

Names: Nsenga Ivan MANZI Reg N<sup>o</sup> 223004392

Claudine IRADUKUNDA Reg N<sup>o</sup> 223007635

Year 3

Computer engineering.

Date: 4<sup>th</sup> Feb 2026

Mobile application systems and design.

## Report summary on dart programming lab.

### Introduction.

In this lab, we worked together to understand and practice the basic concepts of the Dart programming language. Through the given questions, we explored important programming topics such as functions, named and optional parameters and many others.

For each question, we wrote Dart code to apply the concept and observed the output to understand how it works in practice. We also reflected on what each part of the code does and what we learned from it. This report presents a short explanation for each task, describing the purpose of the code, the main concept used, and the knowledge gained from completing the exercise.

### Summary of each question.

Q1) On this question, the code contains a function called `welcomeMessage` that prints a welcome message. This helped us to understand that a function is a reusable block of code designed to perform a specific task and it helps to organize the program.

Q2) On this question, the code contains a function that creates a student using named parameters such as `name` and `age`. This showed us that named parameters are parameters that we pass to a function using their names, not just their position, and they are helpful because they make the code easier to read, reduce mistakes and clearly show what each value represents.

Q3) On this question, the code contains a function that accepts a required name and an optional parameter. If the subject is not provided, a default message is printed. This helped us to learn that optional parameters are parameters that allow a function to work even if some values are not provided, and they work in a way that if a subject is not given, the function

prints a default message.

- Q4) On this question, the code contains a Student class with name and age variables and a constructor to initialize them. This helped us to learn that a constructor is a special method used to initialize values when an object is created from a class, and they are important because they ensure that objects start with the correct data and help to organize how objects are created.
- Q5) On this question, the code contains an object from the student class and print its values. This taught us object creation means making an instance from a class, and the object is used by accessing its values and printing them to show the stored information.
- Q6) On this question, the code contains a Person class with a name variable and an introduce(). This taught us a class is a blueprint used to create objects, and it defines the properties that object will have.
- Q7) On this question, the code contains Student class which inherits from Person class and uses the introduce() method. This taught us that inheritance is a way a class uses the properties and methods of another class, and a Student class inherits from Person class and reuses the introduce() function.
- Q8) On this question, the code contains abstract class called Registrable interface with a method that must be implemented. This taught us an interface is a structure that defines methods a class must implement.
- Q9) The code contains Student class that implement Registrable interface and define the required method. This taught us implementing interface means a class must define all the methods listed in the interface. This enforces rules by making sure the class provides the required functionality.
- Q10) The code contains a mixin called AttendanceMixin to track attendance. This taught us that a mixin is a way to add extra features or behavior to a class without using inheritance.
- Q11) The code contains the mixin to the student class and increased attendance 3 times. This taught us that mixins add behavior by providing extra methods and properties to a class.
- Q12) The code contains a list that stores multiple Student objects. This taught us that a list is a collection used to store multiple values in order, and it is useful for keeping several Student objects together and managing them easily.
- Q13) The code contains a map where the key is a student ID and the value is a student object. This taught us that a map stores data using key value pairs. It is useful when each value needs a unique identifier.

- Q14) On this question, we used an anonymous function to print student names from a list. This helped us learn that an anonymous function is a function without a name, and it is used for short tasks like printing each student's name from a list.
- Q15) On this question, the code allows an arrow function to print a greeting message. This showed us that arrow functions are a shorter way to write simple functions that have only one expression. In short they make the code cleaner, easier to write and easier to read.
- Q16) On this question, the code allows an async function to wait 2 sec before returning a list of students. This helped us to understand that an async function is a function that allows a program to perform tasks that take time such as loading data, and then waiting means the program pauses at a certain point until the task is finished.
- Q17) On this question, the code uses the await keyword to load students and print how many were returned. This helped us to understand that async programming helps in real apps by allowing tasks like loading data from the internet to happen without freezing the app.
- Q18) Mixins are useful because they allow extra features to be added to a class without creating a parent-child relationship. Inheritance shares structure and behavior between classes, while mixins only add specific functionality. Inheritance is useful for hierarchy, while mixins are used for adding reusable features.
- Q19) On this question, the code allows a mixin that prints a message when a student registers for a course. The new mixin was used to add a notification feature to the student class. When student registers for a course, the mixin prints a message.
- Q20) Learning Dart has helped us understand Flutter as Flutter normally uses Dart to build mobile applications, so by learning the concepts in Dart such as classes, functions and async programming, we directly learned the concepts used when creating Flutter apps.

Conclusion.

As the conclusion, this lab helped us to build a strong foundation in the Dart programming language through practical exercises. By completing each question, we gained hands on experience with important concepts such as functions parameters and others. This prepared us for more advanced development especially in learning Flutter, since Flutter relies on Dart for building mobile apps.