**A logo with black text

Description automatically generated**

**National College of Ireland**

**HDip in Computing**

**(HDSDEV\_JANBLY1\_O, HDCSDEV\_INT, HDSDEV\_SEP23\_HDBC\_SEP23OL)**

**Due Date: 19th April 2024**

**Divyaa Manimaran Elango, Sudarshan Deshmukh & Mark Cudden**

**School of Computing**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Distributed Systems**

**Continuous Assessment (CA) Type: Project**

**Weight:** The assignment will be marked out of 100. This CA is worth **60%** of the overall marks for this module.

**Instructions**

* This is an individual assessment.
* All analysis, design, code and report is to be generated by the individual student.
* Report to be written in word and provided in Word or pdf.
* All code is to be delivered in a zip file in working running order.
* Video presentations to be provided in .mp4 format.

**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.

**SUBMISSION DETAILS**

|  |  |
| --- | --- |
| **Deliverables** | **Week** |
| Project submission (including report, code, presentation) | Sunday, 19th April 23.55 P.M |

**Project Description**

For 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. 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 x17068381 then your industry would be “Smart home system” (last digit is “one”).

|  |  |
| --- | --- |
| **Last student digit of student id** | **Domain (Industry)** |
| **0** | Smart offices (connected lighting, smart meeting rooms) |
| **1** | Smart home system (smart devices, smart thermostats) |
| **2** | Smart water/air pollution tracking |
| **3** | Smart transport system (Smart traffic lights, parking) |
| **4** | Smart retail (Autonomous checkout etc.) |
| **5** | Smart warehouse (track stock, automate orders) |
| **6** | Smart hotels (Automated check-in/check-out etc.) |
| **7** | Smart education (Smart classrooms etc.) |
| **8** | Smart Agriculture/farming |
| **9** | Smart cities (automate traffic management, lighting) |

As part of your task, you must 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 should communicate via **gRPC**. **Nodejs/TypeScript or Java** must be used 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 or command-line based)** application should be developed, operating as a main controller that discovers and uses your devices/services.

**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. A brief project proposal must be first defined detailing the description of your application domain and **3 services**. This should be approx. of 1000 words.

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, with 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.
* Naming Services.
* Remote Error Handling & Advanced Features.
* Client - Graphical User Interface (GUI)/Command Prompts.
* GitHub.

**Project Code**

A project, or more than one, with all code, well commented. 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**

Presentation should be recorded and uploaded on Moodle page demonstrating the different parts of your application. The presentations should not exceed a total of 10 minutes in duration.

**TURNITIN**

All report submissions will be electronically screened for evidence of academic misconduct (i.e., plagiarism and collusion)

**Please refer to the Marking Rubric to be used for CA below.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Grade Criterion** | **H1 (> 70%)** | **H2.1 (> 60%)** | **H2.2 (> 50%)** | **Pass (> 40%)** | **Fail (< 40%)** |
| **Report: 10%** | Well written, with no (large) language errors. All figures are well-conceived and easy to read. The report does not exceed the length limits. References are complete, appropriately, and correctly used. | The report has a few language and/or style errors. The figures are well presented. Format and length limits are adhered to. References are complete, and correctly used. | The report is readable with some language and/or style errors. Some figures may be hard to read or presented in a suboptimal manner. References are mostly complete and correctly used. | The report is readable with many language and/or style errors. Most figures are not clear or easy to read. References are few and/or mostly incomplete. | Littered with typos, and/or poor use of English. The figures are poor and hard to read. References (if any) are probably incomplete and poorly used. |
| **Presentation/Viva 10%** | The presentation clearly outlined the project argument. Slides were error-free and logically presented. The speaker was poised and enthusiastic. Questions were excellently answered. | The presentation somewhat clearly outlined the project argument. Slides were somewhat error-free and somewhat logically presented. The speaker was poised and enthusiastic. Questions were very well answered. | The presentation outlined the project argument. Slides were mostly error-free and mostly logically presented. The speaker was poised and enthusiastic. Questions were well answered. | The presentation provided a limited outline of the project argument. Slides were not error-free and not logically presented. The speaker was poised and enthusiastic. Questions were reasonably well answered. | The presentation is unorganised and unclear. Questions were unanswered/poorly answered. |
| **Service Definitions – Use of gRPC (protos) 18%** | Service definitions are well-structured and clearly defined. gRPC usage demonstrates advanced understanding and implementation. Protobuf definitions are comprehensive and effectively utilised. | Service definitions are structured and adequately defined. gRPC usage shows proficiency with some minor issues. Protobuf definitions cover essential aspects but may lack depth in certain areas. | Service definitions are somewhat disorganized, with unclear or inconsistent definitions. gRPC usage demonstrates basic understanding but lacks sophistication. Protobuf definitions are incomplete or poorly implemented. | Service definitions lack structure and clarity, making it challenging to understand. gRPC usage is rudimentary or incorrect. Protobuf definitions are missing or improperly implemented. | Service definitions are absent or completely incorrect. gRPC usage and Protobuf definitions are not implemented or entirely ignored. |
| **Service Definitions - Use of gRPC (protos) (18%)**   * 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 \* 3 = 12%] | Accurately defines and uses proto files for each of the 3 services/devices with complete coverage.              Implements all four RPC invocation styles effectively across the services/devices, showcasing comprehensive understanding and successful integration. | Adequately defines and uses proto files for each of the 3 services/devices, with minor discrepancies in coverage.        Implements three out of four RPC invocation styles effectively across the services/devices, with minor issues or gaps in integration. | Incompletely defines or incorrectly uses proto files for each of the 3 services/devices, leading to gaps in coverage or inefficient integration.        Implements two out of four RPC invocation styles across the services/devices with noticeable deficiencies or limited integration. | Proto files are missing or entirely incorrect, resulting in significant gaps in coverage and ineffective integration.        Implements only one RPC invocation style across the services/devices, with significant deficiencies or incomplete integration. | Proto files are absent or completely wrong, rendering the gRPC implementation impossible or severely impaired.        Fails to implement any of the RPC invocation styles across the services/devices, resulting in a complete failure of gRPC functionality or system integration. |
| **Remote Error Handling & Advanced gRPC (10%)**     * Appropriate error handling for remote invocations and error messaging. Cancelling of messages [5%]                * Appropriate functionality of Error Handling [5%] | Effective error handling mechanisms are implemented for remote invocations, with clear and informative error messaging. Cancelling of messages is implemented correctly and efficiently.                  Error handling functionalities are comprehensive and effectively implemented, covering a wide range of potential errors and ensuring system reliability and stability. | Error handling for remote invocations is implemented with minor issues or inconsistencies in error messaging. Cancelling of messages is implemented but may have some limitations or inefficiencies.        Error handling functionalities are adequately implemented with minor issues or limitations, providing basic error management and system stability. | Error handling for remote invocations lacks clarity or completeness in error messaging, with some errors not properly handled. Cancelling of messages may be incomplete or improperly implemented.        Error handling functionalities lack completeness or effectiveness, with some errors not properly addressed or managed, potentially impacting system performance. | Error handling for remote invocations is inadequate or missing, leading to unclear error messages or unresolved errors. Cancelling of messages is either absent or poorly implemented.        Error handling functionalities are insufficient or missing key components, leading to frequent errors and system instability. | Error handling for remote invocations is entirely absent or incorrect, resulting in frequent errors and unclear messaging. Cancelling of messages is not implemented or misunderstood.            Error handling functionalities are entirely absent or fundamentally flawed, resulting in constant system failures or severe performance issues. |
| **Client - Graphical User Interface (GUI) /Command Interface (10%)**     * 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 [2 services \* 3 = 6%]      * The GUI/Command-line based client can be developed in any language, technology of choice (Java application, web-based, etc) [4%] | The client application provides comprehensive functionality for viewing, controlling, and invoking services/devices for each of the 3 services, with intuitive user interface design and seamless integration.                        The client application is developed using a suitable technology stack with advanced features and robust performance, meeting all requirements and exceeding expectations. | The client application offers basic functionality for viewing, controlling, and invoking services/devices for each of the 3 services, with some usability issues or limitations.            The client application is developed using a chosen technology stack with basic features and acceptable performance, meeting most requirements but with some limitations or shortcomings. | The client application lacks certain features or has limited functionality for viewing, controlling, and invoking services/devices, with noticeable usability challenges or inefficiencies.          The client application is developed using a technology stack with limited capabilities or outdated technologies, resulting in suboptimal performance or functionality. | The client application has minimal functionality or lacks integration with services/devices, making it difficult or impractical to use for viewing, controlling, or invoking.            The client application is developed using a technology stack that does not fully support the required features or fails to meet essential requirements, leading to significant deficiencies or issues. | The client application does not fulfill the requirements of viewing, controlling, and invoking services/devices, rendering it unusable for its intended purpose.              The client application is not developed using an appropriate technology stack or fails to meet the basic requirements, making it unsuitable for its intended purpose. |
| **GitHub (6%)**   * Maintain a repo with a regular commit history | The GitHub repository is actively maintained with a regular commit history, demonstrating consistent updates and contributions over time. | The GitHub repository shows regular activity with a commit history that may have occasional gaps or periods of inactivity, but overall demonstrates a commitment to maintaining the project. | The GitHub repository has sporadic commit activity, with irregular updates and periods of inactivity, indicating inconsistent maintenance. | The GitHub repository lacks regular commit history, with infrequent updates and long periods of inactivity, suggesting poor maintenance practices. | The GitHub repository has no commit history or activity, indicating a complete failure to maintain the project on GitHub. |