I. Mobile App Development: Pros & Cons

• **Pros**:

• Multiplatform support

• Quick to develop

• No internet required

• Saves time

•  **Cons:**

• Poor user experience

• Time-consuming

• Supporting issues

II. Mobile App Development Process

• Strategy

• Analysis & Planning

• UI/UX Design

• App Development

• Testing

• Deployment & Support

III. Mobile App Development Platforms

• iOS (Apple) (primary tool used for iOS dev are Xcode & Swift)

• Android (Google) (Android Studio is used)

• Windows Phone, Blackberry, Tizen

IV. Mobile App Development Framework

• Refers to a software library including software tools, compilers, APIs providing basic structure & support @ the time of mobile app development for a specific platform.

V. Cross-Platform Frameworks

• Tools that allow developers to create mobile apps that can run on multiple computing frameworks (platforms): Android, iOS, Windows, etc.

• Separate coding. Save time & cost.

• Examples:

• Flutter

• React Native

• Ionic

• Xamarin

• Unity

• NativeScript Framework

VI. Flutter (Cross-Platform)

• Open-source mobile app dev platform by Google

• Allows use to build apps using a single codebase

• Offers range of features & tools for building responsive apps.

• Features: hot reload, single codebase, customizable widgets

• Programming Language: Dart

• Apps: eBay, Alibaba, Google Pay

VII. React Native (Cross-Platform)

• Open Source UI-Software framework developed in 2015 by Meta platforms (formerly Facebook)

• Features: hot reload, flexibility, "like native apps"

• Programming Language: Javascript

• Mobile Apps: Skype, Shopify

VIII. Benefits of Cross-Platform App Framework

• Code reusability

• Fast development

• Cost efficiency

• Simplified maintenance

• Wider market reach

• Access to Native Features

IX. Dart

• Open-source, general-purpose, OOP language with C-style syntax developed by Google in 2011.

• Purpose is to help develop apps for many platforms.

X. Dart Features

• Open Source

• Easy to learn

• Type safe

• Browser support

• Object-oriented

• Platform independence

XI. Dart Environment Setup

• Executing Dart script online using Dartpad

• Setting Up on the Local Env

XII. **Dart Syntax and Key Concepts**

• **Syntax**: Rules that define the structure of a language. Includes structure, symbols, punctuation, and words of a programming language.

**• main()**: Predefined method. Acts as an entry point to the application. Dart needs it for execution.

**• Print()**: Predefined function that prints a specified string.

**• Keywords**: Reserved words that give special meaning to the Dart compiler (e.g., import, else, false, true, export, continue, break, switch, return).

**• OOP (Object-Oriented Programming)**: Software paradigm that reflects real-world models. Communicates with each other via a mechanism called message passing.

**• Object**: Real-time representation of any entity. (State, behavior, identity)

**• Class:** A blueprint for creating objects.

**• Method:** Facilitates communication between objects.

• **Variables**: Named space in memory. Variable names are called identifiers.

**• String**: A Dart string is a sequence of UTF-16 code units.

• **Operators in Dart**: Arithmetic, increment/decrement, bitwise, assignment, logical

:IF ELSE IF

void main() {

var num = 2;

if (num > 0) {

print("${num} is positive");

} else if (num < 0) {

print("${num} is negative");

} else {

print("${num} is neither");

}

}

**SWITCH**

void main() {

var grade = "A";

Switch (grade) {

Case "A": { print ("Excellent"); }

break;

Case "B": { print ("Pass"); }

break;

Case "C": { print ("fail"); }

break;

default: { print ("Invalid"); }

break;

}

}

**Fxn**

void main() {

test();

}

test() {

// fcn definition

print("fxn called");

}

Example:

void main() {

print('Enter ur name');

String? name = Stdin.readlineSync();

print('Hello, $name');

}