

A
INTERNSHIP REPORT

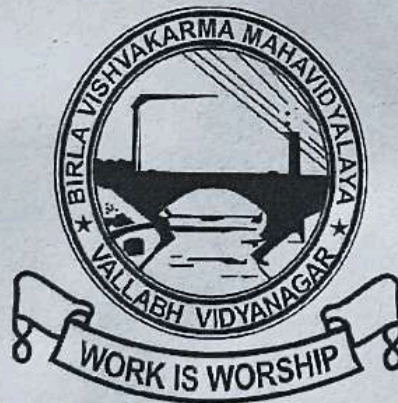
Under Subject of
“Flutter”

SUMMER INTERNSHIP - 1(ITIS1)

B.Tech. SEMESTER-II

Submitted by

Ansh Parikh (22IT433)



Information Technology Department
Birla Vishvakarma Mahavidyalaya Engineering College
(An Autonomous Institution)
AY: 2023-24, Semester II



Birla Vishvakarma Mahavidyalaya Engineering College

(An Autonomous Institution)

Information Technology Department

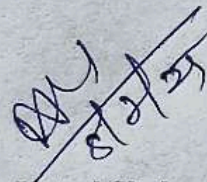
AY: 2023-24, Semester II

CERTIFICATE

This is to certify that Ansh Parikh with ID **22IT433** has been successfully completed summer Internship -1 (ITIS1) at **Mruttyuunjay Born To Develop**

under my guidance during the academic year 2023-24, Semester II.


Date:


Dr. Vatsal Shah

IT Department

BVM

Completion Letter



Born To Develop

AF - 37, 2ndFloor, RBG Complex, Opp. Bahucharaji Temple, Bahucharaji Road, Kareliabaug, Vadodara

Date: 25/06/2024

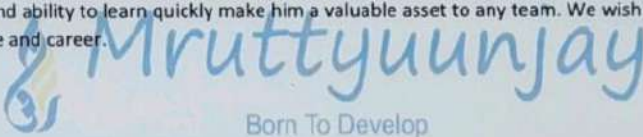
TO WHOM IT MAY CONCERN

This is to certify that **Mr. Ansh Niravbhai Parikh**, student of **Birla Vishvakarma Mahavidyalaya (BVM) Engineering College**, has successfully completed his internship in **Flutter** from **5th June 2024 to 25th June 2024** under the guidance of **Rajan Bavda**

During the period of his training with us, he had been exposed to different processes and was found diligent, hardworking and inquisitive. He has proved himself to be very astute and comprehensive in his knowledge of the technologies and efficient in given tasks.

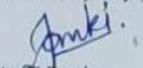
His training covered the fundamentals of Flutter, including the basics of Dart programming. He demonstrated a strong grasp of Dart syntax, data structures, and object-oriented programming principles.

His dedication and ability to learn quickly make him a valuable asset to any team. We wish him every success in his life and career.



Born To Develop

For Mruttyuunjay


Rajan Bavda

Mobile. +91 97241 33357
info@mruttyuunjay.com
www.mruttyuunjay.com

ACKNOWLEDGEMENT LETTER

Date: 25/06/2024

To,
Mruttyuunjay
AF-37, 2nd Floor, RBG Complex,
Opp. Bahucharaji Temple, Bahucharaji Road,
Kareliabaug, Vadodara

Subject: Acknowledgement for Internship Completion

Respected Sir,

I, Ansh Niravbhai Parikh, student of Birla Vishvakarma Mahavidyalaya (BVM) Engineering College, sincerely express my gratitude to Mruttyuunjay for providing me the opportunity to complete my internship in Flutter from 5th June 2024 to 25th June 2024 under the guidance of Mr. Rajan Bavda.

During this internship, I gained valuable knowledge and hands-on experience in Flutter development, Dart programming, and object-oriented principles. The guidance and support from the team have been instrumental in enhancing my technical and problem-solving skills.

I would like to extend my heartfelt thanks to Mr. Rajan Bavda and the entire team at Mruttyuunjay for their constant support and mentorship throughout my training. This experience has been incredibly enriching and will significantly contribute to my professional growth.

Once again, I sincerely appreciate the opportunity and look forward to applying the knowledge and skills I have acquired in future endeavors.

Thank you.

Sincerely,
Ansh Niravbhai Parikh
Birla Vishvakarma Mahavidyalaya (BVM) Engineering College

Table of Contents

Sr. No.	Particulars	Page No.
01.	Introduction	4
02.	Portfolio	15
03.	Flutter Installation and Configuration	18
04.	Introduction to Dart Language	21
05.	Introduction to Widgets	23
06.	Hello World App	27
07.	TO-DO App	29
08.	Calculator App	31
09.	Working With API	35
10.	News App	38

Daily Tasks

MRUTTYUUNJAY

Tasks

June

Date	To-Do	Done ?
5/06/2024	BootStrap Introduction.	Yes
6/06/2024	Grid basics.	Yes
7/06/2024	Navbar, Tables, Progressbar, Pagination, Dropdown, Form validation. Bootstrap CSS.	Yes
8/06/2024	PortFolio.	Yes
9/06/2024	PortFolio.	Yes
10/06/2024	Flutter Installation and Configuration.	Yes
11/06/2024	Introduction to Dart Language.	Yes
12/06/2024	Dart Language Basics.	Yes
13/06/2024	Dart Language Basics.	Yes
14/06/2024	Dart Language Basics.	Yes
15/06/2024	Introduction to Widgets.	Yes
16/06/2024	Study of Different types of Widgets.	Yes
17/06/2024	Study of Different types of Widgets.	Yes
18/06/2024	Study of Different types of Widgets.	Yes
19/06/2024	Create an APP (HELLO WORLD).	Yes
20/06/2024	Create Calculator APP.	Yes
21/06/2024	Create TO-DO APP.	Yes
22/06/2024	Create TO-DO APP.	Yes
23/06/2024	Working with API.	Yes
24/06/2024	Working with API.	Yes
25/06/2024	Create NEWS APP.	Yes

1: Introduction

Introduction to Web Development Technologies

In the realm of web development, several fundamental technologies form the backbone of modern websites. Understanding these technologies is crucial for anyone aspiring to create engaging and dynamic web experiences.

HTML (Hypertext Markup Language):

HTML serves as the foundation of web pages, defining their structure and content. Through a system of tags, HTML specifies how elements such as text, images, videos, and links are organized and displayed on a webpage.

CSS (Cascading Style Sheets):

CSS complements HTML by controlling the visual presentation of web pages. It enables developers to customize the layout, colors, fonts, and other stylistic aspects of HTML elements, thus enhancing the overall aesthetics and user experience.

Bootstrap 5:

Bootstrap 5 stands out as a leading front-end development framework, renowned for its ability to streamline the creation of responsive and mobile-friendly web designs. By offering a comprehensive suite of pre-designed CSS and JavaScript components, Bootstrap empowers developers to rapidly prototype and construct visually appealing web interfaces.

JavaScript:

JavaScript adds interactivity and dynamic functionality to web pages, elevating them beyond static content. As a versatile programming language, JavaScript enables developers to create engaging features such as animations, form validations, and interactive elements by manipulating the HTML and CSS of a webpage in real-time.

In essence, HTML, CSS, Bootstrap 5, and JavaScript constitute the core building blocks of modern web development, each playing a distinct yet interconnected role in crafting captivating and user-friendly online experiences.

Grid Basics:

CSS Grid Layout: The Grid Layout Module offers a grid-based layout system, with rows and columns. The Grid Layout Module makes it easier to design a responsive layout structure,

without using float or positioning. The CSS grid properties are supported in all modern browsers.

Grid Container:

To make an HTML element behave as a grid container, you have to set the display property to grid or inline-grid. Grid containers consist of grid items, placed inside columns and rows.

Grid-template-columns Property: The grid-template-columns property defines the number of columns in your grid layout, and it can define the width of each column. The value is a space-separated-list, where each value defines the width of the respective column.

If you want your grid layout to contain 4 columns, specify the width of the 4 columns, or "auto" if all columns should have the same width.

Code:

```
<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

  display: grid;

  height: 400px;

  align-content: end;

  grid-template-columns: auto auto auto;

  gap: 10px;

  background-color: #2196F3;

  padding: 10px;

}

.grid-container > div {

  background-color: rgba(255, 255, 255, 0.8);

  text-align: center;

  padding: 20px 0;

  font-size: 30px;

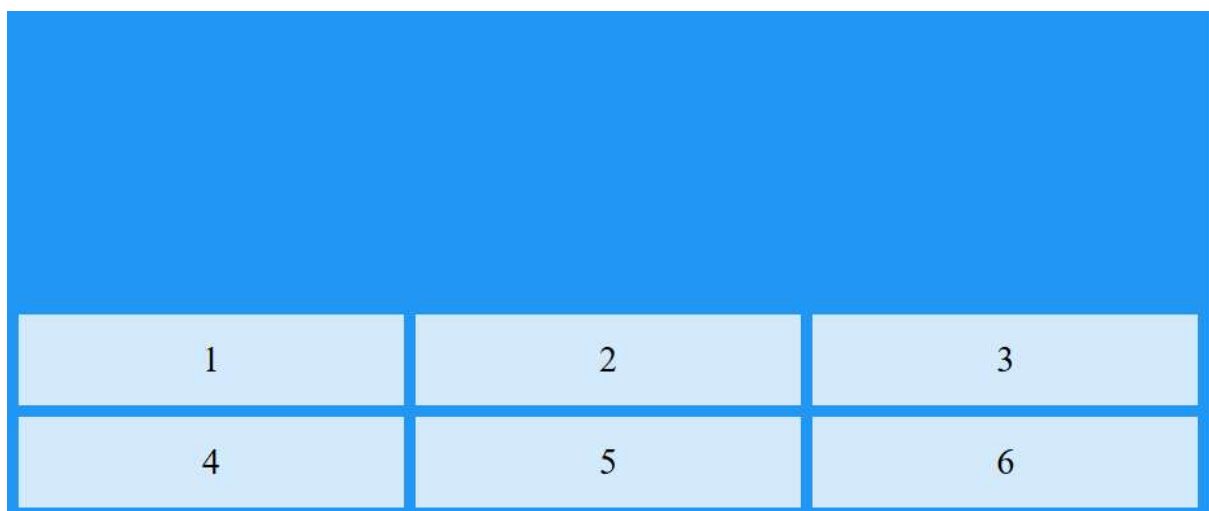
}

</style>

</head>
```



```
<body>
<h1>The align-content Property</h1>
<p>Use the align-content property to vertically align the grid inside the
container.</p>
<p>The value "end" will align the rows at the end of the container:</p>
<div class="grid-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
</div>
</body>
</html>
```

Output:**Grid Items**

A grid container contains grid items. By default, a container has one grid item for each column, in each row, but you can style the grid items so that they will span multiple columns and/or rows.

Code:

```
<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {
  display: grid;
  height: 600px;
  grid-template-columns: 1fr 1fr 1fr;
  gap: 10px;
  background-color: dodgerblue;
  padding: 10px;
}

.grid-container > div {
  background-color: #f1f1f1;
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}

.item1 {
  align-self: start;
}

.item6 {
  align-self: center;
}

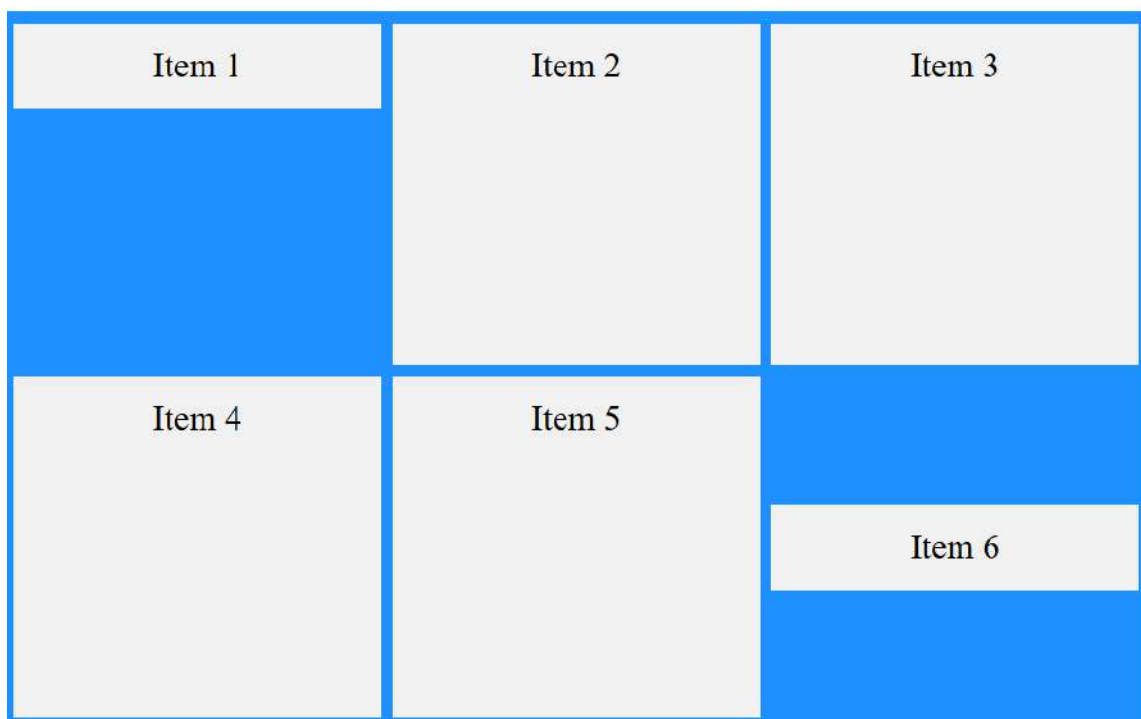
</style>

</head>

<body>
```

```
<h2>The align-self property</h2>
```

```
<div class="grid-container">  
  <div class="item1">Item 1</div>  
  <div class="item2">Item 2</div>  
  <div class="item3">Item 3</div>  
  <div class="item4">Item 4</div>  
  <div class="item5">Item 5</div>  
  <div class="item6">Item 6</div>  
</div>  
  
</body>  
</html>
```

Output:

NavBar:**Code:**

```
<!DOCTYPE html>

<html>

<head>

<style>

ul {

    list-style-type: none;

    margin: 0;

    padding: 0;

    overflow: hidden;

    background-color: #333;

}

li {

    float: left;

    border-right: 1px solid #bbb;

}

li:last-child {

    border-right: none;

}

li a {

    display: block;

    color: white;

    text-align: center;

    padding: 14px 16px;

    text-decoration: none;

}

li a:hover:not(.active) {

    background-color: #111;
```

```

}
.active {
  background-color: #04AA6D;
}
</style>
</head>
<body>

<ul>
  <li><a class="active" href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li style="float:right"><a href="#about">About</a></li>
</ul>

</body>
</html>

```

Output:



ProgressBar:

Code:

```

<!DOCTYPE html>
<html>
<title>P3.CSS</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.example3.com/w4css/4/w4.css">
<body>
<div class="w4-container">

```

<h2>Progress Bar Labels</h2>

<p>Add text inside a w4-container element to add a label to the progress bar.</p>

<p>Use the w4-center class to center the label. If omitted, it will be left aligned.</p>

```
<div class="w4-light-grey">
```

```
  <div class="w4-container w4-green w4-center" style="width:25%">25%</div>
```

```
</div><br>
```

```
<div class="w4-light-grey">
```

```
  <div class="w4-container w4-red w4-center" style="width:50%">50%</div>
```

```
</div><br>
```

```
<div class="w4-light-grey">
```

```
  <div class="w4-container w4-blue" style="width:75%">75%</div>
```

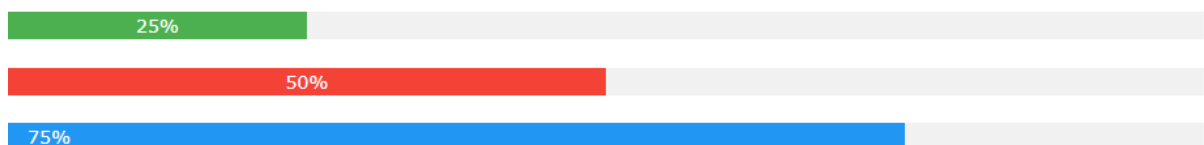
```
</div><br>
```

```
</div>
```

```
</body>
```

```
</html>
```

Output:



Pagination:

Code:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```



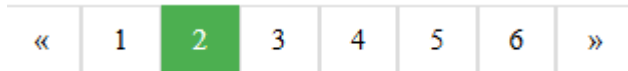
```
.pagination {  
  display: inline-block;  
}  
.pagination a {  
  color: black;  
  float: left;  
  padding: 8px 16px;  
  text-decoration: none;  
  transition: background-color .3s;  
  border: 1px solid #ddd;  
}  
  
.pagination a.active {  
  background-color: #4CAF50;  
  color: white;  
  border: 1px solid #4CAF50;  
}  
.pagination a:hover:not(.active) {background-color: #ddd;}  
</style>  
</head>  
<body>  
<h2>Pagination with Borders</h2>  
<div class="pagination">  
  <a href="#">&laquo;</a>  
  <a href="#">1</a>  
  <a href="#" class="active">2</a>  
  <a href="#">3</a>  
  <a href="#">4</a>  
  <a href="#">5</a>  
  <a href="#">6</a>
```

```
<a href="#">&raquo;</a>
</div>
```

```
</body>
```

```
</html>
```

Output:



Form with Validation:

Code:

```
<!DOCTYPE html>

<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
<div class="container mt-3">
  <h3>Form Validation</h3>
  <p>Try to submit the form.</p>

  <form action="/action_page.php" class="was-validated">
    <div class="mb-3 mt-3">
      <label for="uname" class="form-label">Username:</label>
      <input type="text" class="form-control" id="uname" placeholder="Enter username"
name="uname" required>
```

```

<div class="valid-feedback">Valid.</div>

<div class="invalid-feedback">Please fill out this field.</div>

</div>

<div class="mb-3">

  <label for="pwd" class="form-label">Password:</label>

  <input type="password" class="form-control" id="pwd" placeholder="Enter password"
name="pswd" required>

  <div class="valid-feedback">Valid.</div>

  <div class="invalid-feedback">Please fill out this field.</div>

</div>

<div class="form-check mb-3">

  <input class="form-check-input" type="checkbox" id="myCheck" name="remember"
required>

  <label class="form-check-label" for="myCheck">I agree on blabla.</label>

  <div class="valid-feedback">Valid.</div>

  <div class="invalid-feedback">Check this checkbox to continue.</div>

</div>

<button type="submit" class="btn btn-primary">Submit</button>

</form>

</div>

</body>

</html>

```

Output:

Form Validation

Try to submit the form.

Username:

Please fill out this field.

Password:

Please fill out this field.

☐ I agree on blabla.
Check this checkbox to continue.

2: PORTFOLIO


Index:




About:



Resume:



Ansh Parikh



Home

About

Resume

Portfolio

Services

Contact

Resume

Dedicated IT professional with expertise in web development, proficient in HTML, CSS, JavaScript, PHP, and Oracle database management, seeking to leverage technical skills and problem-solving abilities to contribute effectively to your team.

Summary

ANSH PARIKH
I seek challenging opportunities where i can fully use my skills for the success of the organization.

Education

- BIRLA VISWAKARMA MAHAVIDYALAYA**
Branch: Information Technology
CPI: 7.17/10
- SANSKRUTI HIGHER SECONDARY SCHOOL**
HSC (Science Stream - A Group)
Percentage: 84.05%
- THE WESTERN ENGLISH MEDIUM SCHOOL**
SSC
Percentage: 94%


Skills

- Team building, Problem Solving, Decision making
- Frontend: HTML, CSS, Javascript, Bootstrap
- Backend: PHP
- Database: Oracle
- Programming Language: C, C++
- Core Skills: OOPs, DBMS, Basic Networking


Projects

- CAR BUYING WEBSITE**
A car buying website offers a convenient platform for users to browse, compare, and purchase vehicles online. It typically includes features like online car booking, service booking, and a wide selection of different car models to choose from.
- GYM WEBSITE (FRONTEND)**

Portfolio:



Ansh Parikh



Home

About

Resume

Portfolio

Services

Contact

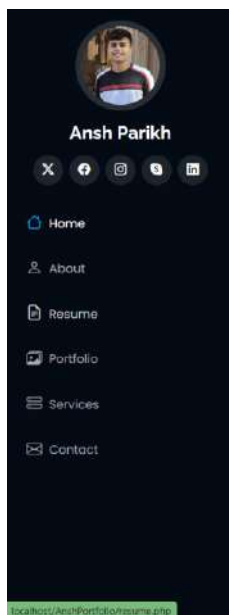
Portfolio

Showcasing my journey as an IT student through projects that blend technical proficiency with creative problem-solving.

ALL WEBSITE GRAPHICS



Services:



Services

Providing innovative IT solutions and support to enhance technology-driven learning and development.



Web Developer

Crafting responsive and scalable web solutions using modern technologies to enhance digital experiences.

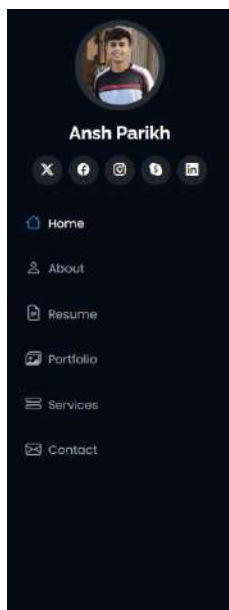


Graphics Designer

Creating visually compelling designs that communicate brands' identities and captivate audiences.

© 2024 **Portfolio** All Rights are Reserved
Developed by Ansh Parikh

Contact:



Contact

Address

12-234, Outside Limba Kul, Near Palak Hospital, Near Dipak sherry, Petlad

Call Us

+91 9106110380

Email Us

anshpetao@gmail.com



Your Name

Your Email

Subject

Message

Send Message

3: Flutter Installation and Configuration

1. System Requirements

- Operating System: Windows, macOS, or Linux.
- Disk Space: Minimum 2.8 GB (not including IDE/tools).
- Tools: Git, Android Studio.

2. Install Flutter SDK

1. Download Flutter SDK:
 - Go to the Flutter official website and download the Flutter SDK for your OS.
2. Extract the Flutter SDK:
 - Extract the zip file to a directory, e.g., C:\flutter (for Windows) or /usr/local/flutter (for Linux/Mac).
3. Update PATH:
 - Add the Flutter bin directory to your PATH:
 - Windows:
Add C:\flutter\bin to the system's PATH environment variable.
 - Linux/Mac:
Add export PATH="\$PATH:/path-to-flutter/bin" in the .bashrc, .zshrc, or .bash_profile file.
4. Verify Installation: Run the command:

bash

Copy code

flutter doctor

This checks the dependencies and shows the current setup.

3. Install Android Studio

1. Download and Install Android Studio:
 - Visit [Android Studio](#) and download the latest version.
2. Install Required SDK Components:
 - Open Android Studio and go to File > Settings > Appearance & Behavior > System Settings > Android SDK.
 - Select SDK Platforms and install the latest Android version and API level.

- Go to the SDK Tools tab and install:
 - Android SDK Build-Tools.
 - Android Emulator.
 - Android Platform Tools.
- 3. Set Up Emulator (Optional):
 - Open AVD Manager from Android Studio.
 - Create a new virtual device, select a hardware profile, and download the recommended system image.

4. Configure Flutter with Android Studio

1. Install Flutter and Dart Plugins:
 - Open Android Studio.
 - Go to File > Settings > Plugins.
 - Search for "Flutter" and install it (Dart plugin is installed automatically).
2. Restart Android Studio.

5. Configure Android Device

1. Enable USB Debugging:
 - On your Android device, go to Settings > About Phone > Tap Build Number (7 times) to enable developer options.
 - In Developer Options, enable USB Debugging.
2. Connect Device:
 - Connect your device via USB.
 - Run the command:

```
flutter devices
```
3. This lists all connected devices.

6. Create and Run Flutter App

1. Create a New App:

```
flutter create my_app  
cd my_app
```

2. Run the App:

- Run the app using a connected device or emulator:

`flutter run`

7. Troubleshooting

1. If Flutter doctor reports missing components, follow its suggestions.
2. Ensure all tools (Flutter, Android SDK, etc.) are updated.

Now, your Flutter environment should be fully configured for Android development!

4: Introduction to Dart Language

Dart is a programming language created by Google. It is used to build apps for mobile, web, and desktop platforms. Dart is the main language for creating apps with Flutter, a framework for making cross-platform apps using a single codebase. It is easy to learn, object-oriented, and treats everything as an object, including numbers and functions.

Dart is fast and powerful because it uses two types of compilation: Just-in-Time (JIT) for quick testing during development and Ahead-of-Time (AOT) for better performance when the app is released. It also supports features like `async` and `await`, which make it simple to handle tasks like loading data from the internet. Dart automatically manages memory with garbage collection, so developers don't have to worry about freeing up memory manually.

Dart's syntax is clean and straightforward. You can use it to define variables, write functions, and create classes to organize your code. It also supports tools like hot reload, which lets you see changes in your app immediately without restarting it, making development faster and easier.

Dart is not just for Flutter apps. It can also be used for web development, where it converts to JavaScript to run in browsers, or for creating server-side applications and scripts. Its flexibility makes it suitable for many types of projects.

Overall, Dart is a simple, fast, and reliable language that is especially useful for creating modern apps with great user interfaces.

Variables:

Syntax:

```
void main() {
  var name = 'Dart'; // Type inference
  String language = 'Flutter'; // Explicit type
  int version = 3;
  print('$name $language $version'); // String interpolation
}
```

Control Flow:

```
void main() {
  for (int i = 0; i < 5; i++) {
    print('Count: $i');
  }
  if (2 > 1) {
```

```
    print('Dart is awesome!');  
  }  
}
```

Functions:

```
void greet(String name) {  
  print('Hello, $name!');  
}  
  
void main() {  
  greet('Flutter Dev');  
}
```

Classes and Objects:

```
class Person {  
  String name;  
  int age;  
  Person(this.name, this.age);  
  void introduce() {  
    print('Hi, I am $name and I am $age years old.');
```

Asynchronous Programming:

```
Future<String> fetchData() async {  
  return 'Data loaded!';  
}  
  
void main() async {  
  String data = await fetchData();  
  print(data); }
```


5:Introduction To Widgets

In Flutter, **widgets** are the basic building blocks of any app. A widget is an immutable object that describes a part of the user interface. Everything you see in a Flutter app, from a button to text, images, and layout structure, is a widget. Widgets define the visual and functional structure of your app.

Widgets in Flutter are classified into two main types: **Stateless Widgets** and **Stateful Widgets**. A **Stateless Widget** does not change its state during the app's lifetime, meaning its appearance and properties remain the same. For example, a Text widget displaying static content is a Stateless Widget. On the other hand, a **Stateful Widget** can change its state based on user interaction or other events, like a button that changes its color when clicked.

Widgets are often arranged in a tree structure, called the **widget tree**. This hierarchical arrangement allows you to nest widgets to build complex layouts. For instance, a Scaffold widget can contain an AppBar, a Body, and a FloatingActionButton, each of which is also a widget.

One of the key features of widgets is that they are **composable**. This means you can combine simple widgets to create more complex ones. For example, you can use a Row or Column widget to arrange multiple Text or Image widgets horizontally or vertically.

Flutter provides a wide variety of widgets for designing your app, including **Material Design widgets** (like AppBar, FloatingActionButton, Card) and **Cupertino widgets** for iOS-style designs. You can also create custom widgets to suit your specific app requirements.

In summary, widgets are the foundation of Flutter apps, used to create both the visual and functional parts of the application. Understanding how to use and combine widgets effectively is key to building efficient and beautiful Flutter apps.

Different Types of Widgets:

1. Basic Widgets

- **Text:** Displays a piece of text.

```
Text('Hello, Flutter!');
```

- **Container:** A versatile widget for adding padding, margins, borders, and backgrounds.

```
Container(
  padding: EdgeInsets.all(10),
  color: Colors.blue,
  child: Text('I am a container!'),
);
```

- **Image:** Displays images from assets, network, or memory.

```
Image.asset('assets/image.png');
```

```
Image.network('https://example.com/image.png');
```

- **Icon:** Displays an icon from Flutter's Material Icons or custom sets.

```
Icon(Icons.home, size: 30, color: Colors.red);
```

2. Layout Widgets

- **Row:** Arranges widgets horizontally.

```
Row(
  children: [Text('Item 1'), Text('Item 2')],
);
```

- **Column:** Arranges widgets vertically.

```
dart
Copy code
Column(
  children: [Text('Item 1'), Text('Item 2')],
);
```

- **Stack:** Overlays widgets on top of each other.

```
Stack(
  children: [
    Container(color: Colors.blue, width: 100, height: 100),
    Text('Overlaid Text'),
  ],
);
```

- **ListView:** Displays a scrollable list of widgets.

```
ListView(
  children: [Text('Item 1'), Text('Item 2')],
);
```

3. Input Widgets

- **TextField:** Allows user input.

```
TextField(decoration: InputDecoration(labelText: 'Enter your name'));
```

- **Button Widgets:**

- **ElevatedButton:** A Material Design button.

```
ElevatedButton(onPressed: () {}, child: Text('Click Me'));
```

- **TextButton:** A flat button.

```
dart
```

```
Copy code
```

```
TextButton(onPressed: () {}, child: Text('Click Me'));
```

- **IconButton:** A button with an icon.

```
IconButton(onPressed: () {}, icon: Icon(Icons.add));
```

4. Scaffold and AppBar

- **Scaffold:** Provides a basic layout structure for an app (header, body, floating button).

```
Scaffold(
  appBar: AppBar(title: Text('My App')),
  body: Center(child: Text('Hello, Flutter!')),
  floatingActionButton: FloatingActionButton(onPressed: () {}, child:
    Icon(Icons.add)),
);
```

- **AppBar:** Displays the title and actions for a screen.

```
AppBar(title: Text('App Title'));
```

5. Navigation Widgets

- **Navigator:** Used for navigating between screens.

```
Navigator.push(context, MaterialPageRoute(builder: (context) =>
  NewScreen()));
```

- **BottomNavigationBar:** For switching between different tabs in an app.

```
BottomNavigationBar(items: [
  BottomNavigationBarItem(icon: Icon(Icons.home), label: 'Home'),
  BottomNavigationBarItem(icon: Icon(Icons.settings), label: 'Settings'),]);
```

6. Styling and Interaction Widgets

- **Padding:** Adds spacing around a child widget.

Padding(

padding: EdgeInsets.all(10),

child: Text('Padded Text'),

);

- **Center:** Centers a child widget within its parent.

Center(child: Text('Centered Text'));

- **GestureDetector:** Captures gestures like taps or swipes.

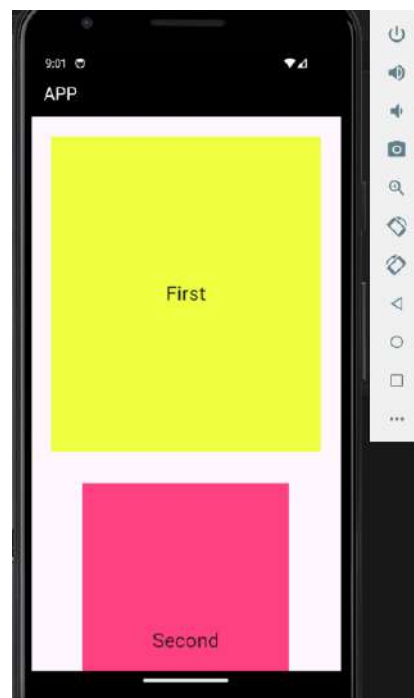
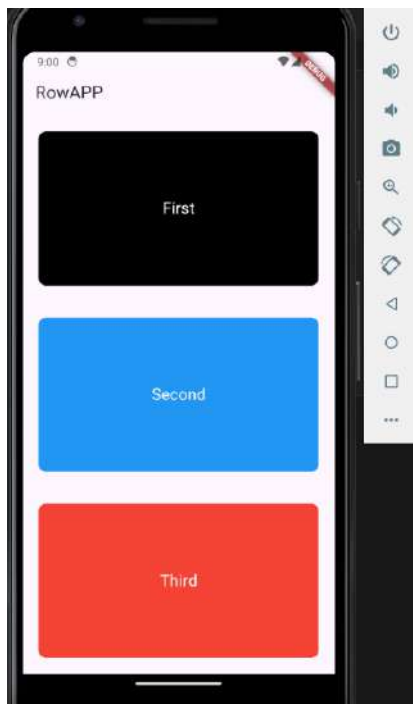
GestureDetector(

onTap: () => print('Tapped!'),

child: Text('Tap Me'),

);

Images:



6:Hello World APP

Code:

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: true, // Shows debug banner
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

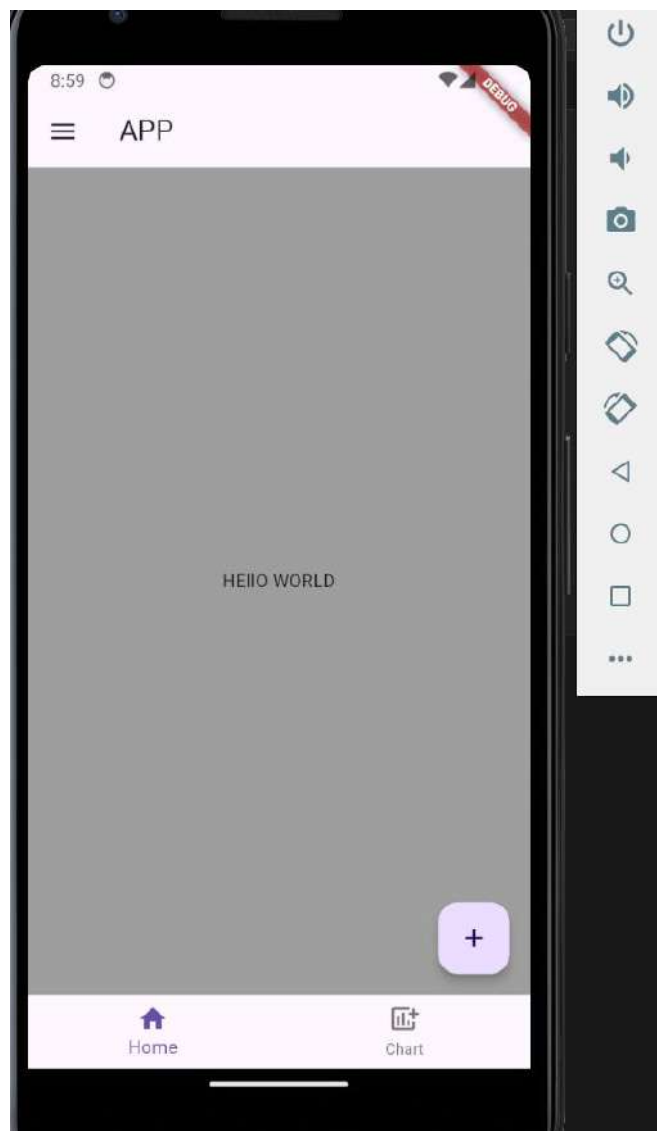
class _MyHomePageState extends State<MyHomePage> {
  int _currentIndex = 0; // For bottom navigation

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("APP"), // App title
        backgroundColor: Colors.purple.shade100, // Light purple background
      ),
```

```
drawer: Drawer(), // Sidebar menu drawer
body: Container(
  color: Colors.grey, // Background color
  child: Center(
    child: Text(
      "HELLO WORLD",
      style: TextStyle(fontSize: 16, color: Colors.black),
    ),
  ),
),
bottomNavigationBar: BottomNavigationBar(
  currentIndex: _currentIndex,
  onTap: (index) {
    setState(() {
      _currentIndex = index;
    });
  },
  selectedItemColor: Colors.purple, // Active tab color
  items: [
    BottomNavigationBarItem(
      icon: Icon(Icons.home),
      label: "Home",
    ),
    BottomNavigationBarItem(
      icon: Icon(Icons.bar_chart),
      label: "Chart",
    ),
  ],
),
floatingActionButton: FloatingActionButton(
```

```
onPressed: () {},  
backgroundColor: Colors.purple.shade100, // Light purple color  
child: Icon(Icons.add, color: Colors.purple), // "+" icon  
,  
floatingActionButtonLocation: FloatingActionButtonLocation.endFloat,  
);  
}  
}
```

Output:



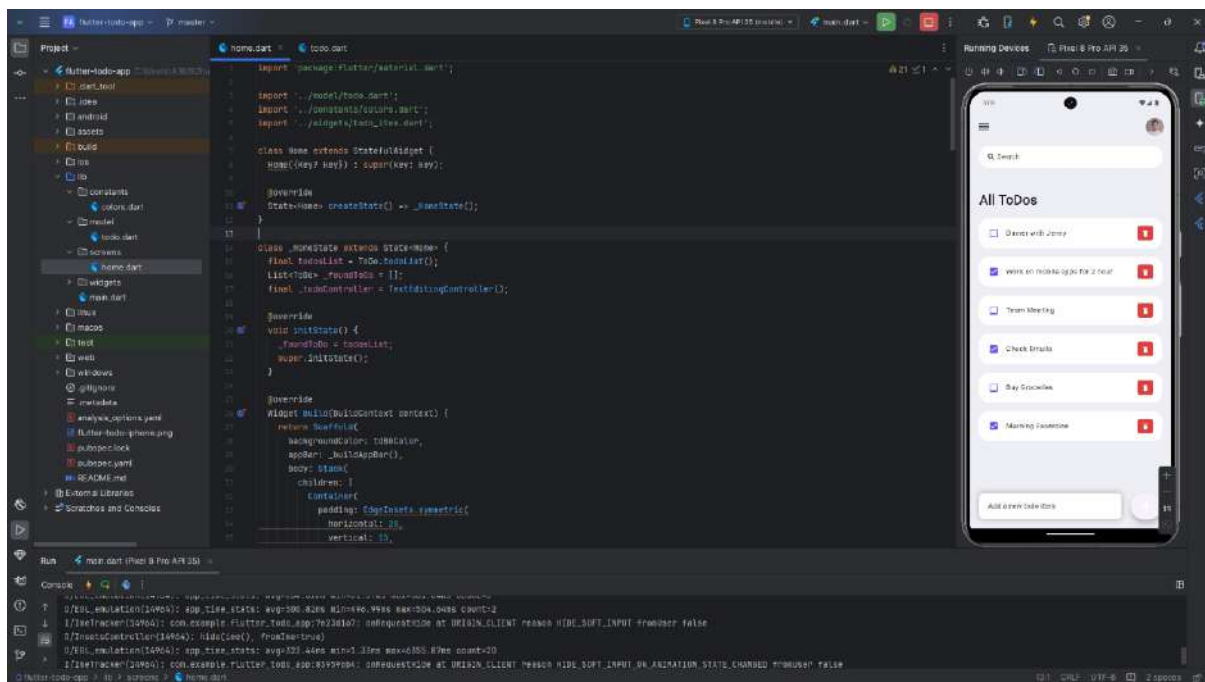
7: TODO APP

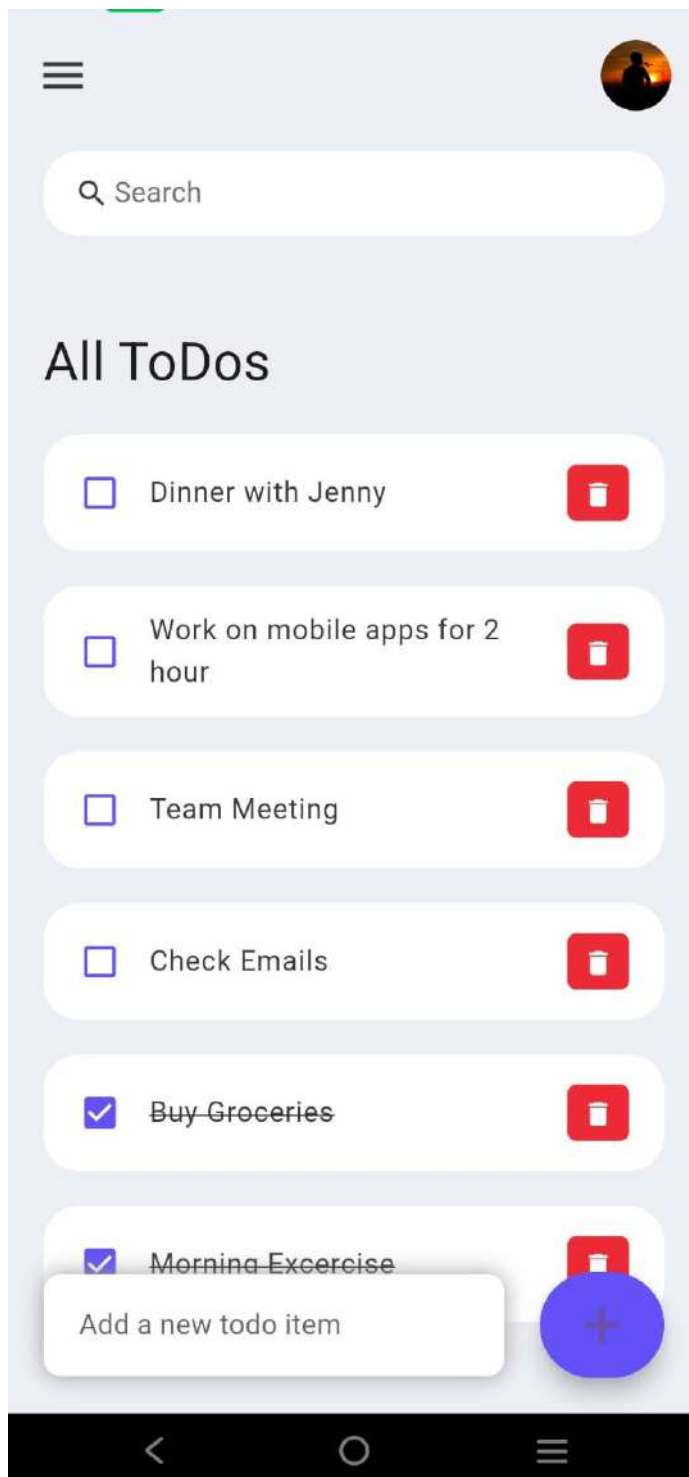
A Flutter To-Do App is a simple task management application that allows users to add, update, delete, and mark tasks as completed. It typically includes features like:

Key Features:

- ✓ Add new tasks with a title and description.
- ✓ Mark tasks as completed or pending.
- ✓ Edit or delete tasks.
- ✓ Store tasks using SQLite, Hive, or Firebase for persistence.
- ✓ UI with ListView or GridView for displaying tasks.
- ✓ Optional: Notifications & reminders using flutter_local_notifications.

Output:





8 : CALCULATOR

A **Flutter Calculator App** is a simple arithmetic calculator that allows users to perform basic mathematical operations like addition, subtraction, multiplication, and division.

Key Features:

- ✓ User-friendly UI with buttons for numbers and operations.
- ✓ Performs basic calculations (+, -, ×, ÷).
- ✓ Displays results instantly.
- ✓ Supports clear/reset functionality.
- ✓ Uses TextField or Text widget to display input and output.
- ✓ Optional: Advanced operations (square root, percentage, etc.).

Output:



9:Working With API

In Flutter, working with APIs involves making HTTP requests to fetch, send, or update data from a web server. The http package is commonly used for this purpose.

Steps to Work with API in Flutter

1. Add Dependencies

Add the http package in pubspec.yaml:

dependencies:

http: ^0.13.6

Run:

flutter pub get

2. Import the Package

import 'package:http/http.dart' as http;

import 'dart:convert';

3. Perform API Requests

a) GET Request (Fetching Data)

```
Future<void> fetchData() async {  
  final response = await  
  http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts/1'));
```

```
  if (response.statusCode == 200) {  
    var data = jsonDecode(response.body);  
    print(data);  
  } else {  
    print('Failed to load data');  
  }  
}
```

b) POST Request (Sending Data)

```
Future<void> sendData() async {  
  final response = await http.post(  
    Uri.parse('https://jsonplaceholder.typicode.com/posts'),
```

```

    headers: {'Content-Type': 'application/json'},
    body: jsonEncode({'title': 'Flutter', 'body': 'Working with API', 'userId': 1}),
);

```

```

if (response.statusCode == 201) {
    print('Data sent successfully: ${response.body}');
} else {
    print('Failed to send data');
}
}

```

c) PUT Request (Updating Data)

```

Future<void> updateData() async {
    final response = await http.put(
        Uri.parse('https://jsonplaceholder.typicode.com/posts/1'),
        headers: {'Content-Type': 'application/json'},
        body: jsonEncode({'id': 1, 'title': 'Updated Title', 'body': 'Updated Content', 'userId': 1}),
    );
    if (response.statusCode == 200) {
        print('Data updated successfully: ${response.body}');
    } else {
        print('Failed to update data');
    }
}

```

d) DELETE Request (Deleting Data)

```

Future<void> deleteData() async {
    final response = await
http.delete(Uri.parse('https://jsonplaceholder.typicode.com/posts/1'));

    if (response.statusCode == 200) {
        print('Data deleted successfully');
    }
}

```

```

    } else {
      print('Failed to delete data');
    }
  }
}

```

4. Using APIs in UI (Example)

```

import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';

class ApiExample extends StatefulWidget {
  @override
  _ApiExampleState createState() => _ApiExampleState();
}

class _ApiExampleState extends State<ApiExample> {
  String data = "Fetching data...";

  Future<void> fetchData() async {
    final response = await
http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts/1'));
    if (response.statusCode == 200) {
      setState() {
        data = jsonDecode(response.body)['title'];
      };
    } else {
      setState() {
        data = "Failed to load data";
      };
    }
  }
}

```

```
}  
  
@override  
void initState() {  
  super.initState();  
  fetchData();  
}  
  
@override  
Widget build(BuildContext context) {  
  return Scaffold(  
    appBar: AppBar(title: Text('API Example')),  
    body: Center(child: Text(data)),  
  );  
}  
  
void main() {  
  runApp(MaterialApp(home: ApiExample()));  
}
```


10: NEWS APP

A **Flutter News App** is an application that fetches real-time news articles from an API and displays them in a user-friendly format.

Key Features:

- ✓ **Real-time News Fetching** – Uses APIs like **NewsAPI.org** to fetch news articles.
- ✓ **Category-Based News** – Displays news by categories like Business, Sports, Technology, etc.
- ✓ **Search Functionality** – Allows users to search for specific news.
- ✓ **Beautiful UI** – Uses ListView or GridView to display articles with images and headlines.
- ✓ **Click to Read More** – Opens full news articles in a web view.
- ✓ **Dark Mode Support** – Switch between light and dark themes.

Output:

