National College of Ireland

HDip in Computing

(HDSDEV_JAN24_Y1_O,HDCSDEV_INT,HDSDEV_SEP24_HDBC_SEP24OL)

Distributed Systems

WEIGHT: 60% of overall marks This project is to be done individually.

CA Deliverables	Submission Date
Project Proposal	Friday 14th March 23:59 PM
Project submission (including report, code, presentation)	Fri, 18 th April 23.59 P.M

Aim

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

You lead a software development consultancy that has been hired to develop an intelligent automation system for a specific industry, based on the last digit of your student ID. The system consists of smart services and devices that interact with each other via gRPC-based communication. The domain of your application is assigned to you based on the last digit of your Student ID. The devices and services you implement must be based on real world objects in the assigned domain. For example if your application domain was a Smart Zoo you might identify devices such as motion sensors, automated feeders, electronic gates or virtual fences.

Last student digit of student id	Domain (Industry)
0	Smart Offices (automated lighting, meeting room booking, air quality)
1	Smart Home (connected devices, smart security, automated climate control)
2	Customer Support (AI chatbots, automated ticketing, sentiment analysis)
3	Smart Transportation (AI-based traffic lights, smart parking)
4	Retail (Personalized shopping recommendations, automated checkout, inventory tracking)
5	Smart Warehouses (automated stock tracking, robotic inventory)
6	Smart Healthcare (remote monitoring, AI-driven diagnostics)
7	Education (Smart learning management, AI tutors, automated grading)
8	Recruitment (Automated hiring, interview scheduling, AI-driven candidate filtering)
9	Smart Cities (traffic automation, AI-based public services)

For example: If your student ID is x17068381 then your industry would be “Smart home system” (last digit is “one”).

Your implementation should include:

- ✅ At least 3 services that simulate devices or operations.
- ✅ Service Discovery: Devices/services should be discoverable automatically.

✅ gRPC Communication: Services communicate using defined .proto files.
✅ Security Features: Basic authentication (e.g., API keys, JWT) for access control.
✅ Data Logging: Capture logs for analytics (e.g., service usage, error tracking).
✅ GUI Controller: A graphical interface for monitoring and controlling 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 hospital, building, office). 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 must communicate via gRPC. You can use Java or Node.js/JavaScript to develop your solutions.

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 the main controller that discovers and uses your devices/services.

Deliverables

Project Proposal (Due on March 14th)
A brief project proposal detailing the description of your application domain and 3 services. This should be 1000 words or less. Describe the application that you will build. Identify your assigned application domain, the overall purpose of the service, the functionalities within each service and overall contribution of the services to the application. Provide the content of the proto file(s) that you will use to define the services with a brief explanation of the content. You can revise the proto files(s) in your final submission if you need to.

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

- Title page (student Id, Project name) should follow NCI standard template
- Domain description: you should best describe the overall purpose of the service, explain the functionalities within each service and overall contribution of the service to the application
- Service definition and RPC (for all the services): you should explain in detail, for example the request and response for each functionality within the service. Explain in detail the parameters.
- Overall, you should include all types of RPC
- Service Definitions
- Service Implementations
- Use of Naming Services.
- Remote Error Handling & Advanced Features
- Client - Graphical User Interface (GUI)
- GitHub - link

Program Code

A project, or more than one, with all code, well commented. Code must be uploaded as a zip file. Code

must also be available in a public GitHub repository, the repo must have a commit history, not a last-minute code dump. The link for the GitHub repository should be available in the report. Codes should be also included as appendix of the report.

Video Presentation

The presentation should be recorded by you with your camera and microphone on and with the camera view of you and the audio in your own voice included in the recording. The presentation should demonstrate the different parts of your project including the application. The presentation should not exceed a total of 10 minutes in duration.

Marking Scheme

Category	**Marks**
Project Proposal	3%
Report (Detailed technical write-up)	10%
Video Presentation & Demo (Clear explanation and demo of your work in a video)	10%
3 services with well-defined proto files (6% each)	6%
Using all 4 gRPC invocation styles (Simple, Server Streaming, Client Streaming, Bi-directional)	12%
3 functional service implementations (9% each)	27%
Service Implementations (Complexity & Functionality)	6%
Appropriate error handling for remote invocations and error messaging. Cancelling of messages	5%
Secure communication & authentication	5%
GUI to monitor & interact with services	6%
Use of appropriate frontend/backend technology	4%
GitHub Repository (Version Control)	6%