Introduction

Advanced Mobile Programming

ITSE-3123

Course Objective

Understand principles and best practices of mobile application development using the Flutter framework

Learning outcomes

After Completing this course you should be able to

- Explain the **fundamentals** of the **Flutter framework**
- Incorporate **widgets** and **state** into your app
- Use **Flutter's tools** to enhance your development process
- Customize your app with **Material Design**, **themes**, **assets**, and more
- Make your **app interactive** with **text input**, **gestures**, and more

Learning outcomes

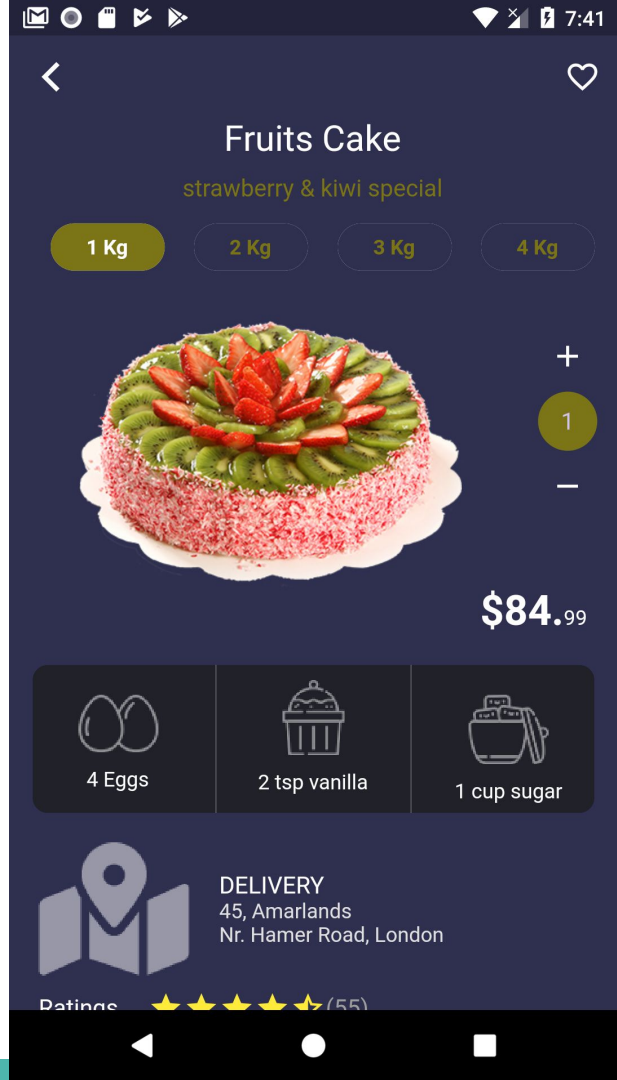
After Completing this course you should be able to

- Retrieve **local** and **real-time** data from the web
- Use **Location** and **Map** services
- **Test** mobile application
- Develop **multimedia applications** in Android

What is Flutter

Flutter is **Google's UI toolkit** for building **beautiful**, **natively compiled** applications for **mobile**, web, and desktop from a **single codebase**

`https://flutter.dev/`



Benefits of using Flutter

Fast Development

Flutter's **hot reload** helps you quickly and easily experiment, build UIs, add features, and fix bugs faster

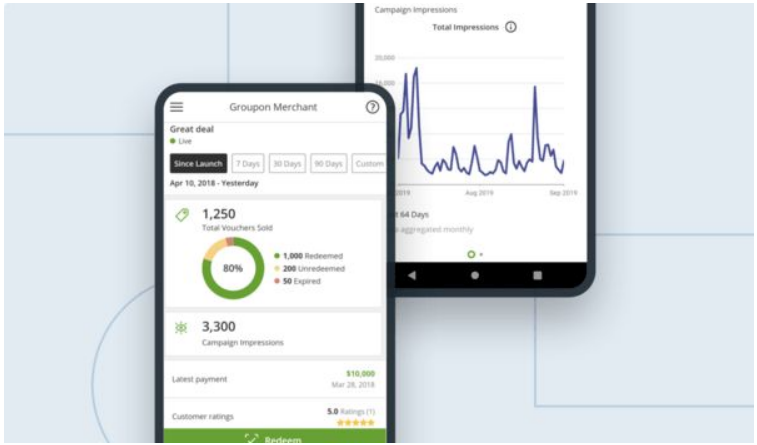
Experience **sub-second reload times**, without losing state, on emulators, simulators, and hardware for iOS and Android.

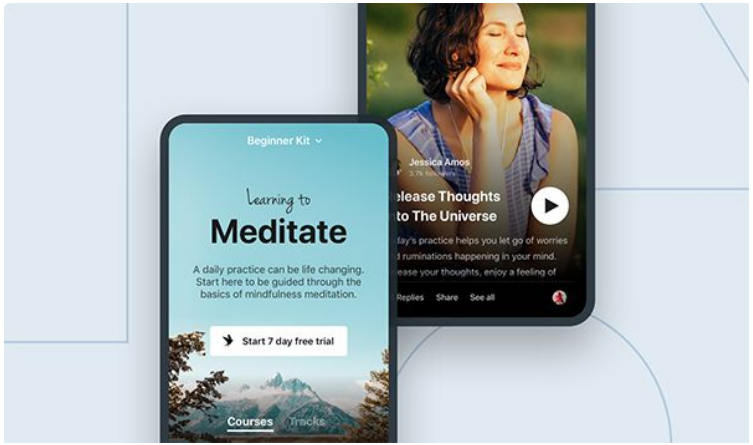
Rich set of fully-customizable widgets

Benefits of using Flutter

Expressive and Flexible UI

Built-in beautiful **Material Design** and **Cupertino (iOS-flavor)** widgets, **rich motion APIs**, **smooth natural scrolling**, and **platform awareness**





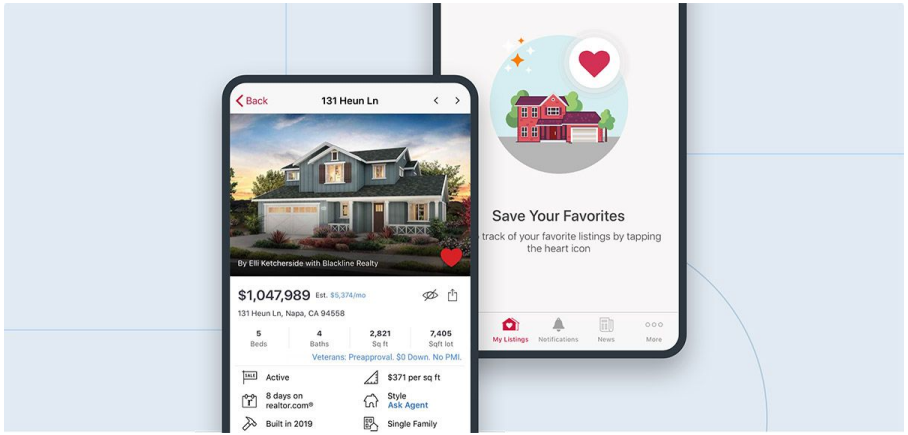
https://flutter.dev/showcase

Benefits of using Flutter

Native Performance

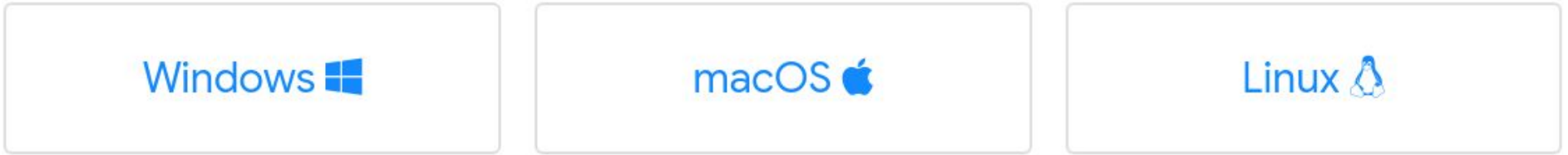
Your Flutter code is **compiled to native ARM**

Thus Flutter gives you **full native performance on both iOS and Android**



How to set up development platform

Go to `https://flutter.dev/docs/get-started/install` and follow the instruction for your platform



Set up an editor

Option One (`https://flutter.dev/docs/get-started/editor?tab=androidstudio`)

Install **Android Studio**

Install the **Flutter** and **Dart plugins**

Option Two (`https://flutter.dev/docs/get-started/editor?tab=vscode`)

Install **VS Code**

Install the Flutter and Dart plugins

Validate your setup with the Flutter Doctor

What is Dart

Dart is a **client-optimized language** for fast apps on any platform

Flutter is built using Dart language

`https://dart.dev/`

Benefits of using Dart

Optimized for UI

A programming language specialized around the needs of user interface creation

Productive development

Make changes iteratively: use hot reload to see the result instantly in your running app

`https://dart.dev/`

Benefits of using Dart

Fast on all platforms

Compile to ARM & x64 machine code for mobile, desktop, and backend

Or compile to JavaScript for the web

https://dart.dev/

Brief introduction to Dart

Dart language features

Variables, Control flow statements, Functions, Comments, Async, Imports, Classes, Inheritance, Mixins, Interfaces and Abstract Classes, Exceptions

https://dart.dev/samples

Hello World

Every app has a `main()` function

To display text on the console, you can use the top-level `print()` function

```
void main() {
 print('Hello world');
}
```

Variables

Even in **type-safe** Dart code, **most variables don't need explicit types**, thanks to **type inference**

```
1 var name = 'Voyager I';
2 var year = 1977;
3 var antennaDiameter = 3.7;
4 var flybyObjects = ['Jupiter', 'Saturn', 'Uranus'];
5 var image = {
    'tags': ['saturn'],
    'url': '//path/to/saturn.jpg'
  };
```

From what you already know in other programming languages, guess the data type of the variable

Control flow statements

If else

```
var year = 2019;
if (year >= 2001) {
  print('21st century');
} else if (year >= 1901) {
  print('20th century');
}
```

Control flow statements

For loop

```
var shapes = ['Circle', 'Rectangle', 'Triangle'];

for(var shape in shapes){
  print(shape);
}
 for (int i = 0; i < shapes.length; i++) {
    print(shapes[i]);
 }
```

Control flow statements

While loop

```
var shapes = ['Circle', 'Rectangle', 'Triangle'];
int count = 0;
while (count < shapes.length) {
  print(shapes[count]);
  count++;
}
```

Functions

It is best practice to specify the types of each function's arguments and return value

```
void main() {
 print(add(2,3));
}

add(x, y) {
 return x+y;
}
```

```
Recommended way of defining functions

        int add(int x, int y) {
         return x+y;
        }
```

Functions

A shorthand **=> (arrow)** syntax is handy **for functions that contain a single statement**

This syntax is especially useful when passing **anonymous functions** as arguments

```
void main() {
 print(add(2,3));
}
int add(int x, int y) => x+y;
```

Comments

```
// This is a normal, one-line comment.


/// This is a documentation comment, used to document
/// libraries, classes, and their members. Tools like IDEs
/// and dartdoc treat doc comments specially.


/* Multi-line comments like this is also

supported */
```

Imports

```dart
// Importing core libraries
import 'dart:math';


// Importing libraries from external packages
import 'package:test/test.dart';


// Importing files
import 'path/to/my_other_file.dart';
```

Class

```dart
import 'dart:math';
class Circle {
 double _radius;
 static const double PI = 3.14;
 Circle(this._radius);
 Circle.inMeter(this._radius);
 double area() => PI * pow(_radius, 2);
 get radius => _radius;
}
```

Class with two properties, two constructors, one method and one getter method

Class

Class with two properties, two constructors, one method and one getter method

```dart
import 'dart:math';
class Circle {
 double _radius;
 static const double PI = 3.14;
 Circle(this._radius);
 Circle.inMeter(this._radius);
 double area() => PI * pow(_radius, 2);
 get radius => _radius;
}
```

Constructors

Getter

Inheritance

Dart has single inheritance

```
abstract class Shape {
 double area();
}
```

```
class Circle extends Shape {
 double _radius;
 static const double PI = 3.14;

 Circle(this._radius);

 @override
 double area() => PI * pow(_radius, 2);
 get radius => _radius;
}
```

Mixins

Mixins are a way of reusing code in multiple class hierarchies

```
mixin Status {
 bool fullTime;
 bool manager;
}
```

```
class Employee with Status{
 String name;
 String address;
 String salary;
}
```

Interfaces and abstract classes

Dart has no interface keyword

All classes implicitly define an interface.

Therefore, you can implement any class

```dart
abstract class Shape {
 double area();
}
```

```dart
class Circle implements Shape {
 double _r;
 static const double PI = 3.14;
 Circle(this._r);
 double area() => PI * _r * _r;
}
```

Exceptions

To raise an exception, use throw

```
void main(){
 int x, y;

 if(x == null || y == null) {
   throw StateError("x or y are null");
 }
}
```

Exceptions

To catch an exception, use a try statement with `on` or `catch` (or both)

```
void main() {
 int x = 1, y = 0;
 try {
   x / y;
 } catch (IntegerDivisionByZeroException) {
   print('Y cannot be zero');
 }
}
```

Assessments

Laboratory Assignments (total mark 15 - 20 points)

Final Exam (60 points)

Project (20 to 25 points)

Missing any of these assessments results in incomplete (NG) grade, if you can not bring any evidence for your abscency, eventually, the NG grade turns to F

General Project Requirements

You should form a group with a **maximum of 5 members**

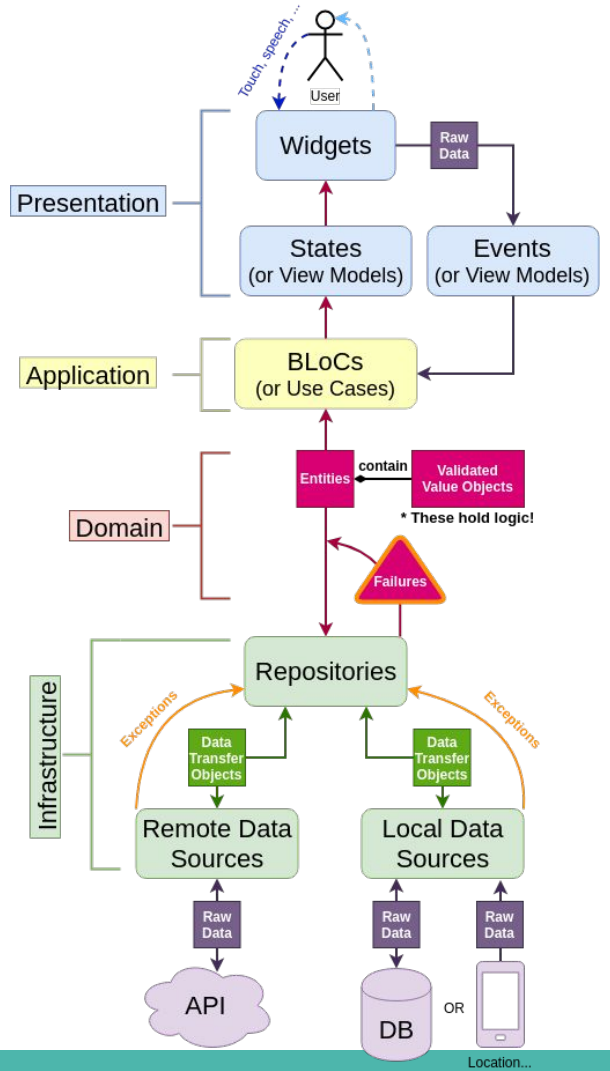
Select a title with the assumption of the following minimal requirements

- Two business features (in addition to authentication/authorization)
- You should have a backend (REST API) that provide the two functionality
- Widget, Unit, Integration Testing (Bonus)

You should use **Github** as your git repository

Track your project progress using Kanban board on Github

Application Architecture



Project Evaluation

Final evaluation for **marking/grading** your work

Resources

Dart

A tour of the Dart language (https://dart.dev/guides/language/language-tour)

Dart cheatsheet (https://dart.dev/codelabs/dart-cheatsheet)

Language Samples (https://dart.dev/samples)

Resources

Flutter

Flutter in Action, By Eric Windmill, 2020 (Text Book)

Beginning App Development with Flutter: Create Cross-Platform Mobile Apps, By Rap Payne, 2019

Flutter Succinctly By Ed Freitas, 2019

Flutter Docs (https://flutter.dev/docs)

Build Native Mobile Apps with Flutter (https://www.udacity.com/)