ASSESSMENT BRIEF	
Module Title:	Advanced Programming
Module Code:	KF7014
Academic Year / Semester:	2024-2025 / Semester 1
Module Tutor / Email (all queries):	Dr Maria Salama maria.salama@northumbria.ac.uk
% Weighting (to overall module):	100% (individual component 60%, group component 40%)
Assessment Title:	Summative assessment – Development of microservices-based application
Date of Handout to Students:	Week commencing 07th October 2024
Mechanism for Handout:	Module Blackboard Site
Deadline for Attempt Submission by Students:	*Individual component:* 20th November 2024 4pm *Group component:* 13th January 2025 4pm
Mechanism for Submission:	upload to module Blackboard site
Submission Format / Word Count	*Individual component:* Development project zip file and technical report PDF file *Group component:* Development project zip file and technical report PDF file. Group demonstration schedule and location will be announced on Blackboard.
Date by which Work, Feedback and Marks will be returned:	Week commencing 10th February 2025
Mechanism for return of Feedback and Marks:	Mark and individual/group written feedback sheet will be uploaded to the Module Blackboard site. For further queries, you can email the module tutor.

LEARNING OUTCOMES

The learning outcomes (LOs) for this module are:

LO1. Demonstrate in-depth knowledge and understanding of current best professional practice in the design and development of applications.

LO2. Design and develop a system that demonstrates a professional understanding of current software development best practices and explore opportunities for innovation.
LO3. Work as part of a self-organising team in the implementation and testing of a software system, using appropriate advanced techniques.
LO4. Demonstrate the effectiveness of the implemented application, in adhering to current software development best practices and critically evaluate your work.
LO5. Demonstrate a professional understanding of the importance of legal, social, and ethical requirements in producing a system that holds third party information. In particular, the need for application-level security both in terms of access control and data security.

This assessment addresses the learning outcomes, as follows:

- Individual component will assess LO1, LO2, and LO5.
- Group component will assess LO1, LO2, LO3, LO4, and LO5.

ASSESSMENT OUTLINE

In this assessment, you will develop a microservices-based application for smart electricity management in smart cities.

Background

Comprehensive smart cities systems enable local authorities to monitor different aspects, such as electricity, water, gas, traffic, parking, etc. Smart Cities systems using Internet of Things (IoT) offer new and innovative ways to monitor electricity supply and mitigate related challenges. Monitoring electricity supply can help citizens to manage their consumption, providers to plan their demands, and local authorities to monitor carbon footprint.

Scenario

The North East region is planning a new pilot project for smart electricity management in a number of cities (Newcastle, Sunderland, Durham, Middlesbrough, Darlington) with three electricity providers (A, B, C). Each city will develop their own system along with microservices of the three different power supply providers. Data will, then, be collected from these cities into a unified interface presenting the North East region.

Assessment structure

This assessment is composed of two components:

The individual component

In this component (individual work – worth 60% of the module mark), you will develop microservices based on current best practices in software development. The microservices should work independently to be utilised and consumed by a front-end application in the group component.

The group component

In this component (group work – worth 40% of the module mark), you will work in a self-organising group to develop a modular front-end application that consumes the individual microservices of the group members. Each group member will develop a front-end component that will interact with their individual microservice, to display and visualise data in the front-end application.

The group will present their work via a 20-minutes live demonstration, where they will have the opportunity to critically evaluate their work and showcase the application's effectiveness in adhering to current software development best practices.

ASSESSMENT INSTRUCTIONS

For the individual component, each student will develop microservices for smart electricity management of one city. For the group component, each student will be part of a self-organising group, where each member of the group will present a different city. The individual and group components shall collectively form a microservices-based application for smart electricity management for the North East region. Specifications of each component are detailed below.

INDIVIDUAL COMPONENT

This component includes microservices development and technical report.

Microservice development

For the individual component, you will create microservices responsible for: (i) collecting electricity consumption from citizens for the three different electricity providers, (ii) collecting data from the different providers in one city. The **citizen microservice** is responsible for the collection of electricity consumption (in kWh) both manual collection through citizen entry and automated collection through smart meter reading. Each citizen/ smart meter submits data to their contracted electricity provider. The **electricity provider microservice** is responsible for collecting and storing data in real-time as long as it is running. Data should be stored permanently in a database. **The smart city microservice** is responsible for collecting data from the different providers of that city, collating and aggregating data.

The developed solution should be structured in the following projects and fulfil the following functionalities:

- Citizen microservice – This project should simulate the smart meters responsible for randomly generating data within a specified data range, as well as the citizens for manual data input. Each meter reading should be more than the previous reading.
- Electricity Provider microservice – This project should contain the API responsible for receiving data from the smart meters and citizens' input. Data should be received in real-time, i.e. at the time of input of the citizen microservice.
- Smart City microservice – This project should contain the API responsible for requesting data from the electricity provider microservice, collating and aggregating data, as well as permanently storing the aggregated data in a database.
- Main program – This should simulate the Smart City system; running one instance of the Smart City microservice, three instances of the Electricity Provider microservice, and 100 citizens for manual entry and 50 smart meters for automated entry contracted with each provider.
- Testing project – This should include Unit Testing for all the microservices.
- Data folder – This folder should include the database files used to permanently store the collected data with sample data stored.

The developed solution should fulfil the following technical requirements:

- Development in Java using any code editor or IDE (e.g. Eclipse).
- Implementing API Gateway for the communication between the microservices.
- Using a suitable database (e.g., MongoDB, SQLite) to store the collected data. Each microservice should have its own database.
- Implementing error and exception handling.
- Testing APIs to ensure functionality and reliability.

- Writing code documentation.

The developed solution should follow the following best practices:

- Solution and projects should be well-structured.
- Object-oriented principles should be used in designing the solution.
- Functions should be used.
- Code conventions should be followed.
- Professional code styling and formatting should be followed.

Notes:

- The developed solution should not call any external APIs.
- The developed solution should not use any datasets for data collection.
- The submitted database files should not include large amounts of stored data, only sample data generated from one run.

Technical Report

The report should document the developed microservices-based solution, and should include:

- Introduction – (~ 100 words) provide a brief overview of the work that has been done, and the document contents that will be presented next to prepare the reader for the following sections.
- Architecture diagram – provide a diagram of the architecture of the developed solution, clarifying the communication between different microservices.
- Database Schema – provide the schema for the database used by the developed solution, in the form of clear diagrams, including tables, fields, and relationships.
- Class diagram – provide the design of the classes implemented in the developed solution, in the form of clear diagrams, including classes, interfaces, properties and functions.
- APIs documentation – list all APIs endpoints and their associated HTTP methods, e.g., GET, POST, PUT, DELETE, their Request and Response, and returned error codes. You can use screenshots of Postman.
- Critical evaluation – (~ 500 words) critically evaluate your product's inception to completion, i.e. what have been done well, what could have been done better and how.
- References – include references to relevant external documentation or resources.

The report should be 600 words. The approximate length defined above in each section is for guidance. More details about the requirements of the report are listed below. You should use the template provided on Blackboard.

Mechanism for submission

The following files should be submitted:

- one zip file (this MUST be a zip file, NOT any alternative, such as rar file) containing a zipped copy of all the files of the development project (i.e. the whole running project including the database folder).
- one PDF file (this MUST be a PDF file, NOT any alternative, such as word file) of the technical report.

Files should be submitted using the corresponding link provided under Assignment Submission in the Assessment and Submission area on Blackboard for this module by 20th November 2024 4pm UK time.

GROUP COMPONENT

Students are expected to work in self-organising group towards the group component, bringing their individual microservices to the application. This component includes the application development, reflective account and the group demonstration.

Application Development

For the group component, you will design and develop a **modular front-end web application** that consumes the microservices of the individual components. Each group member will develop a front-end component that will interact with their individual microservices, presenting data to the front-end UI. The web application should provide user-friendly dashboard for visualising data collected by different providers and cities.

The developed solution should fulfil the following functionalities:

- User authentication and authorisation – Implement user authentication and authorisation for secure access to the web interface. This should include the creation and deletion of users, as well as password change.
- Data View – Implement data view to display historical data of the different components. This should include: (i) data table for each city displaying aggregated data from the different providers – each student is responsible for developing the data table of their city, and (ii) data table for each provider displaying aggregated data from different cities.
- Data Visualisation – Create interactive data visualisation (e.g., charts) for presenting data trends. This should include: (i) graph displaying monthly average of each provider in each city separately – each student is responsible for developing the graph of their city, and (ii) graph displaying monthly average of all cities. Average data should be calculated, not stored in a new database.
- Testing – This should include integration testing in a separate project within the solution. Do not include unit testing of individual microservices.

The developed solution should fulfil the following technical requirements:

- Development in Java using any code editor or IDE (e.g. Eclipse) and UI library.
- UI design should be modular.
- Using RESTful APIs for communication between the front-end UI and microservices.
- Using an in-memory for user creation and authentication.
- Implementing error and exception handling.
- Integration testing to ensure components are working together properly.
- Writing code documentation.

The developed solution should follow the following quality and best practices:

- Solution and projects should be well-structured, having UI in a separate project and each individual component in a separate project.
- Object-oriented principles should be used in designing the solution.
- Functions should be used.
- Versioning of the different APIs and the web application should be used.
- Code conventions should be followed.
- Professional code styling and formatting should be followed.

Notes:

- The developed solution should not store any calculate averages in a new database.
- The developed solution should not use any ready design template for the UI (this is not a design or UI module).

Reflective Account

The reflective account should reflect on the developed solution and the agile development process. This should include:

- Introduction – (~ 100 words) provide a brief overview and description of the application, explaining its purpose and functionality, and the document contents that will be presented next to prepare the reader for the following sections.
- Reflection on the developed solution – this section is related to the developed solution only, and should include:
    - Social, ethical, and security concerns – (~ 300 words) describe social, ethical, and security issues related to the developed solution and the data.
    - Critical evaluation – (~ 300 words) critically evaluate your product's inception to completion, i.e. what have been done well, what could have been done better and how. Note that this section is related to the product only. Do not include critical evaluation of individual microservices.
- Reflection on the development method – this section is related to the agile development method adopted by the group, and should include:
    - Agile development method – (~ 300 words) describe the agile development method adopted by the group and reflect how agile principles were applied.
    - Sprint Planning, Execution and Reviews – (~ 400 words) (i) Sprint Planning – reflect on the process of planning each sprint, whether the goals were realistic, and whether the backlog well-organised and prioritised; (ii) Task Management – describe how tasks were broken down, assigned, and tracked using tools, as well as reflect on whether and how the team adapted when things did not go according to plan; (iii) Sprint Reviews – reflect on how did the team review and demonstrate work at the end of sprints; and (iv) Iteration and Continuous Improvement – reflect on how the iterative nature of Agile helped or hindered progress, and whether there were improvements noticeable between iterations. You should include any necessary diagrams here, with the Project Plan to be added to Appendix 1.
    - Outcome, Results and Areas for Improvement – (~ 400 words) (i) Project success – reflect on the success of the project in meeting its goals, and whether you were able to deliver quality solution; (ii) Future improvements – reflect on what could be improved in future Agile projects and whether there are better practices or tools that could be adopted; and (iii) Lessons Learned – summarise key lessons learned, and how these lessons can be applied to future projects.
- Personal reflections – (500 words for each student) each student should write their own section separately, including: reflection on your own experience in the Agile environment, what you learned about working in Agile, how your own role evolved over time, and how Agile helped in developing any skills.
- References – include references to relevant external documentation or resources.
- Appendix 1 – Project plan (sprints, tasks and assignments) using any forms, tools or templates adopted by the group.
- Appendix 2 – Logbook of group work (details below).
- Appendix 3 – Peer assessment (details below).

The report should be 2000 words, excluding personal reflection (500 words for each student). The approximate length defined above in each section is for guidance. More details about the requirements of the report are listed below. You should use the template provided on Blackboard.

Logbook of group work

Each group should keep a logbook of their meetings and all communication. The logbook should include at least date, attendees, meeting notes/discussion, allocated tasks, and updates from each member. The logbook of group work should be maintained consistently throughout the project and serve as a comprehensive record of the project activities, decisions, and progress. The logbook should be accessible to all the group members and the module leader (i.e. shared online).

The logbook shall facilitate feedback provision, lead to coaching, and help to manage the group work more effectively toward the final product.

The logbook should be included as Appendix 2. You should use the template provided on Blackboard.

Peer assessment

Each student must have a peer assessment form, and each form must be agreed and signed (electronically signed, or printed, signed and scanned) by all the members of the group. Peer assessment forms of all group members should be submitted as Appendix 3.

In the peer assessment form, the performance of each student in the group is graded by several criteria, which measures your peers' views of the quality of your work within the group. The peer assessment criteria are appended in this document and a template word file will be available on Blackboard. The peer assessment mark will be calculated as an average of these criteria and will be used to scale the total mark of the group component of each student.

Mechanism for submission

- one zip file (this MUST be a zip file, NOT any alternative, such as rar file) containing a zipped copy of all the files of the development project (i.e. the whole running project including database files).
- one PDF file (this MUST be a PDF file, NOT any alternative, such as word file) of the technical report and appendices.

Files should be submitted using the corresponding link provided under Assignment Submission in the Assessment and Submission area on Blackboard for this module by 13th January 2025 4pm UK time.

Demonstration will be scheduled during the w/c 20th January 2025 on-campus during working day. Schedule and location will be announced in due course on Blackboard.

Group demonstration

The group will present their work via a 20-minutes live demonstration. All the members of the group should attend the group demonstration and present their own work (i.e. contribution to the group product).

The group demonstration should include a live demo of the running product, and showcase the application's effectiveness in adhering to current software development best practices.

The demonstration should make clear how all the components have been implemented, architecture and design decisions made on the way, as well as security, social and ethical considerations. The presentation has no predefined structure; however, you must cover all the previous points and ensure that all functionalities you have created are shown.

At the end of the demo, you will be asked about the agile development method followed by the group, including: (i) team dynamics (the role of each member and how collaboration and communication occurred within the team, as well as the challenges in team coordination or communication and how they were addressed; and (ii) issues faced during the Agile process, how the team overcome these challenges, and what processes or adjustments were implemented. You should answer the questions thoughtfully, providing details and specific examples.

Your demonstration will be recorded by Panopto. This is to provide evidence of your demonstration to the internal moderator and the external examiner. We will share a copy with the group via Blackboard.

Groups should also attend demonstration of other groups for reflection.

Inequity of contribution

Marks will properly reflect each student's performance levels, and where necessary, marks adjustments can be made to better reflect these levels. Such arrangements can address the issues or concerns raised by group members or observed by the module team of possible 'free riders' and ensure they are encouraged to contribute equitably and receive an appropriate grade if they do not. Such arrangements will also reward individual group members who carry a proportionally heavier load or who make a more significant contribution than their group colleagues. Each group member's contribution (as defined by predetermined criteria) will be assessed using evidence from the team's logbook and peer assessment.

REQUIREMENTS OF TECHNICAL REPORT AND REFLECTIVE ACCOUNT

The contents of the technical report and reflective account are outlined above in each respective component. The following requirements apply to both documents.

Report length

The length of each report and the length of each section in the report are defined above in the details of each Technical Report. This does not include title and content pages, tables, figures, references and appendices. Words count should be declared, as shown in the template. No penalty if the words count is under the word limit or up to 10% over word limit. If the words count is more than 10% over the word limit, the marker will stop reading when they judge that the word count exceeds the recommended word count by more than 10%. For word limits policy, refer to the Assessment Regulations and Policies in the Guidance for Students mentioned in the Assessment Regulations section below.

If your report needs less words count or space, you should not artificially fill in.

Referencing

References should be complete and in the correct format. This means you should reference all your information sources, giving enough citations to show exactly where you used each source. Claims and evidence hold more weight if supported by multiple sources.

A list of references should be given at the end of the document, presented in alphabetical order of authors. In CIS Department, we use the “Harvard Format” for referencing, using brief citations in the text, and giving the full reference in the “References” section at the end of the report. Check the format instructions in ‘Cite them Right’ (Pears and Shields, 2008) for the citations and reference list.

The sources you are referencing should be relevant to the report's content. References should include sources from academia and industry. You should include good-quality, reliable sources, not poor-quality, out-of-date, or unreliable ones. Do not use Wikipedia references. Do not use Generative-AI references. You will lose marks if you omit sources from your reference list or fail to cite them. Remember to provide your perspective in your own words, rather than paraphrasing extensively, and use quotations only when there is a very good reason to do so, etc.

Document presentation

The overall presentation covers several aspects of the documents, not just how good it looks.

The documents should be submitted using the given templates. You should fill in the template of the first page student/group members details (student name and ID). Within the main body of the report, you should have a clear structure (e.g., use the suggested sections, sub-headings and paragraphs). The report should also include a “References” section at the end. The document should be also well-formatted in terms of styles, spacing, etc. Figures, diagrams and screenshots should be clear, readable, numbered and labelled properly.

The spelling and grammar should be correct. The language used should be appropriate for an academic report. That means adopting an appropriate style and tone. Using colloquialisms, aphorisms, and emotive terms is not acceptable. Of course, obscenities, prejudicial statements, and inflammatory comments are unacceptable. While you may passionately hold a position, you should avoid ranting and expressing your views professionally. The use of “I” and “we” is often avoided by the use of “passive voice” in technical reports. For example, “I/ We explored several Smart City systems” would be replaced by “Several Smart City systems were explored”. However, you can use “I” and “we” in the reflective account. For example, you can write in the reflective account “In this project, we adopted agile method as a team” and “In this project, I encountered several challenges when designing the UI”.

A good professional report/ white paper should be readable by intelligent people in different disciplines. If you are using specialist or technical jargon, explain the terminology. If your use of language is so poor that the meaning is obscured, you will also lose marks. A well-structured report uses an understanding of how the reader will best understand the ideas presented, and the points will build on each other logically.

Notes

- Both technical report and reflective account will be checked for plagiarism and use of Generative AI.

PEER ASSESSMENT CRITERIA

Criteria	Unsatisfactory	Poor	Satisfactory	Good	Excellent	Outstanding	Rationale – What is the justification for the score?
Demonstration of relevant skills and knowledge							
Attendance at group activities							
Contribution to group activities							
Contribution to agreed tasks outside of group meetings							
Quality of work on agreed tasks							
Working for consensus on decisions and attempts to resolve conflict rather than promote it							
Trust, support, and respect other team members							
Ability to listen and interpret communication from other's points of view							

Generates and promotes ideas and suggestions of their own							
Considers and uses new ideas and suggestions from others							

ASSESSMENT REGULATIONS

You are advised to read the Guidance for Students regarding assessment policies.

Late submission of work

Where coursework is submitted without approval, after the published submission deadline, the following penalties will apply:

- For coursework submitted up to 1 working day (24 hours) after the published hand-in deadline without approval, 10% of the total marks available for the assessment (i.e.100%) shall be deducted from the assessment mark.
- Coursework submitted more than 1 working day (24 hours) after the published hand-in deadline without approval will be regarded as not having been completed. A mark of zero will be awarded for the assessment and the module will be failed, irrespective of the overall module mark.

These provisions apply to all assessments, including those assessed on a Pass/Fail basis.

Academic Misconduct

The Assessment Regulations for Taught Awards (ARTA) contain the ***Regulations and procedures applying to cheating, plagiarism and other forms of academic misconduct***.

You are reminded that plagiarism, collusion and other forms of academic misconduct as referred to in the Academic Misconduct procedure of the assessment regulations are taken very seriously. Assignments in which evidence of plagiarism or other forms of academic misconduct is found may receive a mark of zero.

Use of Generative AI

The use of Generative AI tools (e.g. Chat GPT, Google Bard) and code completion (e.g. Github Copilot, Amazon Code Whisperer) is permitted, but must be acknowledged – you will not 'lose marks' for using legitimate developer tools and acknowledging this). It is NOT allowed to use any 'human in the loop', 'cheating' or 'ghosting' services (even if branded as AI); i.e. do not pay someone to do your work, do not ask someone to do your work.

The use of Generative AI for writing the technical report and reflective account is strictly prohibited.

Opportunities for feedback

Formative feedback will be delivered on request, in lab and office hours. Exercises similar to each component will also be provided for formative feedback.

Summative feedback will be delivered at most 20 working days after assessment submission.

Notes

- Each student should work on the individual component individually without any interactions with other group members. Details of the individual component should not be communicated to the group prior to the submission deadline of the individual component.
- The submitted individual component should be used as it is in the group component without any further modifications. In case minor modifications are required, this should be documented in the logbook and the APIs should be versioned.
- In case one individual component is not integrated properly in the group component due to mal-developed individual component or lack of effort from that student, inequity of contribution will apply.
- In case the logbook and/or peer assessment forms are not submitted, the group component mark will be capped to 60%. If the peer assessment form of one student is not submitted, this student's mark for the group component will be capped to 60%.
- In case one member of the group has missed the group demonstration without an approved reason, this student's mark for the group component will be capped to 60%.
- A mark of zero will be awarded in case of failure to adhere to the mechanism for submission instructions, including wrong files format, missing files of the developed project, not using templates provided, or not using respective submission links.

ASSESSMENT MARKING SCHEME

Individual component (weight 60% to overall module)	**100 Points**
Microservices development	**83**
Functionalities	**43**
Citizen microservice	
Electricity Provider microservice	
Smart City microservice	
Main program	
Unit Testing	
Technical requirements	**26**
API Gateway	
Database	
Error and exception handling	
Code documentation	
Quality practices	**14**
Project structure	
OO design	
Functions	
Code conventions	
Technical documentation	**17**
Contents	**12**
Architecture diagram	
Database schema	
Class diagram	
APIs documentation	
Critical evaluation	
Quality of document	**5**
Quality of contents	
References	

Group component (weight 40% to overall module)	**100 Points**
Application development	**56**
Functionalities	**32**
User authentication and authorisation	
Data view	
Data visualisation	
Integration Testing	
Technical requirements	**19**
Modular UI	
REST communication between UI and microservices	
In-memory database of users	
Error and exception handling	
Code documentation	
Quality practices	**5**
Project structure	
OO design	
Functions	
Code conventions	
Reflective account	**44**
Contents	**19**
Reflection on solution	
Reflection on agile method	
Project plan	
Logbook	
Quality of document	**5**
Quality of contents	
References	
Demonstration	**10**
Personal reflection	**10**

ASSESSMENT MARKING CRITERIA

General criteria are given below to be applied as a percentage to each component.

Since the elements above are wide ranging, general criteria are given that are applied as a percentage to each component of the assessment. In the following, 'writing' is understood to apply both to coding and English.

Fail/ Pass	Percentage	General Criteria
Fail	0-29%	A very poor contribution showing little awareness of subject area. The developed solution is static, lacks the development of microservices, using appropriate database, or not implemented required functionalities, or not using appropriate technologies. Significant reliance on external code fragments, making it difficult to assess student comprehension. Communication of knowledge is either inarticulate and/or irrelevant and lacks clarity.
	30-39%	Knowledge is limited or superficial. Some awareness of concepts, but there are major omissions or misunderstandings. The developed solution is poorly structured and fails to fulfil several key functionalities and technical requirements. Coding is unstructured and has significant errors. Writing is not clear without suitable presentation of relevant arguments.
	40-49%	Knowledge is barely adequate. A basic understanding of the key issues is demonstrated, but insufficient focus is evident in the work presented. The developed solution does not provide appropriate functionalities and fulfil some technical requirements. Code has minor/few errors. Writing is not fluent and/or is mostly description and/or assertion are used rather than argument or logical reasoning.
Pass	50-59%	Knowledge base is up-to-date and relevant to an appropriate breadth and depth for level 7. The student has demonstrated the ability to apply theory and concepts and identify important relevant issues. The developed solution provides most functionalities and fulfil most technical requirements. Code is clear and accessible, but little uneven. Writing is fluent, and written work is supported by appropriate references.
	60-69%	As above, but there is clear evidence of independent thought and reasoned conclusions. The developed solution provides appropriate functionalities, and some initiative may be shown in better development and/or design of the solution. Coding standards are mostly professionally followed, and security measures are mostly applied. Writing is fluent, focused, accurate and mostly supported by citations using appropriate references.
	70-85%	Excellent scholarship is demonstrated. The developed solution provides good functionalities development, professional fulfilment of technical requirements, as well as good design of the solution. Initiative is shown in developing the solution beyond the required specification. Coding standards are professionally followed, and

		security measures are applied. Writing is fluent, focused, well-researched, and supported by citations using appropriate references. The submission clearly exceeds taught material.
	86-100%	Exceptional scholarship is demonstrated. The developed solution provides excellent functionalities development, professional fulfilment of technical requirements, as well as excellent design of the solution. Exceptional initiative is shown in thoroughly developing the solution beyond the required specification. Coding standards are professionally and consistently followed, and excellent security measures are applied. Writing is exceptionally comprehensive, fluent, well-researched and convincing, supported comprehensively by good set of references. The submission clearly exceeds taught material.