National College of Ireland

HDip inComputing

(**HDSDEV_JAN22, HDCSDEV_INTJAN22**)

Distributed Systems

Project Description

WEIGHT: 60% of overall marks

This **project** is to be done **individually**.

Schedule

Deliverables	Due Date
Project proposal	5 July 2022
Project submission (Including report, code, presentation)	10 August 2022

Aims

The purpose of this project is to gain experience in designing and building distributed systems.
You will be expected to demonstrate that your solution meets the objectives set out in this document. This **project** is to be done **individually**.

Project Description

For the purpose of this assignment, you assume that you lead a Software Development Consultancy company which has been commissioned to create an application on the specific industry based on your **last digit of your student Id**. In particular, from the following list and based on the last digit of your student id please select the corresponding industry.

For example: If your student ID is x22068381 then your industry would be “Smart water/air pollution tracking” (last digit is “one”).

Last student digit	Domain (Industry)
0	Smart farming
1	Smart water/air pollution tracking
2	Smart warehouse (track stock, automate orders)
3	Smart medical environment
4	Smart/automated business processes
5	Wearables
6	Smart transport system/ Traffic management
7	Smart Surveillance/Security

8	Smart building
9	Emergency services

As part of your task, you have to develop a set of protocols/messages and build a reference implementation that **simulates** the operations of a **smart automated environment** (for example transport, building, farming). As a result, your environment could consist of smart services and devices that inter- communicate with each other.

Your devices/services must **publish** themselves and **discover** each other. Your devices/services should communicate via **gRPC**. Java programming language must be used to develop your solution.

You should begin by devising your own scenario. There must be a minimum of 3 separate services that would simulate the operations of smart-automated environment. It is key to specify what operations are supported on each "service/device" in a corresponding proto file.

Finally, to demonstrate your implementation a simple client GUI should be developed, operating as a main controller that discovers and uses your **devices/services.**

Deliverables

Proposal:

A short project proposal must be first defined detailing the description of your application domain and 3 **services**. This should be approx. of 500 words. The proposal should include

- Title page (student Id, Project name) should follow NCI standard template
- Domain description
- Service definition and rpc (for all the services)
- Overall, you should include all types of RPC

Report

A report which details the scenario and services you have chosen. Additionally, this should specify the message formats for data exchange and service actuation. The report must have all the headings of the marking scheme, such as

- Service Definitions
- Service Implementations
- Naming Services.
- Remote Error Handling & Advanced Features
- Client - Graphical User Interface (GUI)
- GitHub

Program Code

A project with all code, well commented. Code must also be available in a private **GitHub** repository, the repo must have a commit history, not a last-minute code dump.

Video Presentation

There will be a presentation, which will act as a demonstrating the different parts of your application. The presentations should be recorded and not exceed a total of 10 minutes in duration.

Marking Scheme

- **Proposal (5%)**
- **Report (5%)**
- **Presentation/Viva (8%) – Present and Explain your application**
- **Service Definitions - Use of gRPC (protos) (14%)**
  - For each of the 3 different services/devices a corresponding proto file is defined and used [3 services * 2= 6%]
  - All 4 different types of RPC invocation styles have been used (simple RPC, server- side streaming RPC, client-side streaming RPC, bidirectional streaming RPC) [4 styles * 2 = 8%]
- **Service Implementations. Implement 3 <u>sufficiently complex service</u> implementations (30%)**
  - The three services should be written in Java. [3 services * 7 = 21%]
  - Additional service complexity and implementation [3 services * 3 = 9%]
- **Naming Services. Use of jmDNS (12%)**
  - Marks for registration [3 services * 2 = 6%]
  - Marks for discovery [3 services * 2 = 6%]
- **Remote Error Handling & Advanced gRPC (10%)**
  - Appropriate error handling for remote invocations, user input validation, and error messaging. Cancelling of messages [5%]
  - Use of Metadata, Deadlines, Authentication etc. [5%]
- **Client - Graphical User Interface (GUI) (6%)**
  - That allows to view (e.g., present, discover), control (parameters) and invoke the services/devices. That is the client for each of the 3 services [3 services * 2 = 6%]
- **GitHub (10%)**
  - Maintain a repo with a regular commit history