# Department of Computing

**CS-415: Mobile Application Development**

**Class: BSCS 9BC**

**Lab 09: Asynchronous programming: futures, async, await**

**CLO 3: Analyze implementation strategies provided by various modern frameworks for creating mobile applications**

**Submitted by:**

**Fatima Zahid**

**CMS ID: 294782**

**Date: 04-April-2023 Time: 09:45 AM - 11:55 AM & 12:00 PM – 02:25 PM**

## Instructor: Dr. Hasan Ali Khattak

**Lab Engineer: Ms. Maham Daoud**

**Lab 09: Asynchronous programming: futures, async, await**

**Objectives**

This lab will get students familiar with Asynchronous programming: futures, async, await.

# Topics to be Covered

* Asynchronous programming
* Using a Future
* Using async/await to remove callbacks
* Completing Futures

# Tools/Software Requirement

* Flutter SDK
* Android Studio
* VS Code
* [FlutLab](https://flutlab.io/)

# Lab Tasks : Future :

A Future addresses a computation that doesn’t finish right away. Though a typical function

returns the outcome, an asynchronous function returns a Future, which will ultimately contain the outcome. The Future will reveal to you when the outcome is prepared.

So, a Future can be in one of three states:

* **Uncompleted:** You called a Future, but the response isn’t available yet.
* **Completed with value:** The output is open, and data is ready (The *then () function* is called*).*
* **Completed with an error:** The output is open, but something went wrong (The *catch Error ()* function is called

# Task # 1

Follow the guidelines mentioned in <https://dart.dev/codelabs/async-await> and try solving the example problems in Flutter. (ETA 30 – 60 minutes) Note: This task is supposed to be completed in the Lab.

# Outputs:

## Part a:

Graphical user interface, text, application, email

Description automatically generated

## Part b:

## Graphical user interface, text, application Description automatically generated

## Part c:

## Graphical user interface, text, application, email Description automatically generated

## Part d:

## Graphical user interface, text, application, email Description automatically generated

## Part e:

## Graphical user interface, text, application, email Description automatically generated

# Task # 2

This task is based on asynchronous programming. Before Implementing you need to go through these two helping link for better understating of asynchronous programming.

* [https://medium.flutterdevs.com/exploring-asynchronous-programming-in-dart-](https://medium.flutterdevs.com/exploring-asynchronous-programming-in-dart-flutter-25f341af32f) [flutter-25f341af32f](https://medium.flutterdevs.com/exploring-asynchronous-programming-in-dart-flutter-25f341af32f)
* [https://medium.flutterdevs.com/explore-futures-in-flutter-](https://medium.flutterdevs.com/explore-futures-in-flutter-50ea5b91fc2#%3A~%3Atext%3DTo%20perform%20such%20tasks%20in%2CLike%20the%20UI%20thread) [50ea5b91fc2#:~:text=To%20perform%20such%20tasks%20in,Like%20the%20UI](https://medium.flutterdevs.com/explore-futures-in-flutter-50ea5b91fc2#%3A~%3Atext%3DTo%20perform%20such%20tasks%20in%2CLike%20the%20UI%20thread)

[%20thread](https://medium.flutterdevs.com/explore-futures-in-flutter-50ea5b91fc2#%3A~%3Atext%3DTo%20perform%20such%20tasks%20in%2CLike%20the%20UI%20thread).

1. Make a network request using the **http** package and call the below API. <https://jsonplaceholder.typicode.com/todos>
   * Install the Postman or add Rest let Client chrome extension to test the above.
   * Check the response of above API by using postman or Rest let Client.
2. Write your model class to store the contents of the JSON file.

1. Write code to receive and parse response from the APIs in a JSON format.
2. You should also write code for initializing your model classes from JSON object(s).
3. Create a List view widget to display the data that you get in API response (as shown below).
   * Show **id** and **title** from json response. (screen shot 1)

# My output:

Graphical user interface, text, application, chat or text message

Description automatically generated

# Useful Resources

Below is the list of useful resources to help you in solving this task:

1. Asynchronous Programming : <https://dart.dev/codelabs/async-await>
2. Asynchronous Programming : [https://medium.flutterdevs.com/exploring-asynchronous-](https://medium.flutterdevs.com/exploring-asynchronous-programming-in-dart-flutter-25f341af32f) [programming-in-dart-flutter-25f341af32f](https://medium.flutterdevs.com/exploring-asynchronous-programming-in-dart-flutter-25f341af32f)
3. JSON Parsing gist: <https://gist.github.com/bizz84/798c1c5a69690af54d09ca6e8469faec>
4. Package Loading: <https://dart.dev/guides/packages>
5. **http** Library: <https://pub.dev/packages/http>
6. JSON Parsing Tutorial: <https://codewithandrea.com/articles/parse-json-dart/>
7. Circular Progress Indicator :

<https://api.flutter.dev/flutter/material/CircularProgressIndicator-class.html>

## Deliverables and other information:

* + All tasks are ideally supposed to be finished in the lab.
  + Each folder should contain fully working and compliable code.
  + You have to submit the code in this link: <https://classroom.github.com/a/-q7BpSLq>
  + This lab is graded between 0 to 10 marks.
  + Marks will be deducted in case if any functionality is incomplete, missing, or code is not compiling.
  + Late submissions will also result in deducting 2 marks.
  + You will be awarded a zero if you fail to submit the task.
  + At the end of each lab or in the next lab, there will be a viva/quiz related to the

tasks.