**EXAMINATION OF Develop Mobile Application using Flutter**

### ****Exam: Basics of Dart****

**Time:** 2 Hours  
**Total Marks:** 100

### ****Section A: Multiple Choice Questions (MCQs) [20 Marks]****

(Each question carries 2 marks)

Which of the following is NOT a feature of Dart?  
a) Strongly Typed  
b) Just-in-Time (JIT) Compilation  
c) Runs only on Android  
d) Supports Asynchronous Programming

What is the primary IDE recommended for Dart development?  
a) Eclipse  
b) Visual Studio Code  
c) NetBeans  
d) Sublime Text

In Dart, which keyword is used to declare a variable?  
a) int  
b) let  
c) var  
d) define

What does Dart SDK include?  
a) Compiler  
b) Debugger  
c) Libraries  
d) All of the above

Which of the following is NOT a valid data type in Dart?  
a) int  
b) float  
c) bool  
d) String

### ****Section B: True or False [10 Marks]****

(Each question carries 2 marks)

1. Dart is primarily used for backend development. (True/False)
2. Functions in Dart must always return a value. (True/False)
3. The switch statement in Dart supports strings as case values. (True/False)
4. Dart is a statically typed language. (True/False)
5. The Dart language supports both synchronous and asynchronous programming. (True/False)

### ****Section C: Short Answer Questions [10 Marks]****

(Each question carries 5 marks)

1. Explain the difference between Native Apps and Cross-Platform Apps.
2. What are the advantages of using Dart for mobile development?

### ****Section D: Installation & Setup [10 Marks]****

(Each question carries 5 marks)

1. List the steps to install Dart SDK on Windows and macOS.
2. How do you integrate Dart with Visual Studio Code?

### ****Section E: Variables & Data Types [10 Marks]****

(Each question carries 5 marks)

1. Write a Dart program that declares a variable of each basic data type (int, double, String, bool).
2. Explain the naming conventions for variables in Dart with examples.

### ****Section F: Control Flow Structures [20 Marks]****

(Each question carries 5 marks)

1. Write a Dart program using an if-else statement to check whether a number is even or odd.
2. Write a Dart program using a switch statement to print the day of the week based on a given number (1 for Monday, 2 for Tuesday, etc.).
3. Write a Dart program that uses a for loop to print numbers from 1 to 10.
4. Write a Dart program using a while loop to find the sum of the first 5 natural numbers.

### ****Section G: Functions in Dart [20 Marks]****

(Each question carries 5 marks)

1. Write a function in Dart that takes two numbers as parameters and returns their sum.
2. What is the difference between built-in functions and user-defined functions in Dart? Provide an example.
3. Write a function in Dart that takes a name as a parameter and prints a greeting message.
4. What are the different ways to pass parameters in Dart functions? Explain with examples.

### ****Solutions (Sample Answers)****

#### ****Section A (MCQ Solutions)****

1. c) Runs only on Android
2. b) Visual Studio Code
3. c) var
4. d) All of the above
5. b) float

#### ****Section B (True/False Solutions)****

1. False
2. False
3. True
4. True
5. True

#### ****Section C (Short Answer Solutions)****

1. **Native Apps** are built specifically for one platform (e.g., Android or iOS) using platform-specific languages like Swift or Kotlin. **Cross-Platform Apps** use frameworks like Flutter to run on multiple platforms with a single codebase.
2. Dart offers fast execution, JIT and AOT compilation, strong typing, and excellent support for UI development with Flutter.

#### ****Section D (Installation & Setup Solutions)****

1. **Windows**: Download Dart SDK → Extract files → Add path to system variables.  
   **MacOS**: Use Homebrew (brew install dart).
2. Install Dart extension in VS Code → Set up launch.json → Verify installation using dart --version.

#### ****Section E (Variables & Data Types Solution)****

dart

CopyEdit

void main() {

int age = 25;

double height = 5.9;

String name = "Alice";

bool isStudent = true;

print("Name: $name, Age: $age, Height: $height, Student: $isStudent");

}

**Naming Conventions**: Use camelCase (myVariable), avoid special characters, use meaningful names.

#### ****Section F (Control Flow Solutions)****

1. **If-Else Statement**

dart

CopyEdit

void main() {

int num = 10;

if (num % 2 == 0) {

print("$num is even");

} else {

print("$num is odd");

}

}

1. **Switch Statement**

dart

CopyEdit

void main() {

int day = 3;

switch (day) {

case 1:

print("Monday");

break;

case 2:

print("Tuesday");

break;

case 3:

print("Wednesday");

break;

default:

print("Invalid Day");

}

}

1. **For Loop**

dart

CopyEdit

void main() {

for (int i = 1; i <= 10; i++) {

print(i);

}

}

1. **While Loop**

dart

CopyEdit

void main() {

int sum = 0, i = 1;

while (i <= 5) {

sum += i;

i++;

}

print("Sum of first 5 natural numbers: $sum");

}

#### ****Section G (Functions Solutions)****

1. **Function to Add Two Numbers**

dart

CopyEdit

int add(int a, int b) {

return a + b;

}

1. **Built-in vs. User-defined Functions**: Built-in functions are pre-defined (e.g., print()), whereas user-defined functions are created by the developer.
2. **Greeting Function**

dart

CopyEdit

void greet(String name) {

print("Hello, $name!");

}

1. **Passing Parameters**

dart

CopyEdit

// Positional parameters

void greet(String name) => print("Hello, $name!");

// Named parameters

void greetUser({String name = "Guest"}) => print("Hello, $name!");

// Optional parameters

void sayHello([String? name]) => print("Hello, ${name ?? 'Stranger'}!");

This exam covers both theoretical and practical aspects of Dart fundamentals. Do you need any modifications or additional topics? 🚀

4o

##### You said:

please try to explain solution

###### ChatGPT said:

Sure! Let's go over the solutions with explanations to ensure you fully understand each concept.

### ****Section A: Multiple Choice Questions (MCQs) - Solutions with Explanations****

**Answer: c) Runs only on Android**

* 1. Dart is not limited to Android. It is a cross-platform language used for both Android and iOS development with Flutter.

**Answer: b) Visual Studio Code**

* 1. While Dart can be used in different IDEs, **Visual Studio Code** is one of the most popular choices because of its lightweight design and Dart extension support.

**Answer: c) var**

* 1. In Dart, the var keyword is used to declare a variable without specifying its type. Dart infers the type automatically.  
     Example:

dart

CopyEdit

var name = "Alice"; // Dart infers that 'name' is a String

**Answer: d) All of the above**

* 1. Dart SDK includes a compiler to compile code, a debugger to test and fix issues, and built-in libraries for efficient programming.

**Answer: b) float**

* 1. Dart does not have a float data type. Instead, it uses double for decimal values.

### ****Section B: True/False - Solutions with Explanations****

**False**

* 1. Dart is primarily used for frontend (UI) development in Flutter, not backend development. However, it can be used for the backend using frameworks like Dart Frog.

**False**

* 1. Functions in Dart can return a value or be void (no return value).  
     Example:

dart

CopyEdit

void sayHello() {

print("Hello!");

}

**True**

* 1. The switch statement in Dart supports strings, numbers, and even enums.  
     Example:

dart

CopyEdit

switch ("Monday") {

case "Monday":

print("Start of the week!");

break;

}

**True**

* 1. Dart is a statically typed language, meaning variable types are determined at compile time.

**True**

* Dart supports **synchronous** (executed in order) and **asynchronous** (executed in the background) programming.

### ****Section C: Short Answer Solutions****

1. **Difference Between Native Apps and Cross-Platform Apps**

* **Native Apps**: Built specifically for one platform (Android - Kotlin/Java, iOS - Swift).
* **Cross-Platform Apps**: Work on multiple platforms with a single codebase (e.g., Flutter uses Dart).

1. **Advantages of Dart for Mobile Development**

* **Fast execution** due to Just-in-Time (JIT) and Ahead-of-Time (AOT) compilation.
* **Single codebase** for both Android & iOS.
* **Hot Reload** for real-time changes in the UI.
* **Rich UI support** with Flutter widgets.

### ****Section D: Installation & Setup Solutions****

1. **Installing Dart SDK**

* **Windows**:
  1. Download Dart SDK from the official website.
  2. Extract files and set the environment path.
  3. Verify installation with dart --version.
* **MacOS**:
  1. Use Homebrew:

sh

CopyEdit

brew install dart

* 1. Verify with dart --version.

1. **Integrating Dart with VS Code**

* Install the **Dart extension** from the marketplace.
* Open a .dart file and start coding.
* Use Run to execute Dart programs.

### ****Section E: Variables & Data Types Solutions****

1. **Dart Program for Different Data Types**

dart

CopyEdit

void main() {

int age = 25;

double height = 5.9;

String name = "Alice";

bool isStudent = true;

print("Name: $name, Age: $age, Height: $height, Student: $isStudent");

}

* **int**: Stores whole numbers.
* **double**: Stores decimal numbers.
* **String**: Stores text.
* **bool**: Stores true/false values.

1. **Variable Naming Conventions**

* Use meaningful names (userName, not x).
* Use **camelCase** (myVariable).
* Avoid using reserved keywords (int, String, etc.).

### ****Section F: Control Flow Solutions****

1. **If-Else Statement to Check Even/Odd**

dart

CopyEdit

void main() {

int num = 10;

if (num % 2 == 0) {

print("$num is even");

} else {

print("$num is odd");

}

}

* Uses % (modulus operator) to check divisibility by 2.

1. **Switch Statement for Days of the Week**

dart

CopyEdit

void main() {

int day = 3;

switch (day) {

case 1:

print("Monday");

break;

case 2:

print("Tuesday");

break;

case 3:

print("Wednesday");

break;

default:

print("Invalid Day");

}

}

* **Break** stops execution after a match.

1. **For Loop to Print Numbers 1 to 10**

dart

CopyEdit

void main() {

for (int i = 1; i <= 10; i++) {

print(i);

}

}

* Runs from 1 to 10.

1. **While Loop to Find Sum of First 5 Natural Numbers**

dart

CopyEdit

void main() {

int sum = 0, i = 1;

while (i <= 5) {

sum += i;

i++;

}

print("Sum of first 5 natural numbers: $sum");

}

* Adds numbers from 1 to 5.

### ****Section G: Functions in Dart Solutions****

1. **Function to Add Two Numbers**

dart

CopyEdit

int add(int a, int b) {

return a + b;

}

* Takes **two parameters**, returns their sum.

1. **Difference Between Built-in and User-defined Functions**

* **Built-in**: Predefined in Dart (e.g., print(), sqrt()).
* **User-defined**: Created by programmers (add(int a, int b)).

1. **Function to Print Greeting**

dart

CopyEdit

void greet(String name) {

print("Hello, $name!");

}

* Calls function as: greet("Alice");.

1. **Different Ways to Pass Parameters**

dart

CopyEdit

// Positional parameters

void greet(String name) => print("Hello, $name!");

// Named parameters

void greetUser({String name = "Guest"}) => print("Hello, $name!");

// Optional parameters

void sayHello([String? name]) => print("Hello, ${name ?? 'Stranger'}!");

* **Positional**: Arguments are passed in order.
* **Named**: Uses {} to specify parameters.
* **Optional**: Uses [], allows skipping parameters.

### ****Conclusion****

This exam covered **Dart basics**, including **variables, data types, control flow, functions, and installation**. Let me know if you need more explanations or additional practice! 🚀

4o

##### You said:

● Applying Object Oriented Programming (OOP) ✓ Classes and Objects ✓ Inheritance ✓ Polymorphism ✓ Encapsulation ✓ Abstraction ● Using dart libraries and packages ✓ Importing and using libraries ✓ Exploring built-in Dart libraries ✓ Managing dependencies with pub (Dart's package manager) ✓ Using external packages for enhanced functionality

###### ChatGPT said:

Now, let's prepare an exam covering **Object-Oriented Programming (OOP) in Dart** and **Dart libraries & packages**. This will include multiple-choice questions, coding exercises, and explanations.

## ****Exam: Object-Oriented Programming & Dart Libraries****

**Time:** 2 Hours  
**Total Marks:** 100

### ****Section A: Multiple Choice Questions (MCQs) [20 Marks]****

(Each question carries 2 marks)

What is the main purpose of **encapsulation** in OOP?  
a) Allow global access to variables  
b) Hide data and provide controlled access  
c) Increase the speed of the program  
d) None of the above

**Polymorphism** allows:  
a) A single function to behave differently based on input  
b) One class to inherit from multiple classes  
c) Restrict the modification of variables  
d) Create objects without classes

Which keyword is used to **inherit** a class in Dart?  
a) extends  
b) implements  
c) inherits  
d) super

What does the **pubspec.yaml** file do in a Dart project?  
a) Stores all Dart classes  
b) Manages dependencies and packages  
c) Compiles Dart code  
d) Runs the Flutter application

Which of the following is NOT a built-in Dart library?  
a) dart:core  
b) dart:async  
c) dart:math  
d) dart:flutter

### ****Section B: True or False [10 Marks]****

(Each question carries 2 marks)

1. A Dart class can have multiple constructors. (True/False)
2. Abstraction means providing unnecessary details to the user. (True/False)
3. import 'dart:math'; is used to include mathematical functions in Dart. (True/False)
4. The super keyword is used to call a function from the parent class. (True/False)
5. pub is the package manager for Dart. (True/False)

### ****Section C: Short Answer Questions [10 Marks]****

(Each question carries 5 marks)

1. Explain the difference between **Encapsulation** and **Abstraction** in Dart.
2. What are the benefits of using external Dart packages?

### ****Section D: OOP Concepts (Coding) [20 Marks]****

(Each question carries 5 marks)

1. Write a **Dart class** named Car with properties brand, model, and a method displayInfo().
2. Implement **inheritance**: Create a Car class and a ElectricCar class that extends Car.
3. Implement **polymorphism**: Create a Shape class with a calculateArea() method. Derive two classes, Circle and Rectangle, that override the method.
4. Implement **encapsulation** by making a variable private and providing getter and setter methods.

### ****Section E: Dart Libraries & Packages [20 Marks]****

(Each question carries 5 marks)

1. Write a Dart program that imports dart:math and generates a random number.
2. Explain how pubspec.yaml helps in dependency management.
3. How do you install and use an external Dart package? Give an example.
4. Write a Dart program that uses the **http** package to fetch data from an API.

## ****Solutions and Explanations****

### ****Section A (MCQ Solutions & Explanations)****

**Answer: b) Hide data and provide controlled access**

* + **Encapsulation** prevents direct modification of object properties and ensures controlled access.

**Answer: a) A single function to behave differently based on input**

* + **Polymorphism** allows a method to have different behaviors in different classes.

**Answer: a) extends**

* + The extends keyword is used for **class inheritance** in Dart.

**Answer: b) Manages dependencies and packages**

* + The **pubspec.yaml** file keeps track of dependencies in a Dart/Flutter project.

**Answer: d) dart:flutter**

* + dart:flutter is **not a built-in Dart library**; it comes with Flutter.

### ****Section B (True/False Solutions & Explanations)****

1. **True** - Dart supports multiple **constructors**, including named and factory constructors.
2. **False** - **Abstraction** hides unnecessary details and only shows essential features.
3. **True** - dart:math contains **math functions** like sqrt(), pi, and Random().
4. **True** - super calls a function from the parent class.
5. **True** - pub is the **Dart package manager** for installing external packages.

### ****Section C (Short Answer Solutions)****

1. **Encapsulation vs. Abstraction**

* **Encapsulation**: Hides data and provides access via methods.
* **Abstraction**: Hides implementation details while exposing only the necessary functionality.

1. **Benefits of External Dart Packages**

* Saves development time.
* Provides well-tested, reusable code.
* Easily integrated via pub.dev.

### ****Section D: OOP Coding Solutions****

1. **Class with Properties and Methods**

dart

CopyEdit

class Car {

String brand;

String model;

Car(this.brand, this.model);

void displayInfo() {

print("Car: $brand $model");

}

}

1. **Inheritance Example**

dart

CopyEdit

class Car {

String brand;

Car(this.brand);

}

class ElectricCar extends Car {

int batteryCapacity;

ElectricCar(String brand, this.batteryCapacity) : super(brand);

}

1. **Polymorphism Example**

dart

CopyEdit

abstract class Shape {

double calculateArea();

}

class Circle extends Shape {

double radius;

Circle(this.radius);

@override

double calculateArea() => 3.14 \* radius \* radius;

}

1. **Encapsulation Example**

dart

CopyEdit

class Person {

String \_name; // Private variable

void setName(String name) => \_name = name;

String getName() => \_name;

}

### ****Section E: Dart Libraries & Packages Solutions****

1. **Using** dart:math **to Generate Random Numbers**

dart

CopyEdit

import 'dart:math';

void main() {

var random = Random();

print(random.nextInt(100)); // Generates a random number between 0-99

}

1. **pubspec.yaml for Dependency Management**

* **Lists dependencies** needed for a project.
* **Manages versions** of installed packages.
* **Example file:**

yaml

CopyEdit

dependencies:

http: ^0.13.3

1. **Installing and Using an External Dart Package**

* Install package:

sh

CopyEdit

dart pub add http

* Import and use in code:

dart

CopyEdit

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

1. **Fetching API Data with** http **Package**

dart

CopyEdit

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

import 'dart:convert';

void main() async {

var response = await http.get(Uri.parse("https://jsonplaceholder.typicode.com/posts/1"));

print(jsonDecode(response.body));

}

## ****Final Thoughts****

This exam covers **OOP principles in Dart** and **Dart libraries**. You get a mix of **theory and hands-on coding** to strengthen your understanding. Let me know if you need more explanations or practice problems! 🚀

4o

##### You said:

Learning outcome 2: Implement UI designs Learning hours: 30 Indicative content ● Preparation of flutter environment ✓ Introduction to Flutter framework Definition Purpose Features ✓ Widgets Definition Types Widget Lifecycle ✓ State management Definition Packages Libraries Methods ✓ Environment Set Up Installation of Flutter SDK Installation of IDE (Android Studio, XCode) Configuration of development environment ✓ Creating a new Flutter project ● Applying flutter's widget system ✓ Stateful and stateless ✓ Widget Tree and Hierarchy Parent-child relationships in Flutter Widget composition Row and Column Container Expanded Stack ✓ Core widgets Text and Styling Image and Asset Interactive (Buttons, Gesture). Layout Widgets ● Implementation of state management. ✓ Using packages. GetX package Provider package ✓ Using pattern Redux pattern Business Logic Component (BLoC) pattern ✓ Using setState() method ✓ Using the riverpod solution. ✓ Using navigation and routing. Navigator Route BottomNavigationBar TabBar and TabBarView ● Using pre-designed widgets ✓ Material Design Widgets ✓ Cupertino Widgets ✓ Flutter Icons ✓ Third-Party Packages Resources required for the indicative content Equipment • Computer (Windows, MacOS), Mobile Phone (Android, iPhone) Materials • Material Design components, Flutter Icons Tools • Internet, VS Code, Android Studio, Xcode, Emulators, Flutter SDK Facilitation techniques or Learning activity • Demonstration, • Simulation • Group w

###### ChatGPT said:

Let's create an **exam** covering **UI Design in Flutter**, including **MCQs, coding exercises, and explanations** based on the provided learning outcomes.

## ****Exam: Implementing UI Designs in Flutter****

**Time:** 2 Hours  
**Total Marks:** 100

### ****Section A: Multiple Choice Questions (MCQs) [20 Marks]****

(Each question carries 2 marks)

What is the main purpose of **Flutter**?  
a) Backend development  
b) Frontend UI development for cross-platform apps  
c) Database management  
d) Web server hosting

Which widget type **does not change its state** once created?  
a) StatelessWidget  
b) StatefulWidget  
c) StreamBuilder  
d) AnimatedWidget

What is the function of setState() in a **StatefulWidget**?  
a) To initialize widgets  
b) To manage navigation  
c) To update UI when state changes  
d) To call backend APIs

Which package is NOT used for **state management** in Flutter?  
a) Provider  
b) Redux  
c) SQLite  
d) GetX

What is the function of BottomNavigationBar in Flutter?  
a) Displaying images  
b) Managing multiple screens  
c) Creating a side menu  
d) Handling API calls

### ****Section B: True or False [10 Marks]****

(Each question carries 2 marks)

1. Flutter uses **Dart** as its programming language. (True/False)
2. A **Row** widget in Flutter arranges children **vertically**. (True/False)
3. The **Navigator** widget in Flutter is used for **page navigation**. (True/False)
4. Expanded is used inside **Row and Column** widgets to allocate flexible space. (True/False)
5. **Material Design** widgets are only available on iOS. (True/False)

### ****Section C: Short Answer Questions [10 Marks]****

(Each question carries 5 marks)

1. Explain the difference between **StatefulWidget** and **StatelessWidget** with an example.
2. What are the advantages of **Flutter’s widget system**?

### ****Section D: UI Development in Flutter (Coding) [20 Marks]****

(Each question carries 5 marks)

1. Write a **Flutter program** that displays "Hello, Flutter!" using a **Text widget** with styling.
2. Create a **Row widget** containing three icons (Home, Search, Profile).
3. Write a **Flutter StatefulWidget** that updates a counter using a FloatingActionButton.
4. Implement **Navigation** between two screens in Flutter.

### ****Section E: State Management [20 Marks]****

(Each question carries 5 marks)

1. Explain **state management** and why it is important in Flutter.
2. Write a Flutter example using **setState()** to update UI dynamically.
3. How does the **Provider package** help manage state in Flutter?
4. Write a basic **GetX state management** example in Flutter.

### ****Section F: Advanced UI Concepts & Packages [20 Marks]****

(Each question carries 5 marks)

1. Explain the role of **Material Design Widgets** in Flutter.
2. What is the difference between **Material Design** and **Cupertino Widgets**?
3. Install and use the google\_fonts package in a Flutter app.
4. Explain **TabBar and TabBarView** with an example.

## ****Solutions and Explanations****

### ****Section A (MCQ Solutions & Explanations)****

**Answer: b) Frontend UI development for cross-platform apps**

* + Flutter is mainly used for building **mobile and web UIs** using a single codebase.

**Answer: a) StatelessWidget**

* + **StatelessWidget** cannot change once created, unlike StatefulWidget.

**Answer: c) To update UI when state changes**

* + setState() triggers a UI **rebuild** whenever the state changes.

**Answer: c) SQLite**

* + **SQLite** is used for database management, not state management.

**Answer: b) Managing multiple screens**

* + BottomNavigationBar helps switch between different screens in an app.

### ****Section B (True/False Solutions & Explanations)****

1. **True** - Flutter uses Dart for app development.
2. **False** - **Row** arranges items **horizontally**, not vertically.
3. **True** - **Navigator** handles screen transitions.
4. **True** - Expanded takes up available space inside **Row and Column**.
5. **False** - Material Design works on both **Android and iOS**.

### ****Section C (Short Answer Solutions)****

1. **Stateful vs. Stateless Widget**

* **StatelessWidget**: UI does not change after rendering.
* **StatefulWidget**: UI changes dynamically using setState().
* Example:

dart

CopyEdit

class MyWidget extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Text("Stateless Widget");

}

}

1. **Advantages of Flutter’s Widget System**

* **Fast UI development**
* **Customizable and reusable widgets**
* **Consistent UI across platforms**

### ****Section D: UI Development in Flutter (Coding Solutions)****

1. **Text Widget with Styling**

dart

CopyEdit

Text(

"Hello, Flutter!",

style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold, color: Colors.blue),

)

1. **Row with Icons**

dart

CopyEdit

Row(

mainAxisAlignment: MainAxisAlignment.spaceAround,

children: [

Icon(Icons.home),

Icon(Icons.search),

Icon(Icons.person),

],

)

1. **Counter App using StatefulWidget**

dart

CopyEdit

class CounterScreen extends StatefulWidget {

@override

\_CounterScreenState createState() => \_CounterScreenState();

}

class \_CounterScreenState extends State<CounterScreen> {

int count = 0;

void increment() {

setState(() {

count++;

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

body: Center(child: Text('Counter: $count')),

floatingActionButton: FloatingActionButton(

onPressed: increment,

child: Icon(Icons.add),

),

);

}

}

1. **Navigation Between Screens**

dart

CopyEdit

Navigator.push(

context,

MaterialPageRoute(builder: (context) => SecondScreen()),

);

### ****Section E: State Management Solutions****

1. **Why State Management?**

* Controls **dynamic UI updates**.
* Improves app **performance and efficiency**.

1. **setState() Example**

dart

CopyEdit

setState(() {

count++;

});

1. **Provider Package**

* Manages **global state** without rebuilding unnecessary widgets.

1. **GetX Example**

dart

CopyEdit

class CounterController extends GetxController {

var count = 0.obs;

void increment() => count++;

}

### ****Section F: Advanced UI Concepts & Packages Solutions****

1. **Material Design Widgets**

* Provides **modern UI components** like buttons, dialogs, and cards.

1. **Material vs. Cupertino Widgets**

* **Material Design** (Google) → Android Look
* **Cupertino** (Apple) → iOS Look

1. **Using** google\_fonts **Package**

yaml

CopyEdit

dependencies:

google\_fonts: ^4.0.0

1. **TabBar & TabBarView Example**

dart

CopyEdit

DefaultTabController(

length: 2,

child: Scaffold(

appBar: AppBar(

bottom: TabBar(

tabs: [Tab(text: "Tab 1"), Tab(text: "Tab 2")],

),

),

body: TabBarView(

children: [Text("Content 1"), Text("Content 2")],

),

),

)

## ****Final Thoughts****

This exam covers **UI development in Flutter**, **widget system**, **state management**, and **pre-designed widgets**. Let me know if you need additional practice! 🚀

4o

##### You said:

Learning outcome 3: Integrate backend functionality. Learning hours: 30 Indicative content ● Integration of external services ✓ Description of HTTP Requests GET POST PUT DELETE UPDATE PATCH ✓ Adding dependencies ✓ Adding calls ✓ Handle Responses ✓ Parsing JSON Data ✓ Perform Authentication and Authorization ✓ Push notifications(firebase) ✓ Implement Security measures Secure Data Storage Secure Network Communication Input Validation and Output Encoding ● Implement storage management. ✓ Data integrity ✓ Security standards. ✓ Use Local Data Storage Working with Shared Preferences Working with SQLite Working with File Storage ● Implementation of microapps ✓ Description of modular microapp Definition Structure ✓ Applying modular microapps concept Project structure Dependency injection Shared components ✓ Perform microapp Build configuration Configure each module pubspec file Android gradle iOS build settings

###### ChatGPT said:

Here's a **detailed exam** for **Integrating Backend Functionality in Flutter**, covering **API integration, authentication, security, and microapps**.

## ****Exam: Integrating Backend Functionality in Flutter****

**Time:** 2 Hours  
**Total Marks:** 100

### ****Section A: Multiple Choice Questions (MCQs) [20 Marks]****

(Each question carries 2 marks)

What type of HTTP request is used to **retrieve data** from a backend?  
a) POST  
b) GET  
c) PUT  
d) DELETE

Which package is commonly used for making API calls in Flutter?  
a) sqflite  
b) provider  
c) http  
d) firebase\_messaging

What is the purpose of **JSON parsing** in Flutter?  
a) To store large files  
b) To format UI elements  
c) To convert API response into Dart objects  
d) To manage local databases

Which authentication method is considered **secure** for mobile apps?  
a) Storing passwords in SharedPreferences  
b) Using hardcoded API keys  
c) Implementing OAuth or JWT  
d) Sending credentials in plain text

What is the main function of **SharedPreferences** in Flutter?  
a) Managing user authentication  
b) Storing small amounts of local data persistently  
c) Making network requests  
d) Rendering UI components

### ****Section B: True or False [10 Marks]****

(Each question carries 2 marks)

1. HTTP PUT requests are used to **delete** a resource. (True/False)
2. Firebase can be used for **push notifications** in Flutter. (True/False)
3. **SQLite** is an in-memory database with no persistent storage. (True/False)
4. **Dart's HTTP package** is required to make API calls in Flutter. (True/False)
5. Modular microapps allow **code reuse and better project organization**. (True/False)

### ****Section C: Short Answer Questions [10 Marks]****

(Each question carries 5 marks)

1. Explain the difference between **GET, POST, and DELETE** HTTP requests with examples.
2. What are the security measures to protect **user data** in a Flutter app?

### ****Section D: API Integration (Coding) [20 Marks]****

(Each question carries 5 marks)

1. Write a **Flutter function** to fetch data from an API using http.get().
2. Implement **JSON parsing** in Flutter by converting an API response into a Dart object.
3. Write a **POST request** to send form data to an API.
4. Implement **authentication** using Firebase in Flutter.

### ****Section E: Storage Management [20 Marks]****

(Each question carries 5 marks)

1. How can **SharedPreferences** be used for storing user preferences?
2. Write a **SQLite database helper class** in Flutter.
3. What is **secure data storage**, and how can it be implemented in Flutter?
4. Explain the role of **File Storage** in mobile apps.

### ****Section F: Microapps and Modularization [20 Marks]****

(Each question carries 5 marks)

1. What are **microapps**, and why are they important in Flutter development?
2. Explain how **dependency injection** is used in modular microapps.
3. Write a **pubspec.yaml file configuration** for a microapp module.
4. What are the key steps in configuring **Android Gradle and iOS build settings** for microapps?

## ****Solutions and Explanations****

### ****Section A (MCQ Solutions & Explanations)****

1. **Answer: b) GET** → GET requests fetch data from a server.
2. **Answer: c) http** → The http package is commonly used for API calls in Flutter.
3. **Answer: c) Convert API response into Dart objects** → JSON parsing is needed to map API responses to Dart models.
4. **Answer: c) Implementing OAuth or JWT** → Secure authentication should avoid storing credentials directly.
5. **Answer: b) Storing small amounts of local data persistently** → SharedPreferences is used for saving key-value pairs locally.

### ****Section B (True/False Solutions & Explanations)****

1. **False** → PUT updates a resource, DELETE removes it.
2. **True** → Firebase Messaging supports push notifications.
3. **False** → SQLite is a persistent database stored on the device.
4. **True** → The http package is required for API calls in Flutter.
5. **True** → Microapps allow **better code reuse** and organization.

### ****Section C (Short Answer Solutions)****

1. **GET vs. POST vs. DELETE**

* GET: Retrieves data → http.get(Uri.parse(url))
* POST: Sends data → http.post(Uri.parse(url), body: jsonEncode(data))
* DELETE: Removes a resource → http.delete(Uri.parse(url))

1. **Security Measures in Flutter Apps**

* Use **OAuth/JWT authentication**
* Store sensitive data using **secure storage (flutter\_secure\_storage)**
* **Encrypt network communication** (HTTPS/TLS)
* Implement **input validation** to prevent SQL injection

### ****Section D: API Integration (Coding Solutions)****

1. **Fetch Data using GET Request**

dart

CopyEdit

Future<void> fetchData() async {

final response = await http.get(Uri.parse('https://api.example.com/data'));

if (response.statusCode == 200) {

print(response.body);

} else {

print('Failed to load data');

}

}

1. **JSON Parsing Example**

dart

CopyEdit

class User {

final String name;

final int age;

User({required this.name, required this.age});

factory User.fromJson(Map<String, dynamic> json) {

return User(name: json['name'], age: json['age']);

}

}

1. **POST Request Example**

dart

CopyEdit

Future<void> sendData() async {

final response = await http.post(

Uri.parse('https://api.example.com/create'),

headers: {"Content-Type": "application/json"},

body: jsonEncode({"name": "John", "age": 25}),

);

print(response.body);

}

1. **Firebase Authentication**

dart

CopyEdit

FirebaseAuth.instance.signInWithEmailAndPassword(

email: 'test@example.com',

password: 'password123',

);

### ****Section E: Storage Management Solutions****

1. **SharedPreferences Example**

dart

CopyEdit

SharedPreferences prefs = await SharedPreferences.getInstance();

prefs.setString('username', 'JohnDoe');

1. **SQLite Helper Class**

dart

CopyEdit

class DatabaseHelper {

static Database? \_database;

Future<Database> get database async {

if (\_database != null) return \_database!;

\_database = await openDatabase('app.db');

return \_database!;

}

}

1. **Secure Data Storage in Flutter**

* Use **flutter\_secure\_storage** for storing sensitive data.
* **Encrypt local files** before saving.

1. **File Storage in Flutter**

* Used for saving **images, documents, and offline data caching**.

### ****Section F: Microapps and Modularization Solutions****

1. **Microapps in Flutter**

* Independent **modular components** that improve scalability.

1. **Dependency Injection Example**

dart

CopyEdit

final service = Get.put(ApiService());

1. **Microapp** pubspec.yaml **Configuration**

yaml

CopyEdit

dependencies:

flutter:

sdk: flutter

shared\_module:

path: ../shared\_module

1. **Android Gradle & iOS Build Settings**

* Modify build.gradle for Android.
* Update Podfile for iOS dependencies.

## ****Final Thoughts****

This exam covers **API integration, backend authentication, security, storage management, and modular microapps**. Let me know if you need **practice problems** or **real-world projects**! 🚀

4o

##### You said:

● Perform Error Handling ✓ Description Definition of error handling Error classes ✓ Exception Management try-catch block OnError CatchError ✓ Rethrowing exceptions Finally block on clause ● Perform testing ✓ Description Definition Importance Testing Levels ✓ Implement Types of testing Unit Tests Widget Tests Integration Tests Functional Tests UI Tests Performance Tests Regression Tests Cross-Platform Testing Security testing End-to-End (E2E) Tests Mocking and Stubbing Code Coverage Tests ✓ Test device Responsiveness. Select testing tools Test using Emulator and Simulator Test using Physical Device Perform manual Testing Perform automated testing ✓ Test Reliability Unit and Widget Testing Integration and End-to-End Testing Edge Case and Stress Testing Performance Testing and User Acceptance Testing (UAT) ● Debug Codebase issues. ✓ Description of key terms Codebase Debug Logging Flutter DevTools Isolation Assertion Breakpoints Print statement ✓ Applying debugging methods ✓ Code Reviews ✓ Prepare documentation

###### ChatGPT said:

Here's a **detailed exam** on **Error Handling, Testing, and Debugging in Flutter**, covering **exception handling, testing strategies, and debugging techniques**.

## ****Exam: Error Handling, Testing, and Debugging in Flutter****

**Time:** 2 Hours  
**Total Marks:** 100

### ****Section A: Multiple Choice Questions (MCQs) [20 Marks]****

(Each question carries 2 marks)

What is the purpose of error handling in Flutter?  
a) To crash the application safely  
b) To prevent and manage runtime errors gracefully  
c) To slow down the application  
d) To remove logs automatically

Which Flutter class is used to **handle exceptions**?  
a) ErrorHandler  
b) Exception  
c) FlutterCatch  
d) ErrorManager

Which block in Dart executes **whether an error occurs or not**?  
a) catch  
b) finally  
c) on  
d) throw

What is **the main purpose of unit testing** in Flutter?  
a) Testing individual functions or methods in isolation  
b) Testing the entire app’s UI  
c) Checking server API responses  
d) Evaluating network performance

What tool is commonly used for **debugging Flutter applications**?  
a) Visual Studio Code only  
b) Android Logcat only  
c) Flutter DevTools  
d) Firebase Analytics

### ****Section B: True or False [10 Marks]****

(Each question carries 2 marks)

1. The catchError method is only used with synchronous code in Dart. (True/False)
2. Integration testing in Flutter ensures that **multiple modules work together correctly**. (True/False)
3. Debugging is **only useful for fixing syntax errors**. (True/False)
4. assert statements in Flutter are used **to check conditions during development**. (True/False)
5. Performance testing in Flutter is used to check **app speed and responsiveness**. (True/False)

### ****Section C: Short Answer Questions [10 Marks]****

(Each question carries 5 marks)

1. Explain **try-catch-finally** in Dart with an example.
2. What are **three key benefits of automated testing** in Flutter?

### ****Section D: Error Handling (Coding) [20 Marks]****

(Each question carries 5 marks)

1. Write a Flutter function that handles a **division by zero exception** using try-catch.
2. Demonstrate **rethrowing an exception** in Dart with a custom error message.
3. Implement a **catchError** method in a Future function.
4. Show an example of using on for handling specific exceptions in Dart.

### ****Section E: Testing in Flutter [20 Marks]****

(Each question carries 5 marks)

1. Write a **basic unit test** in Flutter for a function that adds two numbers.
2. Implement a **widget test** for a simple Flutter button.
3. Explain the difference between **unit tests, widget tests, and integration tests** in Flutter.
4. How does **mocking and stubbing** help in testing API calls in Flutter?

### ****Section F: Debugging and Code Reviews [20 Marks]****

(Each question carries 5 marks)

1. What is **Flutter DevTools**, and how can it help debug performance issues?
2. How do **breakpoints** help in debugging Flutter applications?
3. Write a Flutter function that **logs error messages** using debugPrint().
4. Explain the **importance of code reviews** in Flutter development.

## ****Solutions and Explanations****

### ****Section A (MCQ Solutions & Explanations)****

1. **Answer: b) To prevent and manage runtime errors gracefully** → Error handling ensures a smooth user experience.
2. **Answer: b) Exception** → Dart's Exception class is used for handling errors.
3. **Answer: b) finally** → The finally block executes no matter what happens.
4. **Answer: a) Testing individual functions or methods in isolation** → Unit tests focus on **single methods or functions**.
5. **Answer: c) Flutter DevTools** → DevTools helps analyze performance and debug UI issues.

### ****Section B (True/False Solutions & Explanations)****

1. **False** → catchError is used for **asynchronous** code, like Future methods.
2. **True** → Integration tests ensure different modules work **together correctly**.
3. **False** → Debugging helps fix **logical errors, UI issues, and crashes**.
4. **True** → assert() is used for **debugging and development** purposes.
5. **True** → Performance testing checks **speed, efficiency, and responsiveness**.

### ****Section C (Short Answer Solutions)****

1. **try-catch-finally Example**

dart

CopyEdit

void divideNumbers(int a, int b) {

try {

double result = a / b;

print(result);

} catch (e) {

print('Error: Cannot divide by zero');

} finally {

print('Division operation attempted.');

}

}

1. **Three Benefits of Automated Testing**

* **Early Bug Detection**: Finds issues before release.
* **Faster Development**: Reduces manual testing time.
* **Consistency**: Ensures stable and predictable app behavior.

### ****Section D: Error Handling (Coding Solutions)****

1. **Handling Division by Zero Exception**

dart

CopyEdit

void safeDivide(int a, int b) {

try {

double result = a / b;

print(result);

} catch (e) {

print('Error: Division by zero is not allowed.');

}

}

1. **Rethrowing Exception**

dart

CopyEdit

void processFile() {

try {

throw Exception("File not found");

} catch (e) {

print("Caught exception: $e");

rethrow; // Re-throws the same exception

}

}

1. **Using catchError with Future**

dart

CopyEdit

Future<void> fetchData() async {

await Future.delayed(Duration(seconds: 2), () {

throw Exception('Network error');

}).catchError((error) {

print('Error caught: $error');

});

}

1. **Using** on **for Specific Exception Handling**

dart

CopyEdit

void readFile() {

try {

throw FileSystemException("File is missing");

} on FileSystemException catch (e) {

print("File error: $e");

} catch (e) {

print("General error: $e");

}

}

### ****Section E: Testing in Flutter (Solutions)****

1. **Unit Test Example**

dart

CopyEdit

test('Addition test', () {

int add(int a, int b) => a + b;

expect(add(2, 3), 5);

});

1. **Widget Test Example**

dart

CopyEdit

testWidgets('Button test', (tester) async {

await tester.pumpWidget(MaterialApp(home: ElevatedButton(onPressed: () {}, child: Text('Click me'))));

expect(find.text('Click me'), findsOneWidget);

});

1. **Difference Between Tests**

* **Unit Test**: Tests individual functions.
* **Widget Test**: Tests a single UI component.
* **Integration Test**: Tests multiple components together.

1. **Mocking & Stubbing in API Calls**

* **Mocks** replace real API calls.
* **Stubs** provide pre-defined responses.
* Example: Using mockito in Flutter for fake API responses.

### ****Section F: Debugging and Code Reviews****

1. **Flutter DevTools** → Helps inspect UI, logs, memory usage, and performance.
2. **Breakpoints** → Stop execution at a point to inspect variables and flow.
3. **Logging Errors using debugPrint()**

dart

CopyEdit

void logError(String message) {

debugPrint('Error: $message');

}

1. **Importance of Code Reviews** → Improves code quality, reduces bugs, and ensures best practices.

This exam covers **error handling, testing strategies, and debugging techniques**. Let me know if you need **practice exercises**! 🚀

4o

##### You said:

Learning outcome 4: Publish Application Learning hours: 20 Indicative content ● Generation of installable files ✓ Description Types of builds Installable file (Android & iOS) ✓ Compilation Models Just-in-Time (JIT) Compilation Ahead-of-Time (AOT) Compilation Hot Reload and Hot Restart mechanisms ✓ Perform builds generation iOS (IPA) Android (APK) ● Submission of application files ✓ Prepare Store Assets App Icon App Screenshots App promotional materials/previews ✓ Create Developer account registration Google Developer Console Apple Developer Account ✓ Generate App Release Builds(.ipa, .aab) ✓ Configure app setting App ID(Android & iOS) App Description Set up app listing Distribution ✓ upload App bundles (Android & iOS) ● Address post deployment issues ✓ Monitor crash reports (UXCam,Sentry) ✓ Performance degradation ✓ Applying of App Store Optimization (ASO) ✓ Compatibility problems Based on Operating System type Based on Operating System version (API Level, iOS version) ✓ Perform Hot fixing

###### ChatGPT said:

Here’s a **detailed exam** on **Publishing a Flutter Application**, covering **build generation, app store submission, and post-deployment issues**.

## ****Exam: Publishing a Flutter Application****

**Time:** 2 Hours  
**Total Marks:** 100

### ****Section A: Multiple Choice Questions (MCQs) [20 Marks]****

(Each question carries 2 marks)

What is the primary difference between **JIT (Just-in-Time) and AOT (Ahead-of-Time) Compilation** in Flutter?  
a) JIT is faster for production builds, while AOT is used for debugging  
b) JIT is used for development, while AOT is used for release builds  
c) JIT creates APK files, while AOT creates IPA files  
d) JIT is slower than AOT in debugging mode

Which file format is used to distribute an **Android app on the Play Store**?  
a) .ipa  
b) .apk  
c) .exe  
d) .dmg

What is required before submitting an app to the **Google Play Store**?  
a) A Developer Console account  
b) A paid Apple Developer account  
c) iOS certificate signing  
d) None of the above

**Which tool can be used to monitor Flutter app crashes** after deployment?  
a) UXCam  
b) Firebase Crashlytics  
c) Sentry  
d) All of the above

What is **App Store Optimization (ASO)** used for?  
a) Improving an app’s visibility in the app store  
b) Making the app UI responsive  
c) Debugging app crashes  
d) Encrypting user data

### ****Section B: True or False [10 Marks]****

(Each question carries 2 marks)

1. An **.ipa** file is required to publish apps on the **Google Play Store**. (True/False)
2. **Hot Reload** helps developers quickly update and test changes without restarting the app. (True/False)
3. A **developer account** is required before an app can be published on **both the Google Play Store and Apple App Store**. (True/False)
4. **Performance degradation** can occur after app deployment due to inefficient memory usage. (True/False)
5. **Hotfixing** allows developers to fix critical issues without resubmitting the app to the store. (True/False)

### ****Section C: Short Answer Questions [10 Marks]****

(Each question carries 5 marks)

1. What is the **difference between an APK and an AAB file** in Android app distribution?
2. Explain **three key factors that improve an app's ranking using App Store Optimization (ASO)**.

### ****Section D: Build and Deployment (Coding & Configuration) [20 Marks]****

(Each question carries 5 marks)

1. Write the command to **generate an APK** in Flutter.
2. Write the command to **generate an iOS build (.ipa file)** for App Store submission.
3. What is the process to configure an **App ID and Package Name** for Android and iOS?
4. How do you set up an **app icon** in a Flutter project?

### ****Section E: Submission of Application Files [20 Marks]****

(Each question carries 5 marks)

1. What are the **required assets for publishing an app** on Google Play and the Apple App Store?
2. Describe the **steps to upload an app bundle (.aab file) on the Google Play Store**.
3. How can developers manage **app versions** when releasing updates?
4. What are the **distribution options** for releasing an iOS app?

### ****Section F: Post-Deployment Issues and Monitoring [20 Marks]****

(Each question carries 5 marks)

1. How do developers **track app crashes** after deployment? Name at least two tools.
2. How does **Firebase Performance Monitoring** help in post-deployment issue detection?
3. What are the **common compatibility problems** developers face after app deployment?
4. How can developers perform **hotfixing** for a deployed Flutter app?

## ****Solutions and Explanations****

### ****Section A (MCQ Solutions & Explanations)****

1. **Answer: b) JIT is used for development, while AOT is used for release builds** → JIT is for debugging, and AOT compiles optimized machine code for production.
2. **Answer: b) .apk** → Android apps are distributed as .apk or .aab files.
3. **Answer: a) A Developer Console account** → Developers must create a **Google Developer Console** account to publish apps.
4. **Answer: d) All of the above** → UXCam, Firebase Crashlytics, and Sentry are tools used to monitor app crashes.
5. **Answer: a) Improving an app’s visibility in the app store** → ASO helps in increasing downloads and ranking.

### ****Section B (True/False Solutions & Explanations)****

1. **False** → .ipa is for iOS, and .apk is for Android.
2. **True** → Hot Reload applies changes instantly without restarting the app.
3. **True** → Both Google Play and Apple App Store require a developer account.
4. **True** → Poor coding or high memory usage can cause performance issues.
5. **True** → Some **critical hotfixes** can be pushed without a full resubmission.

### ****Section C (Short Answer Solutions)****

1. **APK vs AAB**

* **APK**: A directly installable file for Android devices.
* **AAB (Android App Bundle)**: A format optimized for **Google Play** that generates APKs dynamically for specific device configurations.

1. **Three ASO Factors**

* **Keyword Optimization**: Using relevant keywords in the app title and description.
* **High-Quality Screenshots & Videos**: Engaging visuals increase downloads.
* **Positive Reviews & Ratings**: Higher ratings improve rankings.

### ****Section D: Build and Deployment (Coding Solutions)****

1. **Generate APK**

sh

CopyEdit

flutter build apk --release

1. **Generate IPA (iOS)**

sh

CopyEdit

flutter build ios --release

1. **Configuring App ID and Package Name**

* **Android**: Edit android/app/build.gradle and AndroidManifest.xml.
* **iOS**: Modify ios/Runner.xcodeproj/project.pbxproj.

1. **Set Up App Icon**

* Use flutter\_launcher\_icons package and run:

sh

CopyEdit

flutter pub run flutter\_launcher\_icons:main

### ****Section E: App Submission (Solutions)****

1. **Required Store Assets**

* App Icon
* Screenshots (various sizes)
* App description & promotional video

1. **Upload App Bundle (.aab) to Google Play Store**

* Step 1: Open **Google Play Console**
* Step 2: Create a new release
* Step 3: Upload the .aab file
* Step 4: Fill in store listing and submit

1. **Managing App Versions**

* Update versionCode and versionName in pubspec.yaml.
* Example:

yaml

CopyEdit

version: 1.1.0+2

1. **iOS App Distribution Options**

* **App Store**
* **TestFlight for beta testing**
* **Enterprise distribution**

### ****Section F: Post-Deployment Monitoring (Solutions)****

1. **Crash Monitoring Tools**

* **Firebase Crashlytics**
* **Sentry**

1. **Firebase Performance Monitoring**

* Tracks **slow UI interactions**
* Monitors **network requests**

1. **Common Compatibility Problems**

* **OS version updates** breaking UI
* **API changes** in dependencies

1. **Hotfixing Methods**

* **Use Firebase Remote Config**
* **Patch using OTA (Over-the-Air) updates**

This exam covers **Flutter app publishing, deployment, and post-release management**. Let me know if you need **practice exercises**! 🚀

4o

窗体顶端

O

Search

Reason

窗体底端

ChatGPT can make mistakes. Check important info.