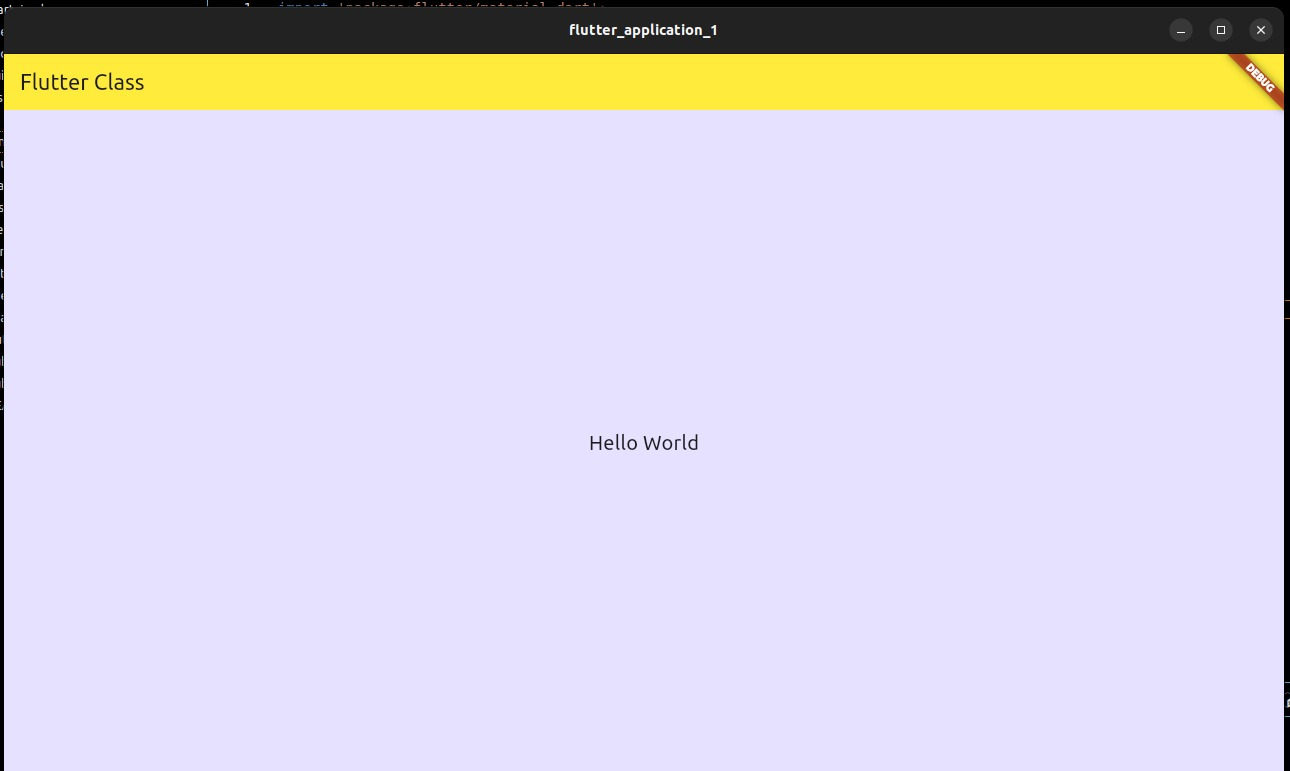
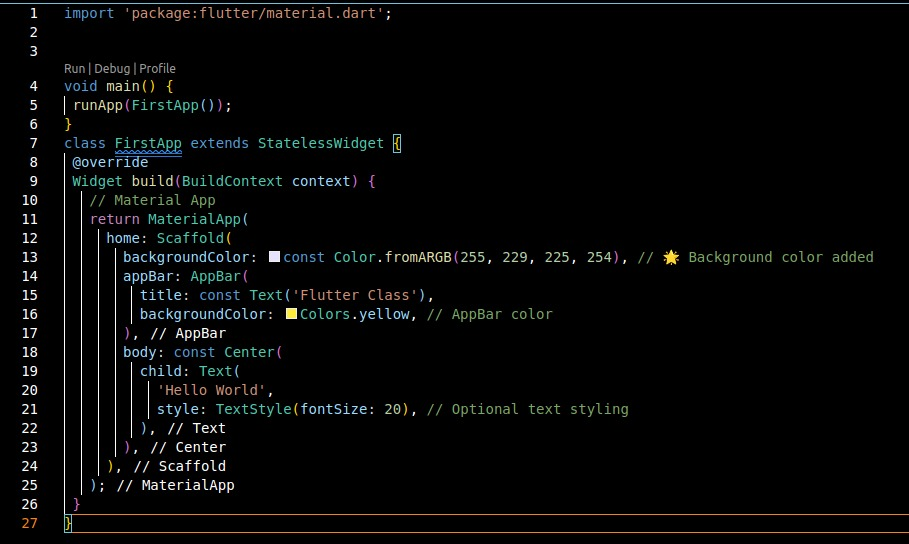
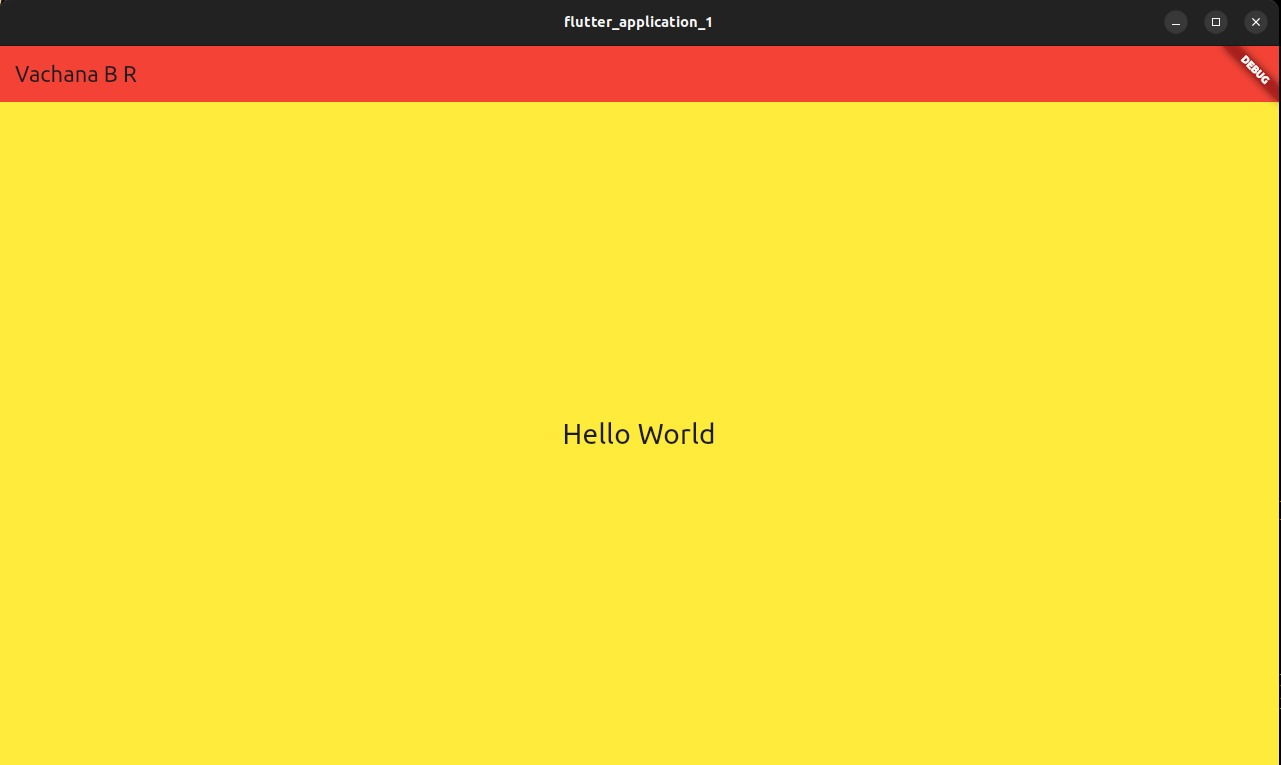
Internship day-2

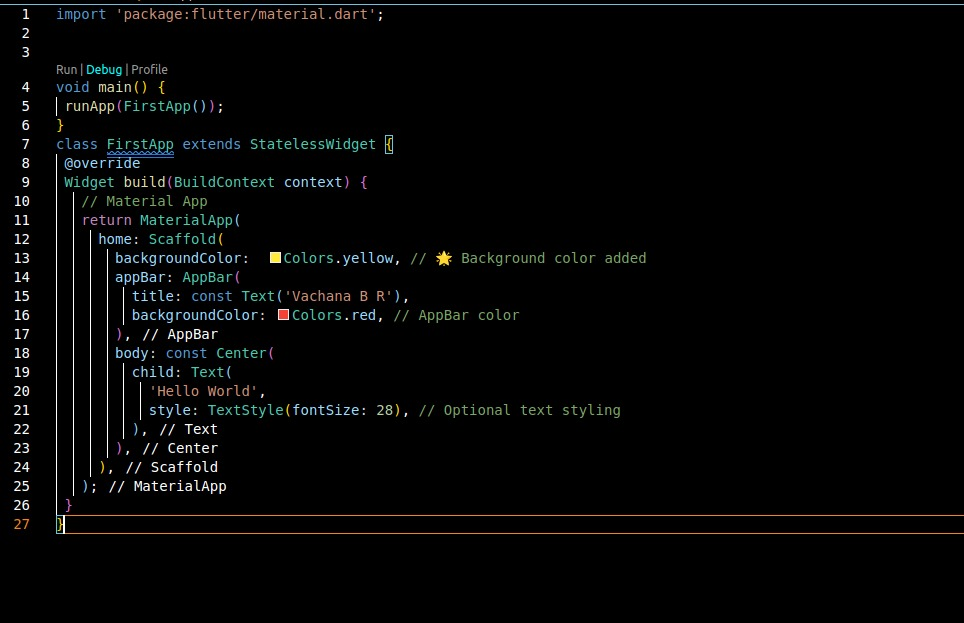
1.



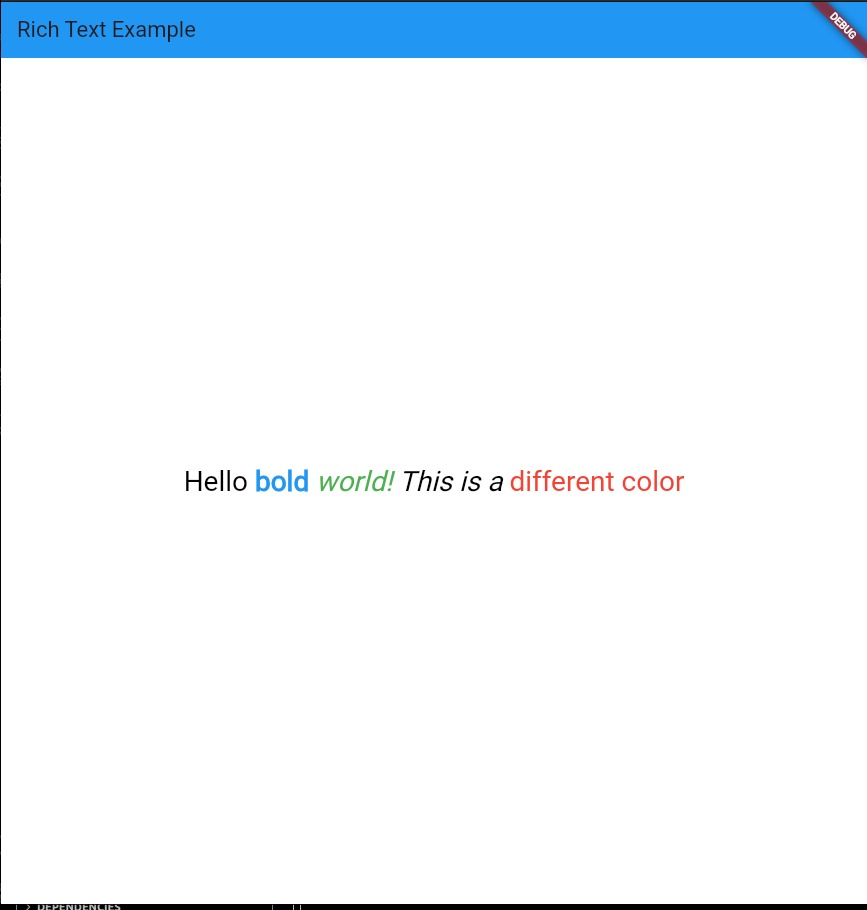


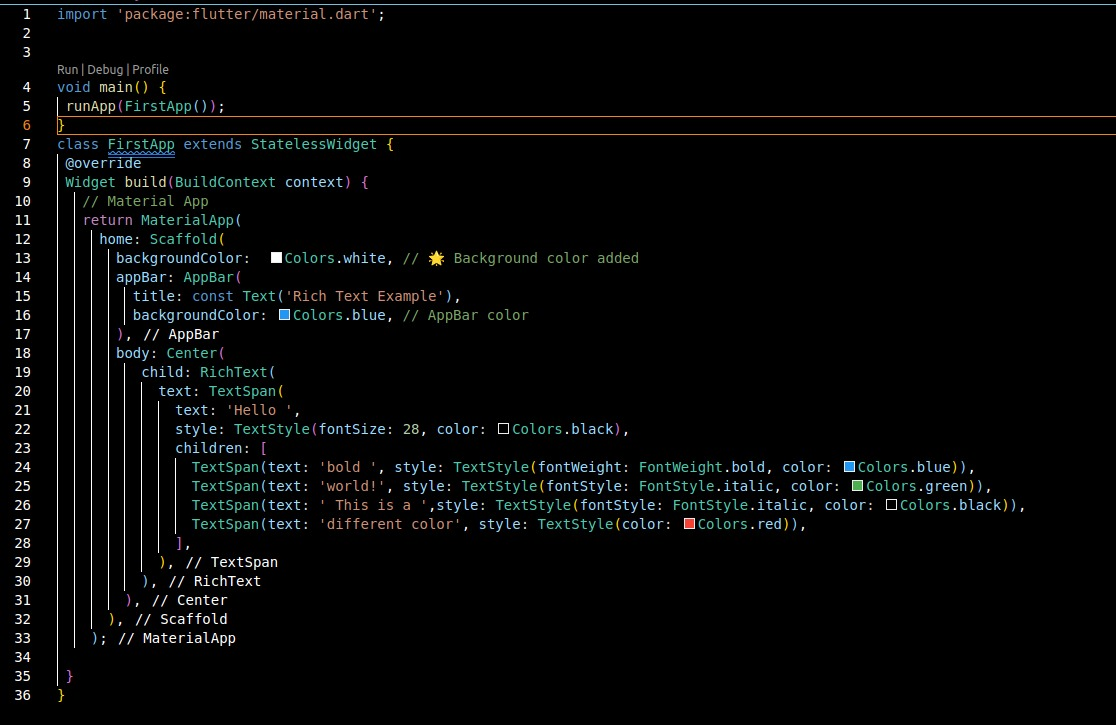
2.



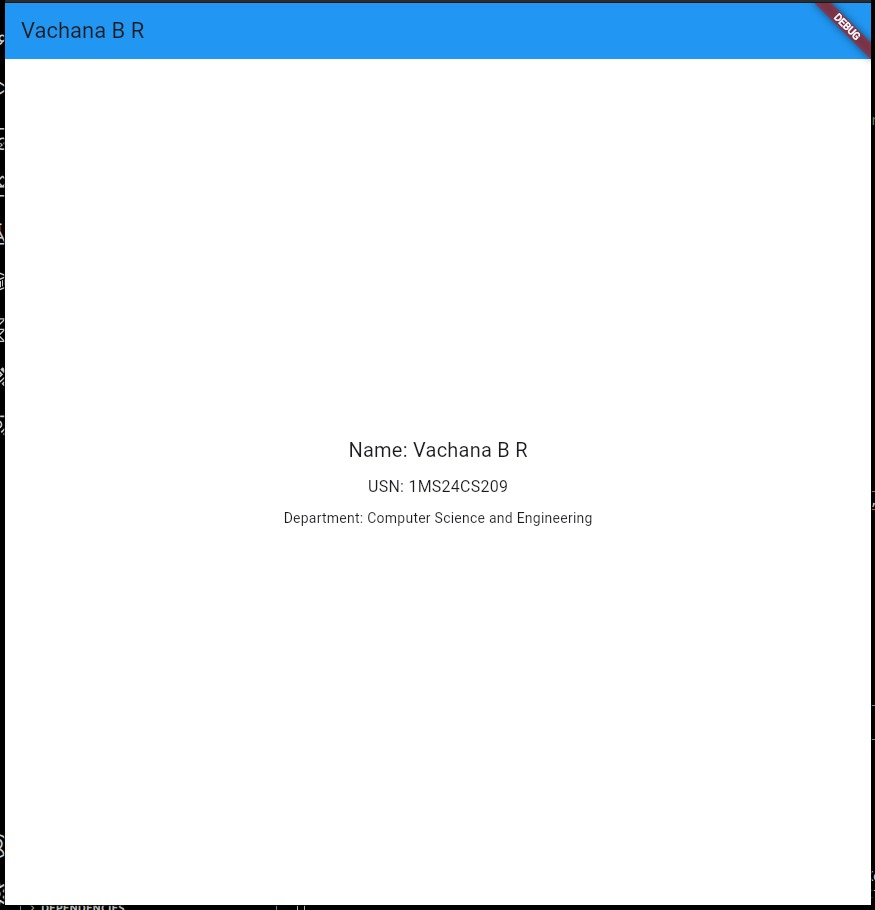


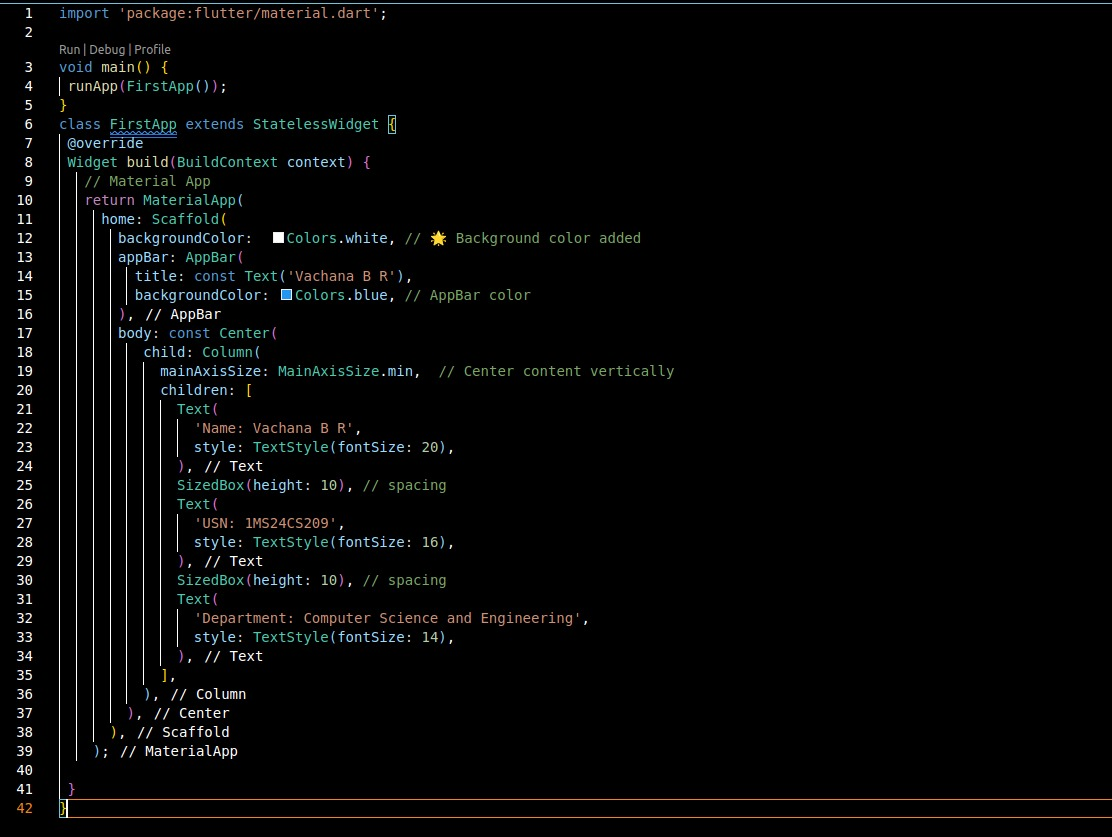
3.



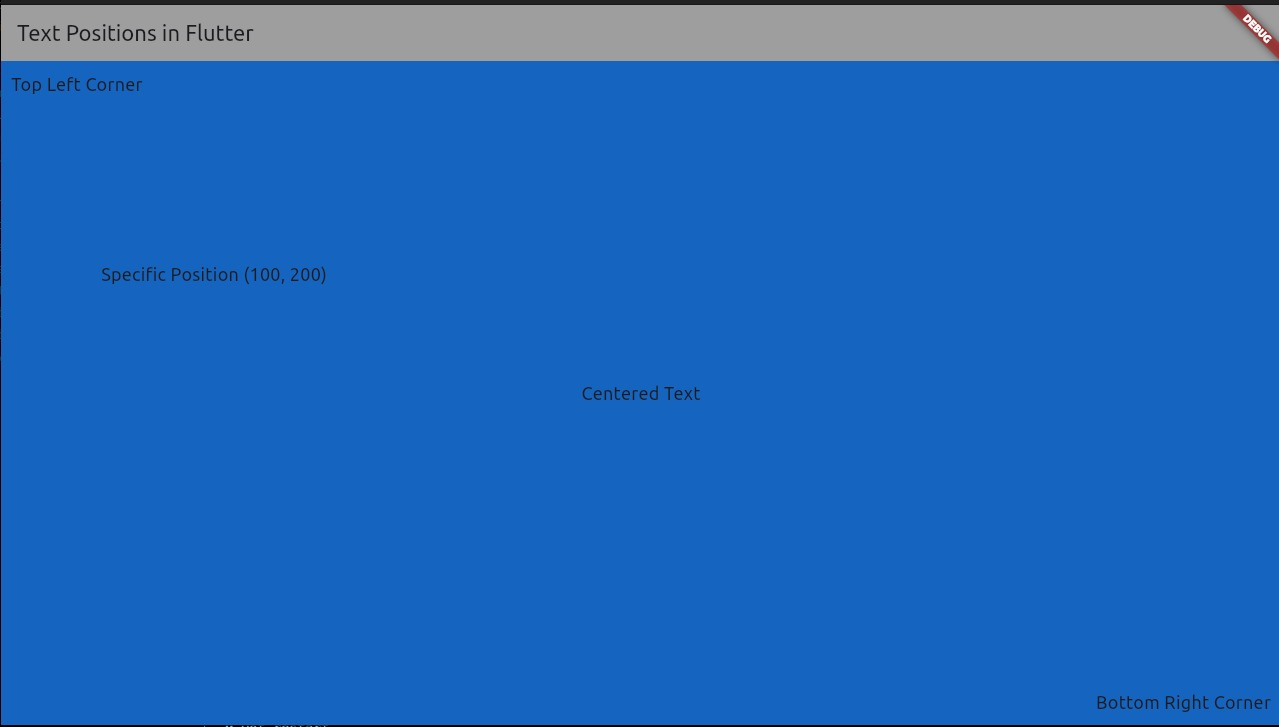


4.



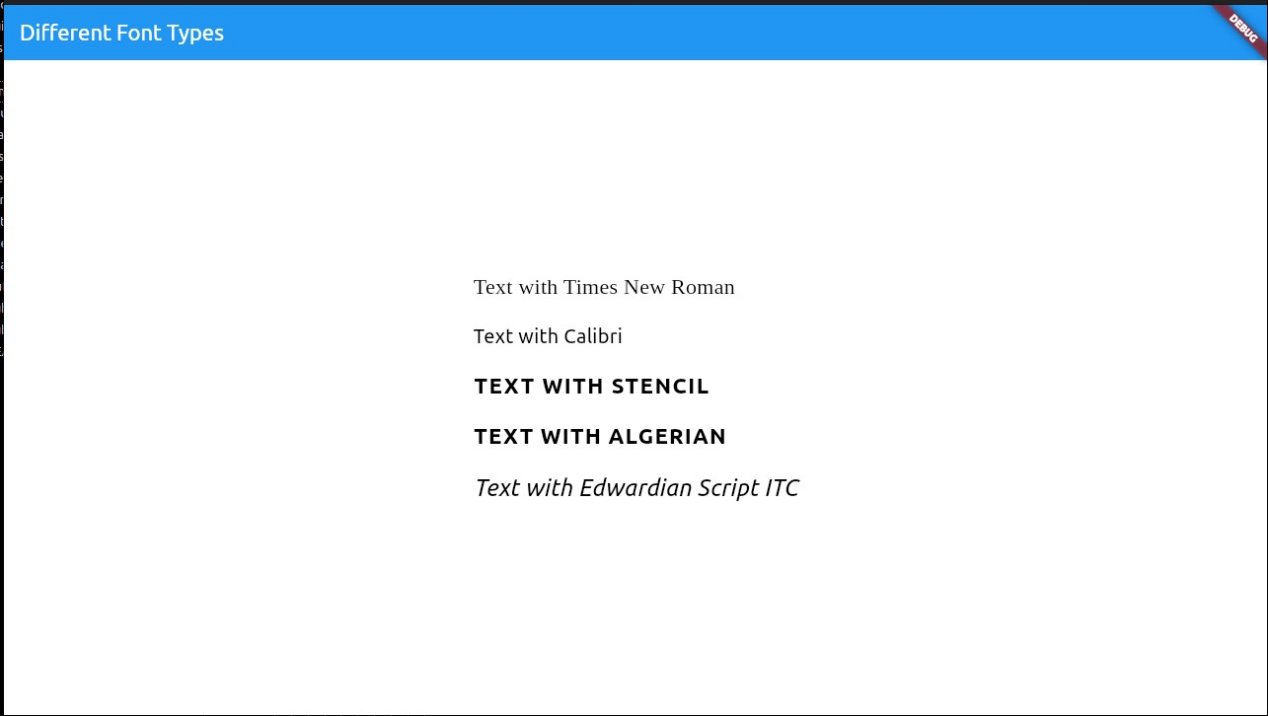


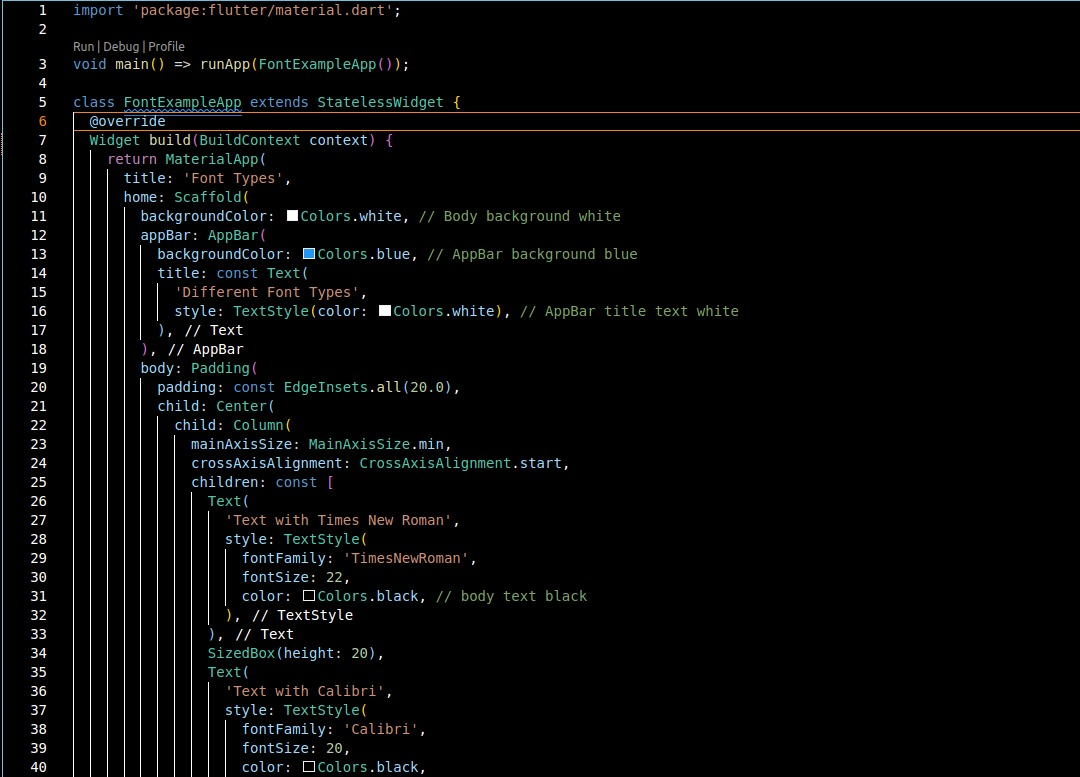
5.

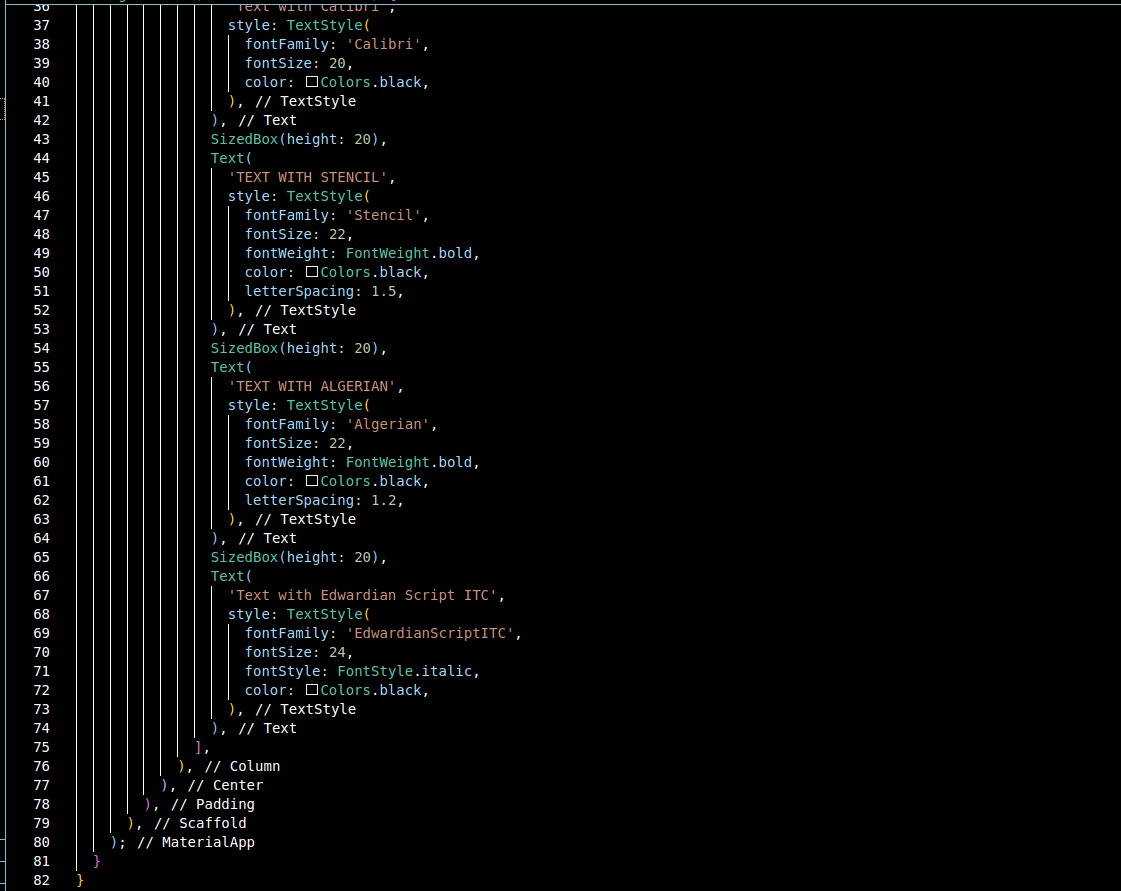




6.

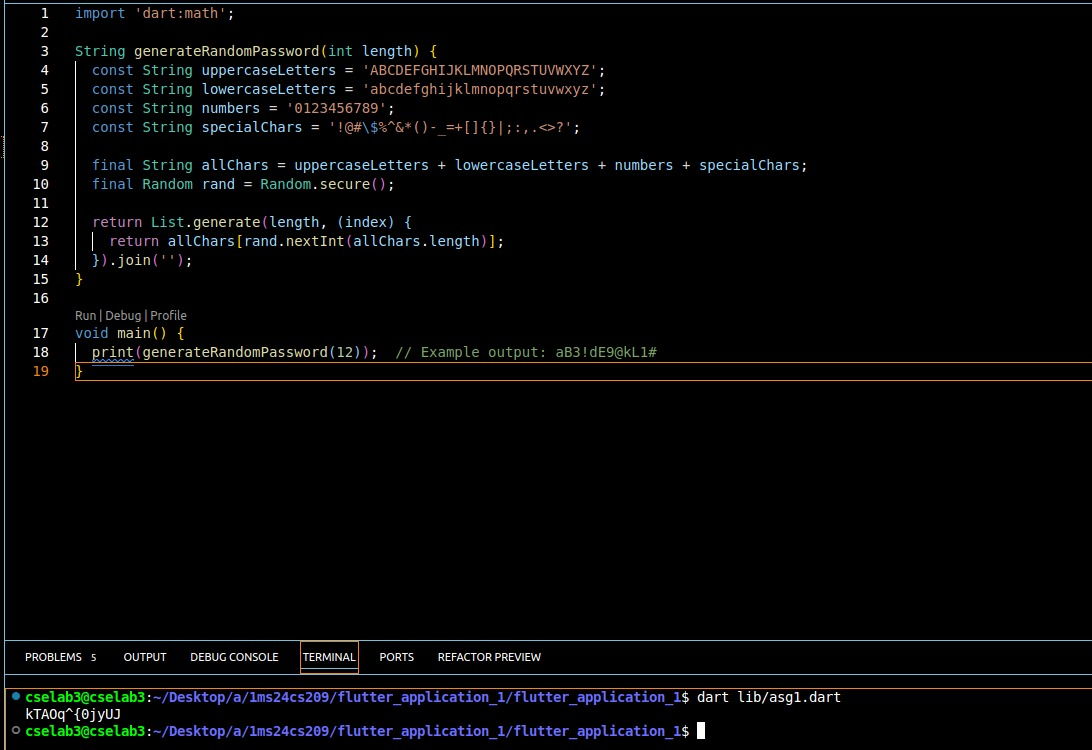




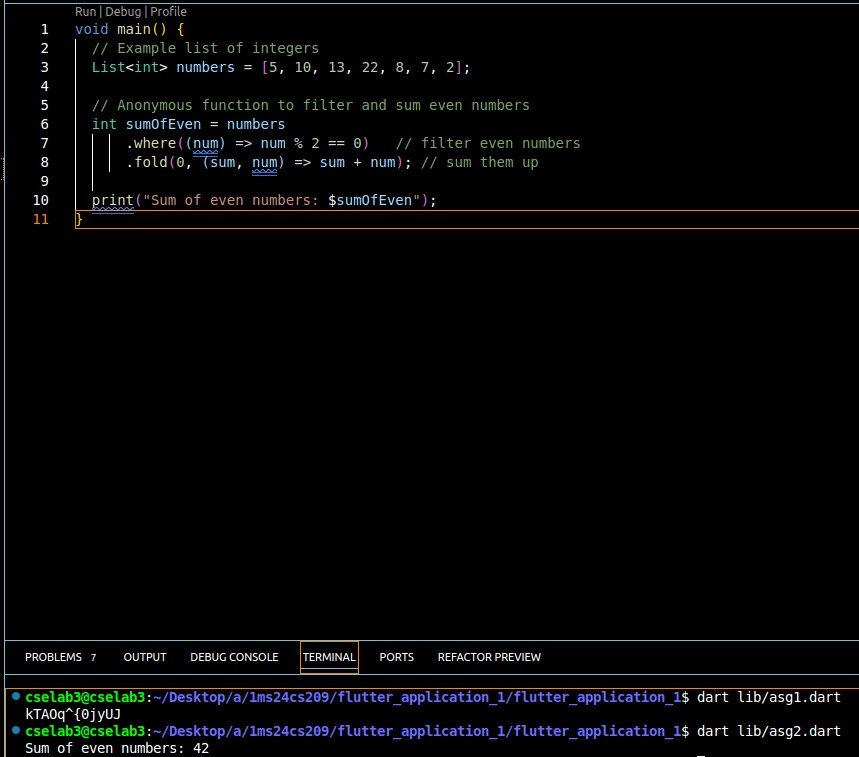


Assignment:

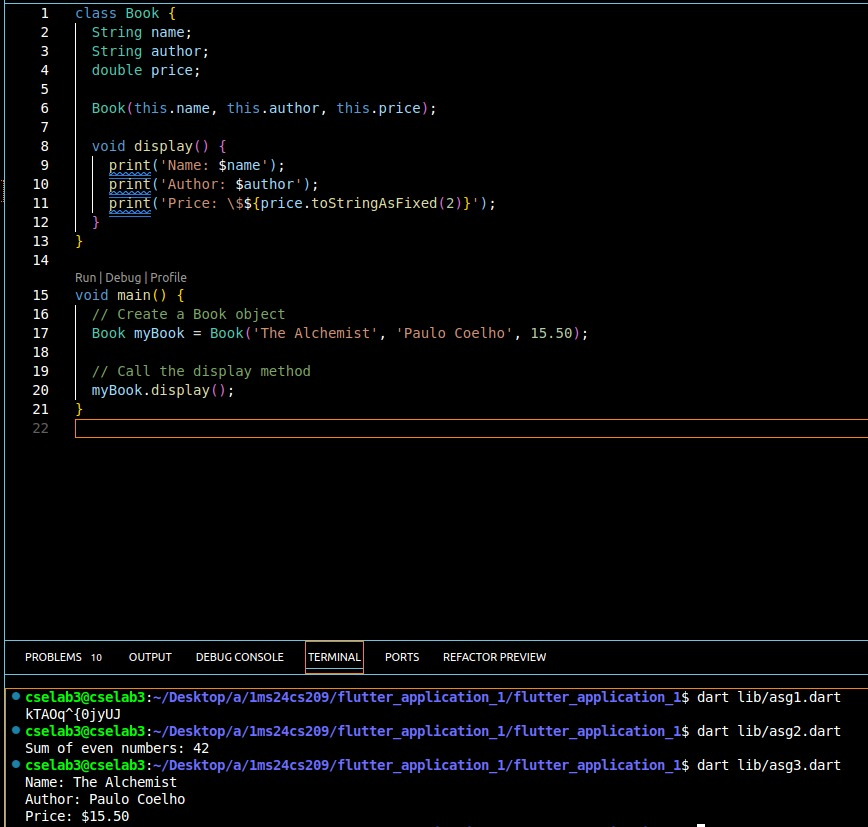
1.



2.



3.



4. Code:

class Vehicle {

String make;

String model;

Vehicle(this.make, this.model);

void displayInfo() {

print('Make: $make');

print('Model: $model');

}

}

// Subclass Car inherits Vehicle

class Car extends Vehicle {

int numberOfDoors;

Car(String make, String model, this.numberOfDoors) : super(make, model);

// Override displayInfo to add numberOfDoors info

@override

void displayInfo() {

super.displayInfo();

print('Number of doors: $numberOfDoors');

}

}

// Subclass Motorcycle inherits Vehicle

class Motorcycle extends Vehicle {

String engineType;

Motorcycle(String make, String model, this.engineType) : super(make, model);

// Override displayInfo to add engineType info

@override

void displayInfo() {

super.displayInfo();

print('Engine type: $engineType');

}

}

void main() {

// Create a Car object

Car car = Car('Toyota', 'Camry', 4);

print('Car details:');

car.displayInfo();

print('\n');

// Create a Motorcycle object

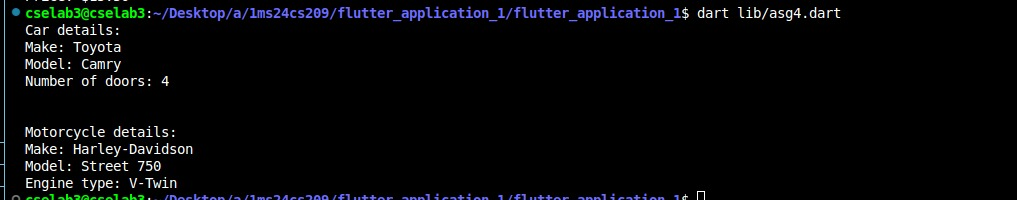
Motorcycle motorcycle = Motorcycle('Harley-Davidson', 'Street 750', 'V-Twin');

print('Motorcycle details:');

motorcycle.displayInfo();

}

Output:



5. Code:

// Base class

class Shape {

void draw() {

print('Drawing a shape.');

}

}

// Subclass Circle

class Circle extends Shape {

@override

void draw() {

print('Drawing a circle.');

}

}

// Subclass Rectangle

class Rectangle extends Shape {

@override

void draw() {

print('Drawing a rectangle.');

}

}

void main() {

// List of shapes with different types

List<Shape> shapes = [

Circle(),

Rectangle(),

Circle(),

];

// Iterate through the list and call draw()

for (var shape in shapes) {

// Identify the type

if (shape is Circle) {

print('Shape type: Circle');

} else if (shape is Rectangle) {

print('Shape type: Rectangle');

} else {

print('Shape type: Unknown');

}

// Call the appropriate draw method (polymorphism)

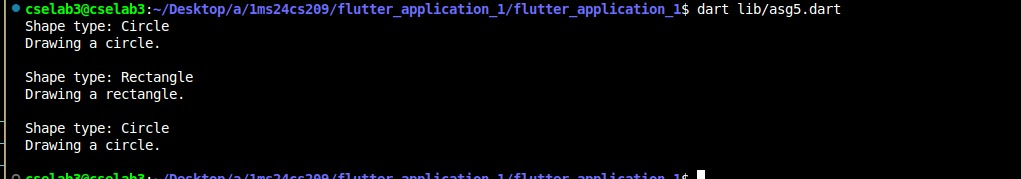
shape.draw();

print(''); // For spacing

}

}

Output:



6. Code:

import 'dart:math';

class PasswordGenerator {

int length;

bool includeSpecial;

String password = '';

PasswordGenerator(this.length, this.includeSpecial);

void generatePassword() {

String letters = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';

String numbers = '0123456789';

String special = '!@#\$%^&\*';

String chars = letters + numbers;

if (includeSpecial) {

chars += special;

}

Random random = Random();

password = '';

for (int i = 0; i < length; i++) {

password += chars[random.nextInt(chars.length)];

}

}

void displayPassword() {

print('Password: $password');

}

}

void main() {

PasswordGenerator gen = PasswordGenerator(10, true);

gen.generatePassword();

gen.displayPassword();

}

Output:



7. Code:

Map<String, int> calculateSumAndProduct(List<int> numbers) {

int sum = 0;

int product = 1;

for (int num in numbers) {

sum += num;

product \*= num;

}

return {

'sum': sum,

'product': product,

};

}

void main() {

List<int> list1 = [2, 3, 5];

List<int> list2 = [1, 4, 6, 8];

print('List 1: $list1 → ${calculateSumAndProduct(list1)}');

print('List 2: $list2 → ${calculateSumAndProduct(list2)}');

}

Output:



8. Code:

class TodoItem {

String title;

String description;

bool isCompleted;

TodoItem(this.title, this.description) : isCompleted = false;

void markCompleted() {

isCompleted = true;

}

@override

String toString() {

return '[${isCompleted ? "✓" : " "}] $title: $description';

}

}

class TodoList {

List<TodoItem> items = [];

void addItem(String title, String description) {

items.add(TodoItem(title, description));

}

void markItemCompleted(int index) {

if (index >= 0 && index < items.length) {

items[index].markCompleted();

} else {

print('Invalid index.');

}

}

void displayItems() {

if (items.isEmpty) {

print('No to-do items.');

} else {

for (int i = 0; i < items.length; i++) {

print('$i. ${items[i]}');

}

}

}

}

void main() {

TodoList myList = TodoList();

myList.addItem('Buy groceries', 'Milk, Eggs, Bread');

myList.addItem('Complete homework', 'Math and Science assignments');

myList.addItem('Call mom', 'Check on her health');

myList.markItemCompleted(1); // Mark 'Complete homework' as done

print('\nMy To-Do List:');

myList.displayItems();

}

Output:

