IN-COURSE ASSESSMENT (ICA) SPECIFICATION

|  |  |
| --- | --- |
| Module Title:  **Cloud Computing DevOps** | Module Leader:  **Tyrone Davison** |
| Module Code:  **CIS3032-N** |
| Assignment Title:  **ThAmCo System** | Deadline Date:  **Tuesday 7th January 2025** |
| Deadline Time:  **4:00pm** |
| **Submission Method:**  Online (Blackboard) þ  Middlesbrough Tower ¨ |

|  |
| --- |
| **Online Submission Notes:**   * Please follow carefully the instructions given on the Assignment Specification * When Extenuating Circumstances (e.g. extension) has been granted, a fully completed and signed Extenuating Circumstances form must be submitted to the School Reception or emailed to [scedt-assessments@tees.ac.uk](mailto:scedt-assessments@tees.ac.uk). |

FULL DETAILS OF THE ASSIGNMENT ARE ATTACHED  
INCLUDING MARKING & GRADING CRITERIA

Cloud Computing DevOps (CIS3032-N)  
Assessment Specification

Contents

[Introduction 1](#_Toc146470563)

[Scenario 3](#_Toc146470564)

[Instructions 5](#_Toc146470565)

[Deliverables 7](#_Toc146470566)

[Guidance 7](#_Toc146470567)

[Marking Criteria 8](#_Toc146470568)

[Learning Outcomes 9](#_Toc146470569)

[Apprentice Knowledge, Skills and Behaviours (KSB) 9](#_Toc146470570)

# Introduction

This document describes the assessment for the module. It is an entirely individual in-course assignment. The first stage of the assignment requires you to complete an architecture design for an enterprise-style, cloud-capable, software system to satisfy the problem scenario given in this document. The next stage requires you to select and implement some of the individual system containers that would contribute towards the software system you proposed. The final stage requires you to demonstrate the resultant software system and the DevOps tools and practices you have used to deploy it and requires you to provide a narrative report to explain the consequences of the technology choices you made.

Read the sections below for the precise details of what you must do and submit.

*Apprentices on the Digital and Technology Solutions course will normally undertake a negotiated work-based project that is agreed by the employer, apprentice and module leader/tutor. The project will be tailored to the apprentice’s job role and the in-house software platforms available to the employer. However, where required, the case study or sections of it may be used.*

# AI Permission

This module has an AI Permission of Amber. You can, if you wish, use AI as a companion for specific tasks, explicitly permitted in the instructions section below, when producing your assessment work.

# Scenario

You work as a software developer for the fictitious ThreeAmigos Corp. The company aims to:

* Buy products from approved third-party suppliers.
* Re-sell the products to registered customers via a web app.

You are responsible for designing, constructing and deploying a new software system to help deliver the above based upon the functional requirements below. The system should be well engineered to enable growth and new business opportunities to be easily incorporated. You are required to utilise only management approved technologies.

The public should be able to:

* Browse and filter the products we sell (regardless of stock status).
* Loose search (within name and descriptor) for products.
* Register (name, email address, payment address) with our web app.

Registered customers should be able to:

* Securely sign into the web app.
* Update their profile (edit their name, delivery address and telephone number).
* See the stock status of products when browsing – stock status needs updating at least every 5mins.
* See how much money (funds) they have available to spend with us – assume their account funds are topped up outside of this system.
* Order a product (if the stock is available and they have sufficient funds in their account) – a delivery address and telephone number must exist in the customer’s profile, and we must successfully purchase the stock from our supplier before completing the order with the customer.
* Receive an email when an order is created or when its status is updated containing the details.
* View their order history and the status of each order.
* Request their account be deleted.

Staff should be able to:

* View the list of orders that need to be dispatched and mark an order as dispatched (recording the date and time it was).
* View the profile, funds, and order history (including product details) for any customer.
* Delete a customer account – personal data must be erased/anonymised, but all other data retained.

Product requirements:

* Products must come from a given list of source suppliers.
* Each product must only appear once in listings regardless of how many suppliers provide it.
* Products should be listed using the cheapest price from our suppliers plus 10%.
* Product catalogue and prices should automatically update once per day.

# Instructions

1. You have been provided with the beginnings of a software architecture diagram (this can be found on Blackboard). You should analyse the system requirements as defined in the scenario section above and relate it to the diagram provided. You will then **create one or more architecture diagrams** (e.g. C4 container diagram, deployment and infrastructure diagram). These diagram(s) should identify the system containers and their:

* Technologies (e.g. ASP.NET Core 8, SQL Server, etc.)
* Interfaces (e.g. endpoints/URLs)
* Connectivity (e.g. direction of GETs and POSTs)
* Protocols (e.g. HTTPS, HTML)
* Deployment (e.g. which machine, resource group, etc)

You may use or replace whatever you wish from the provided diagram. You may use AI as a companion for working through architecture ideas. Obtain formative feedback on your proposed software architecture design from your tutor.

1. You will **create a development plan** detailing which containers from the overall system you are going to construct. The implementation of the system is therefore divided amongst a larger (virtual) team. Your development plan must:

* Be approved by the tutor, usually **within two weeks** of starting the assessment. Once approved, it should only be revised with tutor agreement.
* Require approximately 20% of the effort that would be required to construct the whole software system and include at least two software containers.
* Address all the assessment criteria (later in this document) to ensure that you can achieve full marks, especially connectivity between at least two software containers and demonstration of security.
* Identify the containers, the system requirements addressed, the key features and the interfaces to other containers. You may use AI as a companion here.
* Use the template provided on Blackboard.

1. You will **implement** your development plan**.** When constructing your software containers:

* Create an application codebase that supports **automated testing** and **configuration management** (test/live deployment).
* Ensure they are **secure**, and that they are **resilient** in the event of failures in the network or other system containers.
* **Fake** interfaces to components that would be developed by others upon which your software will interact or depend.
* Use a **DevOps** workflow (Git commits leading to automated building and testing leading to automated deployment).
* Perform weekly **integration tests** to ensure that the containers created and provisioned function together correctly within a deployed environment.

You may use AI as a companion for resolving coding problems during your development work.

1. You will **write a** **short narrative** explaining and taking responsibility for the consequences of the technology related decisions you took during the project.
2. You will **submit** your **architecture diagram(s)**, **development plan**, **narrative,** and the **code** for your completed system containers, together with **demonstration media** (screenshots and screen movie captures) and a completed **marking proforma** (template provided on Blackboard) directly to *Blackboard* as detailed in the deliverables section below. Your demonstration mediamust show the level of **resilience** and **security** that has been achieved, along with the **DevOps practices and tools** that you used.

# Deliverables

You must submit your work as a ZIP file via *Blackboard* with the following directory structure:

* ***Media***: this directory should contain full resolution screen shots (PNG or JPG) and short (20-30 seconds) full resolution movie captures (MP4 H.264) of your work, showing both the containers working and your DevOps tools and practices.
* ***Source***: this directory should contain the entire source code for your contribution to the system, including any necessary project and solution configuration files. Ensure that you have *cleaned* (i.e. deleted intermediate and non-essential files for) the project(s) by deleting the *build*, *release*, *debug, bin, and obj* directories, and deleting all the *packages* sub-folders.
* ***Documentation***: this directory should contain any documents created to demonstrate and evidence that your work meets the requirements of the marking criteria (e.g. architecture diagrams, narrative, marking proforma).

Include a *readme* file in the root directory containing your name and user id.

# Guidance

Use an agile / iterative development method with the tutors as *product owners* to give external drive and direction to the project.

Use source control (e.g. Git) and other development tools (e.g. bug trackers). Use source control fully (i.e. branch, merge, and tag) as appropriate to your dev method.

Do not become obsessed with functional requirements at the expense of the other marking criteria – the module is about your DevOps practices.

Perform frequent integration testing in a deployed environment of your containers. Do not assume it will all just come together successfully at the end.

Start the assignment’s required documentation early and develop it iteratively. Use it productively to structure the work, rather than as something to develop just for marking purposes at the end.

# Marking Criteria

| **Criterion** | **Excellent (85-100%)** | **Very Good (70-85%)** | **Good (55-70%)** | **Satisfactory (40-55%)** |
| --- | --- | --- | --- | --- |
| **System Architecture**  **(30%)** | System containers and their technology, interfaces, connectivity, protocols, and deployment are documented clearly and any deviations in the implementation identified.  Design appears plausible and scalable using recognised industry approaches. | System containers and their technology, interfaces, connectivity, protocols, and deployment are documented in sufficient detail.  Design appears plausible and potentially scalable, although approaches are not necessarily industry standard. | System containers are documented with much of the required information, but there is a general lack of clarity.  Design is workable but there appear to be risks for the scalability. | System containers are documented but the details of how they will work are lacking.  Design is potentially workable but the lack of detail means it is not demonstratable. |
| **System Implementation**  **(70%)** | Requirements implemented to be distributed, feature appropriate authorisation, and continuation following failure.  A comprehensive range of suitable industry-standard tools and frameworks used thoroughly and extensively.  Thorough automated verification testing integrated within a CI/CD pipeline demonstrated using test doubles to their full potential.  Suitable continuous software delivery pipeline through multiple environments with configuration demonstrated.  Narrative explains consequences of tech choices and shows ownership of any mistakes and failures. | Requirements implemented to be distributed with appropriate call resilience and feature suitable authorisation.  A reasonable range of suitable industry-standard tools and frameworks used thoroughly and extensively.  Thorough automated verification testing demonstrated using test doubles to their full potential.  Suitable automated software build and deployment with environment-based configuration demonstrated.  Narrative explains consequences of tech choices and accountability for them, but without taking ownership. | Requirements implemented with distribution across networked containers, but more should be done on security and resilience.  Suitable industry-standard tools and frameworks used satisfactorily.  Suitable automated verification testing demonstrated, but it is not sufficiently extensive.  Suitable automated software build and deployment demonstrated with environment aware configuration in code.  Narrative explains consequences of tech choices but does not attribute accountability. | Requirements implemented with suitable partitioning via databases, but there is little to address resilience or security.  Basic use of suitable industry-standard tools and frameworks is demonstrated.  Automated verification testing demonstrated, but it is not entirely suitable.  A suitable manual build and deployment process demonstrated.  Narrative only explains some consequences of the tech choices. |

# Learning Outcomes

On successful completion of this assessment, you will have shown you can:

1. Concisely present a software system architecture effectively to both an operations and developer audience via diagram-oriented documentation.
2. Accept accountability for the consequences of technology related decisions taken.
3. Problem-solve during the development and deployment of a complete software system, using a wide range of computer programming languages and technologies.
4. Demonstrate a comprehensive knowledge of, and experience with, a wide range of industry-standard software development platforms, security mechanisms, frameworks, libraries and DevOps tools.
5. Design of a cloud-capable software system, using architectural and software patterns, and determine suitable cloud-based platform and infrastructure solutions to be used for its deployment.
6. Incorporate professional DevOps and security practices into the development and deployment of one or more working cloud-capable software components.

# Apprentice Knowledge, Skills and Behaviours (KSB)

This module contributes to the following apprenticeship KSBs:

|  |  |
| --- | --- |
|  | **Core Skills** |
| **C1** | **Information Systems:** |
|  | Is able to critically analyse a business domain in order to identify the role of information systems, highlight issues and identify opportunities for improvement through evaluating information systems in relation to their intended purpose and effectiveness |
| **C2** | **Systems Development** |
|  | Systems Development: analyses business and technical requirements to select and specify appropriate technology solutions. Designs, implements, tests, and debugs software to meet requirements using contemporary methods including agile development. Manages the development and assurance of software artefacts applying secure development practises to ensure system resilience. Configures and deploys solutions to end users. |
| **C3** | **Data** |
|  | Identifies organisational information requirements and can model data solutions using conceptual data modelling techniques. Is able to implement a database solution using an industry standard database management system (DBMS). Can perform database administration tasks and is cognisant of the key concepts of data quality and data security. Is able to manage data effectively and undertake data analysis. |
| **C6** | **IT Project Management** |
|  | follows a systematic methodology for initiating, planning, executing, controlling, and closing technology solutions projects. Applies industry standard processes, methods, techniques and tools to execute projects. Is able to manage a project (typically less than six months, no inter-dependency with other projects and no strategic impact) including identifying and resolving deviations and the management of problems and escalation processes. |
|  | **Core Technical Knowledge** |
| **C8** | How business exploits technology solutions for competitive advantage. |
| **C9** | The value of technology investments and how to formulate a business case for a new technology solution, including estimation of both costs and benefits. |
| **C10** | Contemporary techniques for design, developing, testing, correcting, deploying and documenting software systems from specifications, using agreed standards and tools. |
| **C11** | How teams work effectively to produce technology solutions. |
| **C13** | Common vulnerabilities in computer networks including unsecure coding and unprotected networks. |
| **C14** | The various roles, functions and activities related to technology solutions within an organisation. |
|  | **Core Behavioural Skills** |
| **C19** | Makes concise, engaging and well-structured verbal presentations, arguments and explanations. |
| **C21** | Is able to identify the preferences, motivations, strengths and limitations of other people and apply these insights to work more effectively with and to motivate others. |
| **C22** | Competent in active listening and in leading, influencing and persuading others. |
| **C23** | Able to give and receive feedback constructively and incorporate it into his/her own development and life-long learning. |
| **C24** | Applies analytical and critical thinking skills to Technology Solutions development and to systematically analyse and apply structured problem solving techniques to complex systems and situations. |
| **C29** | Ability to perform under pressure |
| **C30** | A thorough approach to work |
| **C31** | Logical thinking and creative approach to problem solving |
|  | **Specialism Outcomes - Software Engineer** |
|  | **Skills** |
| **SE1** | Create effective and secure software solutions using contemporary software development languages to deliver the full range of functional and non-functional requirements using relevant development methodologies. |
| **SE2** | Undertake analysis and design to create artefacts, such as use cases to produce robust software designs. |
| **SE3** | Produce high quality code with sound syntax in at least one language following best practices and standards. |
| **SE4** | Perform code reviews, debugging and refactoring to improve code quality and efficiency. |
| **SE5** | Test code to ensure that the functional and non-functional requirements have been met. |
| **SE6** | Deliver software solutions using industry standard build processes, and tools for configuration management, version control and software build, release and deployment into enterprise environments. |
|  | **Technical Knowledge (knows and understands)** |
| **SE7** | How to operate at all stages of the software development lifecycle. |
| **SE8** | How teams work effectively to develop software solutions embracing agile and other development approaches. |
| **SE9** | How to apply software analysis and design approaches. |
| **SE10** | How to interpret and implement a design, compliant with functional, non-functional and security requirements. |
| **SE11** | How to perform functional and unit testing. |
| **SE12** | How to use and apply the range of software tools used in software engineering. |