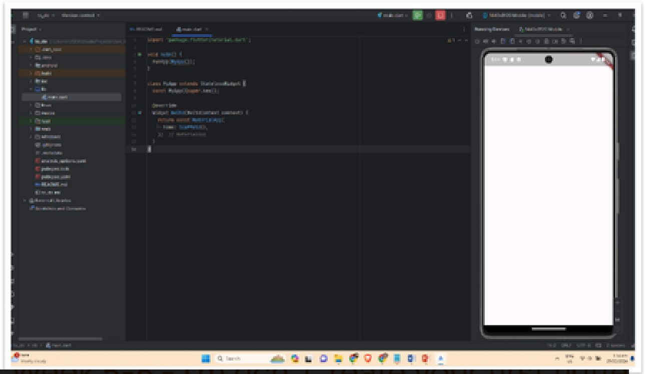
LIST (Ordered, Can have duplicates)
List<int> numbers = [1, 2, 3, 1, 2, 3];

Set (Unordered, unique elements
List<string> names = ["JP", "OJ", "AJ"];

Map (Stored as key-value pairs)
```
Map user = {
    'fullName' : "John Paulo",
    'age' : 27,
    'course' : "BS Information Technology"
}
```

SCAFFOLD

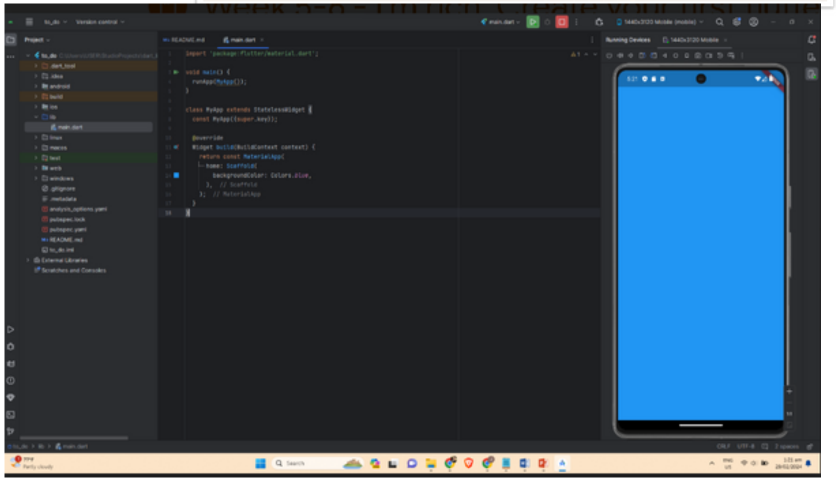
Scaffold provides many widgets or we can say APIs like Drawer, Snack-Bar, Bottom-Navigation-Bar, Floating-Action-Button, App-Bar, etc. Scaffold will expand or occupy the whole device screen.



```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: Scaffold(),
    ); // MaterialApp
  }
}
```

We can change its background color by adding:

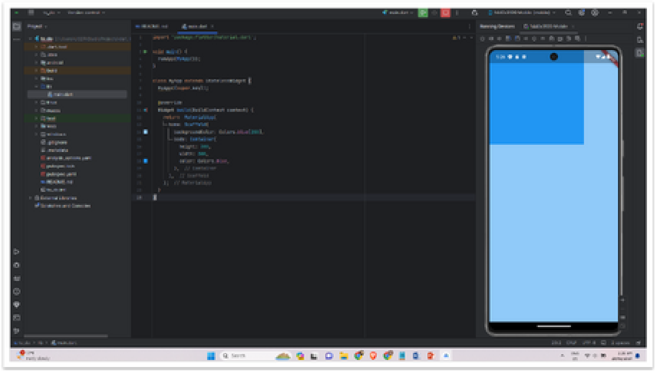
`backgroundColor: Colors.blue`



```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: Scaffold(
        backgroundColor: Colors.blue,
      ), // Scaffold
    ); // MaterialApp
  }
}
```

BODY

The primary content of the scaffold. Displayed below the appBar, above the bottom of the ambient MediaQuery's MediaQueryData. viewInsets, and behind the floatingActionButton and drawer.

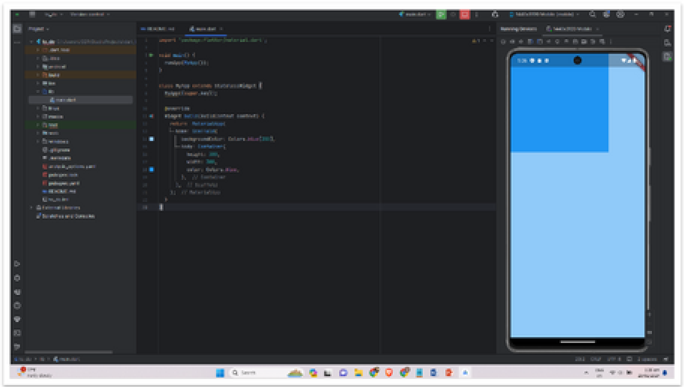


```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        backgroundColor: Colors.blue[100],
        body: Container(
          height: 300,
          width: 300,
          color: Colors.blue,
        ), // Container
      ), // Scaffold
    ); // MaterialApp
  }
}
```

MOBILE DEVELOPMENT APP (REVIEWER)

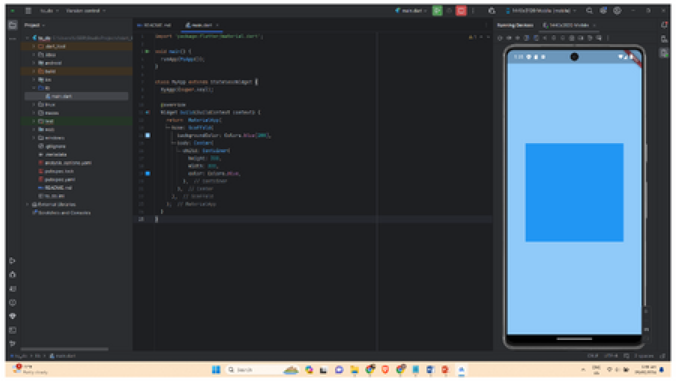
CONTAINER

A Flutter container is a widget/class used to apply styling effects to a widget. It serves as a container that can house a widget in it. It is equivalent to HTML's <div> tag or React's <View> component.



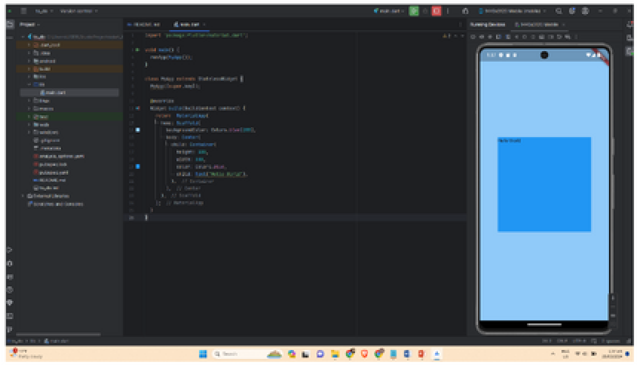
CENTER

Center widget comes built-in with flutter, it aligns its child widget to the center of the available space on the screen. The size of this widget will be as big as possible if the widthFactor and heightFactor properties are set to null and the dimensions are constrained.



CHILD

The child contained by the container. If null, and if the constraints are unbounded or also null, the container will expand to fill all available space in its parent, unless the parent provides unbounded constraints, in which case the container will attempt to be as small as possible.



PADDING

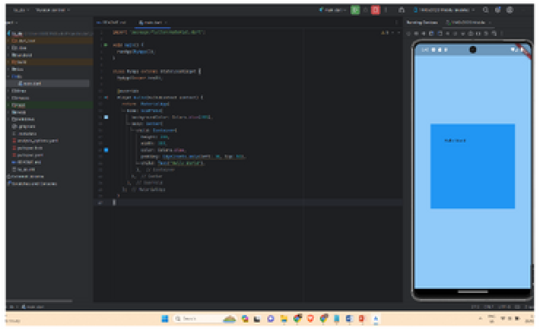
padding: EdgeInsets.all(50)

padding: EdgeInsets.symmetric(horizontal: 50)

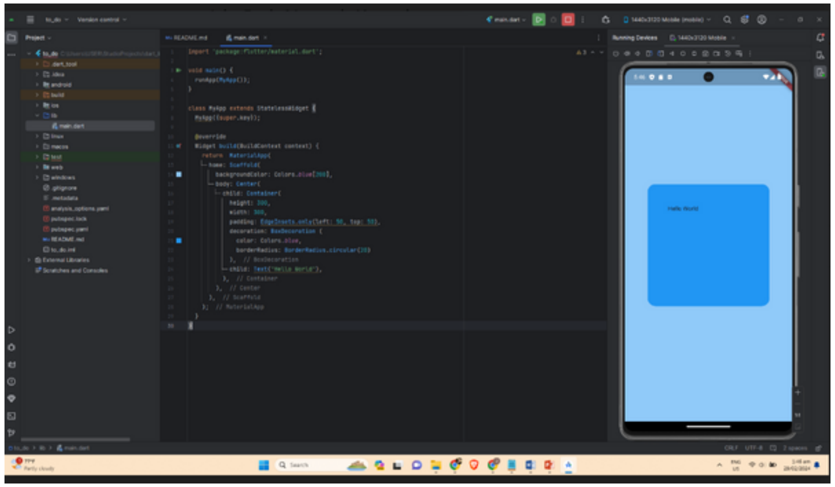
Or

padding: EdgeInsets.symmetric(vertical: 50)

padding: EdgeInsets.only(left: 50, top: 50)



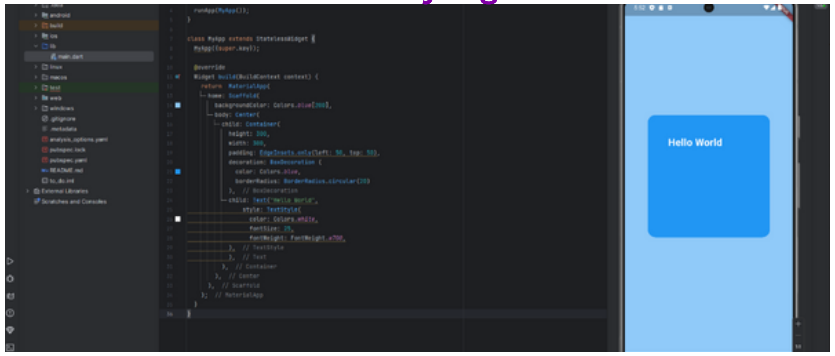
Decoration



```dart
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return  MaterialApp(
      home: Scaffold(
        backgroundColor: Colors.blue[200],
        body: Center(
          child: Container(
            height: 300,
            width: 300,
            padding: EdgeInsets.only(left: 50, top: 50),
            decoration: BoxDecoration (
              color: Colors.blue,
              borderRadius: BorderRadius.circular(20)
            ),  // BoxDecoration
            child: Text('Hello World'),
          ),  // Container
        ),  // Center
      ),  // Scaffold
    );  // MaterialApp
  }
}
```

Hello World

Text Styling

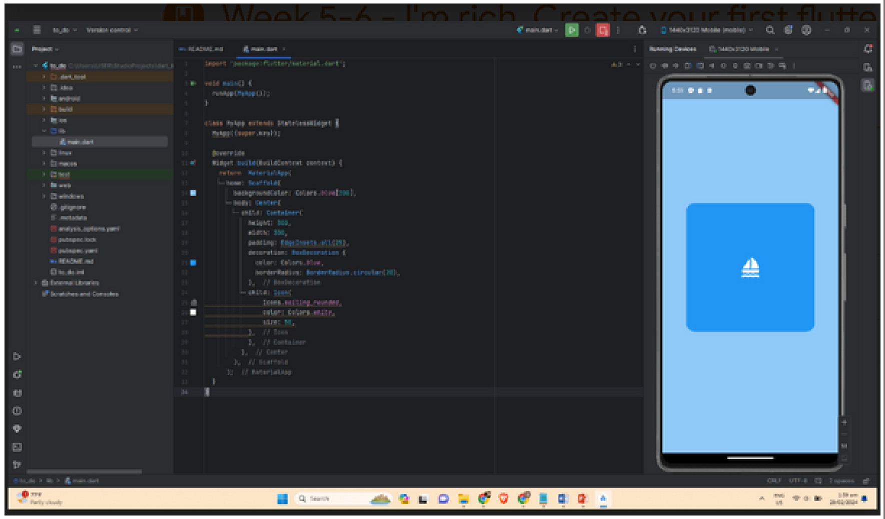


```dart
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return  MaterialApp(
      home: Scaffold(
        backgroundColor: Colors.blue[200],
        body: Center(
          child: Container(
            height: 300,
            width: 300,
            padding: EdgeInsets.only(left: 50, top: 50),
            decoration: BoxDecoration (
              color: Colors.blue,
              borderRadius: BorderRadius.circular(20)
            ),  // BoxDecoration
            child: Text('Hello World',
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 25,
                  fontWeight: FontWeight.w700,
            ),  // TextStyle
            ),  // Text
          ),  // Container
        ),  // Center
      ),  // Scaffold
    );  // MaterialApp
  }
}
```

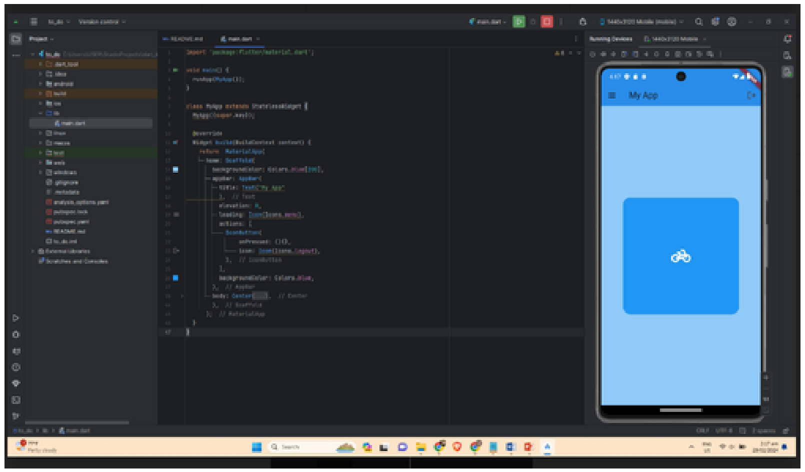
Hello World

MOBILE DEVELOPMENT APP (REVIEWER)

Icons



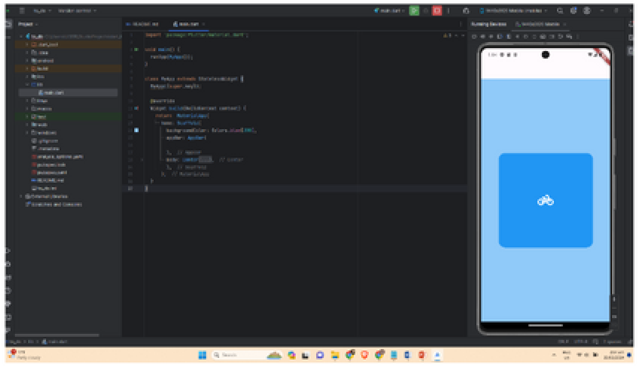
```dart
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        backgroundColor: Colors.blue[200],
        body: Center(
          child: Container(
            height: 300,
            width: 300,
            padding: EdgeInsets.all(20),
            decoration: BoxDecoration (
              color: Colors.blue,
              borderRadius: BorderRadius.circular(20),
            ), // BoxDecoration
            child: Icon(
                Icons.sailing_rounded,
                color: Colors.white,
                size: 50,
            ), // Icon
          ), // Container
        ), // Center
      ), // Scaffold
    ); // MaterialApp
  }
}
```



```dart
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        backgroundColor: Colors.blue[200],
        appBar: AppBar(
          title: Text("My App"
          ), // Text
          elevation: 0,
          leading: Icon(Icons.menu),
          actions: [
            IconButton(
                onPressed: (){},
                icon: Icon(Icons.logout),
            ), // IconButton
          ],
          backgroundColor: Colors.blue,
        ), // AppBar
        body: Center(...), // Center
      ), // Scaffold
    ); // MaterialApp
  }
}
```

APPBAR

AppBar is usually the topmost component of the app (or sometimes the bottom-most), it contains the toolbar and some other common action buttons. As all the components in a flutter application are a widget or a combination of widgets



```dart
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        backgroundColor: Colors.blue[200],
        appBar: AppBar(

        ), // AppBar
        body: Center(...), // Center
      ), // Scaffold
    ); // MaterialApp
  }
}
```

MOBILE DEVELOPMENT APP (REVIEWER)