Diploma in Information Technology
CIT2C22 - DevOps Essentials
Project Specification
AY2025/2026 October Semester

Introduction

You are required to form a group comprising of **3 to 4 members**. The project aims to mirror a real-world development team making collaborative efforts to infuse DevOps principles and practices in development projects. There are no restrictions on the type of web application your team develops; however, you must not use a Resource Management Web Application, as this example is already covered in class.

Your team and you must adhere to the DevOps principles and practices during development, testing and deployment. Each team member is expected to develop **ONE feature** with a level of complexity ranging from average to high. You will be **assessed individually** based on your contribution, understanding of the DevOps principles and practices and own implementation of the advanced features.

The project consists of three parts and involves developing a working Node.js web application.

1. **Part 1** involves developing the **backend and frontend of your proposed feature** and preparing a document outlining the Node.js web application (supported by JSON database), your contributions and reflections on version control and collaboration.

2. **Part 2** involves implementing **backend unit testing, API testing and frontend testing** for your proposed feature, as well as gathering **code coverage**. You will provide a document detailing your test cases and offering an analysis of the results.

3. **Part 3** involves setting up a Continuous Integration/Continuous Deployment (CI/CD) **pipeline** to **automate the build, testing, and deployment processes**.

Please note the use of **Generative AI tools** — including but not limited to ChatGPT, GitHub Copilot, and similar platforms — for both **report writing** and **generation of code or script** are **strictly prohibited** in all parts of the project. Violations will be **subjected to disciplinary actions**.

You may, however, use Generative AI to support your understanding of concepts or to research potential additional features — but not for writing of the report or generation of any part of the actual code or script.

Part 1a: Implementation of DevOps Practices - Web Application (20%)

For Part 1 of the project, **each member** is expected to fully implement the **backend and frontend code of their proposed feature (strictly ONE each)**. Each feature must demonstrate a complexity level ranging from average to high. As a guide, features that **interact with the database** and **encompasses alternate scenarios (success and at least two other error cases)** will typically meet the required complexity level. Your proposed features should differ from another team member's proposed features. Note that the team MUST use JSON files for data persistency – no other databases are allowed.

The **frontend** is assumed to **handle input validation**, so the **backend** only needs to **manage success and error responses**. Each member's proposed feature must be unique and the team must use **JSON files** for **data persistency** – other databases are NOT allowed.

Each student is to produce an **individual report** independently on the Node.js web application developed by your team. The development process must follow the DevOps principles and practices that were discussed in class.

Implementation Guidelines:

You are required to adhere to the folder and file structure defined in class:

- Each member shall have his/her **own backend .js file** in **utils** folder
  - YourName]Util.js (e.g., JohnTanUtil.js - YOUR backend logic)
- Each member shall have his/her **own frontend .js file** in **public/js** folder
  - your-name.js (e.g., john-tan.js - YOUR frontend logic)
- The following files are to be **shared** among members:
  - index.js
  - public/index.html
  - public/css/styles.css
  - utils/your-database.json (e.g., students.json – team's JSON data storage)
  - utils/your-database.template.json (e.g., students.template.json)

The report (refer to LMS for template) should NOT exceed **8 pages** and must include the following minimum requirements:

Project Overview (approx. 2 pages)	Provide a **summary of your web application**, including: • Application name and all key features, including those developed by other team members • Project timeline for Project Part 1 and evidence of project/task management tools (e.g., Jira) • Visual aids (recommended): E.g.: Use Case Diagram, Wireframe

Individual Contribution (approx. 2 pages)	Describe the **feature you proposed and developed**, focusing on: • How users interact with it (inputs, actions, expected outputs) • How it integrates with the database • Visual aids (recommended): E.g.: Sequence Diagram, Flowchart, Decision Tree
Version Control (approx. 2 pages)	Explain your **team's approach to source code management**, covering: • Project directory structure, naming conventions, version control system and branching strategy • Code conflicts faced during code merging and how they were resolved or prevented • Include relevant evidence and insights for improvement
Additional Features (if implemented – approx. 2 pages)	Discuss any **SCM-related enhancements or tools** you implemented, with supporting evidence and rationale. It is important to note that mere addition of features does NOT qualify as an additional feature unless it illustrates the capability of the DevOps tool. The exploration of different SCM tool(s) may be regarded as a valid additional feature. You are required to include references that guided you in the implementation of these additional features or enhancements.

You are free to be as creative as you can but do ensure that the web application can be completed within the specified timeframe and aligns with TP policies (e.g., gambling sites are not permitted). Keep in mind that the focus of this subject is on applying DevOps principles and tools, rather than solely on building the web application itself. You are encouraged to discuss your web application ideas with your tutor.

Some suggested applications include but are not limited to the following:

Sample Applications	**Possible CRUD (each member implements ONE feature)**
Blogging Platform	• Create a new blog post, view the content of a post, update the post, delete a post. • Add new comments to blog posts, retrieve comments for a post, update comment, delete a comment.
Recipe Manager	• Create a new recipe, view recipe details, update a recipe, delete a recipe. • Add new ingredients, view all ingredients, update an ingredient's details, remove an ingredient.

Student Management System	• Add a new student, view student details, update student information, delete a student. • Add new courses, view courses offered, update course details, delete a course.
Task Manager	• Create new tasks, retrieve all tasks, update task information, delete tasks. • Create new categories, update category, retrieve tasks by category, delete categories.

The suggested additional features include but are not limited to the following:

Sample Additional Features
Use of .gitkeep - Guide provided in LMS
Use of .gitattributes
Use of GitHub Issues
Use of GitHub Labels and Milestones
Use of SCM related tools: GitKraken, LazyGit, GitLens, etc

Note that the use of paid services or any service requiring credit card information will NOT be accepted as additional feature. Similarly, branch protection and Dependabot implementation will NOT be considered, as these can only be configured by the repository owner.

Project Part 1a Submission Instructions

Submit your Microsoft Word document into TP LMS under the Assessments folder. Save your report as:

YourName_StudentID_YourClass_Part1.docx

e.g.: *JohnTan_1234567D_P01_Part1.docx*

Please submit the Word document by **Week 6 (24 Nov, Monday, 9.00AM)** and ensure that the online submission is successful and the uploaded file is correct and not corrupted.

Part 1b: Implementation of DevOps Practices – Presentation and Demonstration (10%)

Each group is expected to do a group presentation and demonstration during the assignment evaluation in **Week 6 (Week of 24 Nov) - Timings to be advised by your tutor**.

The group presentation and demonstration must NOT exceed **15 minutes**. Each member is expected to have a part in presenting the adopted DevOps practices and/or performing the live demonstration of the web application. The presentation slides should cover, but are not restricted to the following information:

- Full names and matriculation numbers of all team members, and class
- Overview of the features in web application
- Clear explanation of team's SCM strategies and best practices.
- Live demonstration showcasing SCM and features of web application.
- List of additional features contributed by each team member (if any).

Project Part 1b Submission Instructions

Assign **one member per group** to submit the project source code as a zipped file and the PPT slides into TP LMS under the Assessments folder.

Before submission, ensure that all branches have been merged into the main branch. Then, pull the latest code from main and compress the project files into a zipped file.

The zipped file and PPT slides should be named according to the following format:

GroupNumber_YourClass.zip
e.g.: *01_P01.zip*

GroupNumber_YourClass.pptx
e.g.: *01_P01.pptx*

Please submit the zipped file and PPT slides by **Week 6 (24 Nov, Monday, 9.00AM)** and ensure that the online submission is successful and the uploaded file is correct and not corrupted.

Part 2a: Automated Testing – Implementation (20%)

In Part 2 of the project, each team member will **clone the group repository** and **work independently** on a separate private repository to implement **backend unit tests, API tests and frontend tests**, as well as gather the **code coverage** for their proposed feature.

You are **NOT** required to set up the CI/CD workflow. Additionally, you do **NOT** need to implement backend unit testing, API testing and frontend testing of your group member's features. The test cases and code coverage should focus only on your own feature.

You are to submit your **project source code** and produce an **individual report** that comprehensively covers the test cases you implemented and includes your reflection on the test results.

The report (refer to LMS for template) should NOT exceed **10 pages** and must include the following minimum requirements:

Introduction (approx. 1 page)	Provide a **brief overview of your proposed feature**, including: • A clear description of its purpose and role within the web application • The names of your backend and frontend files

Testing and Analysis (approx. 7 pages)	You are required to perform backend unit testing, API testing, and frontend testing for your proposed feature. Your documentation should include:
	• **Objectives**
	  ○ Describe the purpose of the test suites and test cases for each testing type (backend unit, API, and frontend).
	• **Code Coverage**
	  ○ Present code coverage results for both backend and frontend tests.
	  ○ Support your results with evidence such as screenshots, reports, or logs.
	  ○ Provide analysis and discussion of the results.
	  ○ Explain contributing factors or reasons affecting the results.
	• **Test Files**
	  ○ Specify the names of your test files for each testing component.
	• **Reflection**
	  ○ Provide a short summary of testing effectiveness, key lessons learned and possible improvements for future testing.

Additional Features (approx. 2 pages)	Describe any **testing-related enhancements** you implemented, supported by evidence and rationale. It is important to note that mere addition of features does NOT qualify as an additional feature unless it illustrates the capability of the Testing tool. Include **references** to resources that guided your implementation.

The suggested additional features include but are not limited to the following:

Sample Additional Features
Frontend code coverage threshold - Guide provided in LMS
Alternative testing tool/framework
Alternative code coverage tool/framework

Note that the use of paid services or any service requiring credit card information will NOT be accepted as additional feature.

Project Part 2 Submission Instructions

Submit your Microsoft Word document into TP LMS under the Assessments folder. Save your report as:

YourName_StudentID_YourClass_Part2.docx
e.g.: *JohnTan_1234567D_P01_Part2.docx*

Submit the project source code as a zipped file into TP LMS under the Assessments folder. The zipped document file should be named according to the following format:

YourName_StudentID_YourClass_Part2.zip
e.g.: *JohnTan_1234567D_P01_Part2.zip*

Please submit the above by **Week 13 (12 Jan, Monday, 9.00AM)** and ensure that the online submission is successful and the uploaded file is correct and not corrupted.

Part 2b: Automated Testing – Presentation and Demonstration (10%)

Each student is required to perform an **individual demonstration of their implemented test cases and code coverage** during the assignment evaluation in **Week 13 (Week of 12 Jan)**. Timings will be advised by your tutor.

The demonstration must **NOT** exceed **10 minutes** and **NO** presentation slides are required

During the live demonstration, you will be required to run your backend unit tests, API tests, and frontend tests to demonstrate the testing process.

Part 3a: Continuous Integration/ Continuous Deployment (CI/CD) - Pipeline Implementation (25%)

Project Part 3 requires you to **create a customised CI/CD pipeline** using **Jenkins**, **Docker**, and **Minikube**. The pipeline should **automate** the **build**, **testing**, and **deployment** processes.

To fulfil the requirement for Self-Directed Learning (SDL), you MUST attempt to implement at least **ONE additional feature**. You are required to document the process in the PPMR report. Even if the additional feature implementation is unsuccessful, you are still expected to **complete** the four sections: **Plan**, **Process**, **Monitoring** and **Reflection**.

You are to submit your **project source code** and produce an **individual report** documenting your CI/CD configurations and processes.

The report (refer to LMS for template), excluding the PPMR section, should NOT exceed **8 pages** and must include the following minimum requirements:

CI/CD Pipeline Setup (approx. 5 pages)	Document your **pipeline implementation** across build, test, and deployment phases. Include evidence such as screenshots, configuration files, or logs showing your pipeline setup of each phase. Describe any challenges faced and how you resolved them.
Additional Features (approx. 3 pages)	Implement at least **one additional feature** related to **CI/CD**, **containerisation**, and/or **container orchestration**. It is important to note that mere addition of features does NOT qualify as an additional feature unless it illustrates the capability of the CI/CD, containerisation and/or container orchestration tool. Include **references** to resources that guided your implementation.
PPMR Report	Journal the **Plan**, **Process**, **Monitoring**, and **Reflection** sections for ONE additional feature.

The suggested additional features include but are not limited to the following:

Sample Additional Features
Implementation of Visual Tools/Dashboard- Guide provided in LMS
Implementation of Jenkins Email Notification
Adoption of different CI/CD tool (e.g., Travis CI, GitHub Actions)
Adoption of different containerisation tool
Adoption of different container orchestration tool

Note that the use of paid services or any service requiring credit card information will NOT be accepted as additional feature. If you wish to explore cloud-based orchestration, you may use Microsoft Azure, which offers Student account with USD 100 credits and does not require a credit card.

Project Part 3a Submission Instructions

Submit the project source code as a zipped file and your Microsoft Word document into TP LMS under the Assessments folder. Save your files as:

YourName_StudentID_YourClass_Part3.zip
e.g.: *JohnTan_1234567D_P01.zip*

YourName_StudentID_YourClass_Part3.docx
e.g.: *JohnTan_1234567D_P01.docx*

Please submit the above by **Week 17 (9 Feb, Monday, 9.00AM)** and ensure that the online submission is successful and the uploaded file is correct and not corrupted.

Part 3b: Continuous Integration/Continuous Deployment (CI/CD) – Presentation and Demonstration (15%)

Each student is expected to do an individual demonstration of their CI/CD pipeline during the assignment evaluation in **Week 17 (Week of 9 Feb) - Timings to be advised by your tutor**.

The demonstration must **NOT** exceed **10 minutes** and **NO** presentation slides are required.

During the live demonstration, you will be required to make **minor code modifications** and commit your changes to the SCM system to demonstrate the CI/CD process. You will have only **ONE** attempt to do this. To ensure the demonstration stays within the 10-minute time limit, please **adjust your build trigger to a 1-minute** interval, particularly if it is currently set to poll the SCM for changes.

File Submission and Backup Policy

When submitting assignments, each student is responsible for ensuring that they do not submit an empty zip file or a corrupted file.

Students must also ensure they are using a suitable device when working on assignments / projects, maintain proper backups of their work and do not cite 'laptop issues' as a reason for assignment extensions as such reasons will not be considered.

Penalty for Late Submission

late and <1 day : 10% deduction from absolute mark given for the assignment
late >=1 and <2 days : 20% deduction from absolute mark
late >=2 days : No marks awarded

Project - Grading Criteria

The grading criteria for **Part 1: Implementation of DevOps Practices (30%)** will be based on the following:

Criteria	In Context	Performance Level				
		Excellent	**Good**	**Average**	**Poor**	**Below Standard**
Web Application (20%)	**Project Overview (5 marks)**	Synopsis of web application includes ALL the following • Application name, • Key features Project timeline includes all essential elements (milestones, tasks, deadlines). Evidence(s) of tools/ methodologies used is present and well-documented. Writeup is clear and comprehensive. A variety of visual aids are used and all have effectively helped to enhance clarity.	Synopsis of web application includes ALL the following • Application name, • Key features Project timeline is mostly complete but may be missing one elements (milestones, tasks, deadlines). Evidence(s) of tools/ methodologies used is present but lacks details. Writeup is mostly clear but lacks some details. At least one visual aid used had effectively helped to enhance clarity.	Synopsis of web application includes ALL the following • Application name, • Key features Project timeline is incomplete and missing two elements (milestones, tasks, deadlines). Evidence(s) of tools/ methodologies used is not present or poorly documented. Writeup is mostly clear but lacks some details. Visual aid is used but does not contribute to clarity.	Synopsis of web application includes SOME of the following • Application name, • Key features Project timeline is missing or missing all elements (milestones, tasks, deadlines). Evidence(s) of tools/ methodologies used is not present or poorly documented. Writeup is mostly unclear. Visual aid is not present.	Section is largely incomplete, non-submission, or Clear evidence of plagiarism detected (disciplinable offence)
	Individual Contribution (10 marks)	Contributed feature is well-documented with user and database interactions. Writeup is clear and comprehensive. A variety of visual aids are used and all have effectively helped to enhance clarity.	Contributed feature is documented with some detail but may lack completeness. Writeup is mostly clear but lacks some details. At least one visual aid used effectively helped to enhance clarity.	Contributed feature is documented with minimal detail. Writeup is mostly clear but lacks some details. Visual aid is used but does not contribute to clarity.	Contributed feature is minimally documented. Writeup is mostly unclear. Visual aid is not present.	

	Version Control (10 marks)	Writeup includes all details regarding project directory structure, naming conventions, version control system and branching strategy, supported by evidence(s). Writeup on conflict resolution or prevention within the group is insightful, well-documented and supported with evidence(s). Reflections on improvements on team's SCM strategy are insightful and practical.	Writeup includes some details regarding project directory structure, naming conventions, version control system and branching strategy. Writeup on conflict resolution or prevention within the group is valid, well-documented and supported with evidence(s). Reflections on improvements on team's SCM strategy are valid and practical.	Writeup includes little details regarding project directory structure, naming conventions, version control system and branching strategy. Writeup on conflict resolution or prevention within the group is valid, documented and supported with some evidence(s). Reflections on improvements on team's SCM strategy are valid.	Writeup missing details regarding project directory structure, naming conventions, version control system and branching strategy. Writeup on conflict resolution or prevention within the group is absent or vague, poorly documented and supported with little or no evidence(s). Reflections on improvements on team's SCM strategy is absent or vague.	Section is largely incomplete, non-submission, or Clear evidence of plagiarism detected (disciplinable offence).
	Implementation Guidelines (5 marks)	Perfectly adheres to the defined folder and file structure	Mostly adheres to the folder and file structure with minor discrepancies.	Some key issues with folder and file structure.	Significant issues with folder and file structure.	
	Feature Implementation (5 marks)	Fully implements both backend and frontend code for the proposed feature, meeting or exceeding the required complexity.	Implements both backend and frontend code with good functionality, but with some minor issues or missing features.	Implements basic backend and frontend functionality, but with significant gaps in complexity.	Implements backend and frontend functionality, but with many missing or broken features.	

	Additional Features (5 marks)	Writeup demonstrates that students have clear understanding of the additional features and rationales provided are relevant and valid. Multiple additional features related to SCM are implemented meaningfully with no bugs and errors and well-documented with evidence(s) and reference(s).	Writeup demonstrates that students have good understanding of the additional features and rationales provided are somewhat relevant and valid. At least two additional features related to SCM is implemented meaningfully with no bugs and errors and well-documented with evidence(s) and reference(s).	Writeup demonstrates that students have some understanding of the additional features and rationales provided are weak. At least one additional feature related to SCM is implemented with no bugs and errors and documented with evidence(s) and reference(s).	Writeup demonstrates that students have some understanding of the additional features. At least one additional feature related to SCM is implemented with no bugs and errors and documented with evidence(s) and reference(s).	No attempts of any DevOps-related additional feature, or Clear evidence of plagiarism detected (disciplinable offence)
Presentation and Demonstration (10%)	**Delivery (10 marks)**	Live demonstration provides clear explanation and insights on the team's SCM strategies. Visual aids are well prepared and are used to make presentation more effective.	Live demonstration provides explanation and some insights on the team's SCM strategies. Visual aids are adequate and appropriately used.	Live demonstration provides little explanation and insights on the team's SCM strategies. Visual aids are adequate but not well used	Live demonstration is largely unclearly and/or provides little insights on the team's SCM strategies. Very little or poor use of visual aids	Student was absent for project presentation and demonstration without a valid reason Clear evidence of plagiarism detected (disciplinable offence)
	Q&A (10 marks)	Answers all technical/code-related questions confidently, including reasoning behind implementation choices; explains key parts of the code accurately and in detail Present within the given time-limit.	Answers most technical/code-related questions confidently; gives reasonable explanation of code and logic. Present within the given time-limit.	Struggles with some code-related questions; explanations lack depth or clarity.	Unable to respond to most code or technical questions; shows limited understanding of own code.	

The grading criteria for **Part 2: Automated Testing (30%)** will be based on the following:

Criteria	In Context	Performance Level				
		Excellent	**Good**	**Average**	**Poor**	**Below Standard**
Test Cases Implementation and Analysis (20%)	**Introduction (4 marks)**	Brief overview includes ALL the following • Feature name, • Success case • Error cases, and Writeup is clear and comprehensive.	Brief overview includes MOST of the following • Feature name, • Success case • Error cases, and Writeup is mostly clear but lacks some details.	Brief overview includes SOME of the following • Feature name, • Success case • Error cases, and Writeup is mostly clear but lacks some details.	Brief overview includes SOME of the following • Feature name, • Success case • Error cases, and/or, Writeup is mostly unclear.	Section is largely incomplete, non-submission, or Clear evidence of plagiarism detected (disciplinable offence).
	Files identification (6 marks)	Test files and files related to test cases are all correctly identified and labelled	Test files and files related to test cases are mostly correctly identified and labelled.	Some test files and files related to test cases are correctly identified and labelled.	Few test files and files related to test cases are correctly identified and labelled.	
	Unit Testing (15 marks) **API Testing (15 marks)** **Frontend Testing (15 marks)**	Test cases accurately identify and cover ALL the following: • Positive and negative tests • Edge cases • All branches • Error handling and exceptions validations Writeup provides a clear, comprehensive and accurate explanation of the test logic for all test cases. Demonstrates a deep understanding of how each test validates functionality.	Test cases identify and cover ALL the following: • Positive and negative tests • Edge cases • All branches • Error handling and exceptions validations Writeup provides a clear and mostly accurate explanation for most test cases. Demonstrates good understanding of the test logic, though a few details or justifications may be missing or slightly unclear.	Test cases identify and/or cover MOST of the following: • Positive and negative tests • Edge cases • All branches • Error handling and exceptions validations Writeup provides basic explanations for some test cases. Demonstrates partial understanding of the test logic, lacks clarity or completeness in describing expected behavior or reasoning.	Test cases identify and/or cover SOME of the following: • Positive and negative tests • Edge cases • All branches • Error handling and exceptions validations Provides incomplete or vague explanations for test cases. Shows minimal understanding of the test logic or incorrectly describes expected outcomes.	

	Code Coverage (10 marks)	Writeup includes and correctly labeled code coverage results demonstrating comprehensive code coverage (⩾90%), supported by evidence(s).	Writeup includes and correctly labeled code coverage results demonstrating good coverage (75–89%), supported by evidence(s).	Writeup includes and correctly labeled code coverage results demonstrating moderate coverage (60–74%), supported by evidence(s).	Writeup includes and correctly labeled code coverage results demonstrating limited coverage (40–59%), supported by evidence(s).	
	Reflection (5 marks)	Reflections on testing effectiveness are insightful and comprehensive, demonstrating clear understanding of what worked well and what didn't. Key lessons learned and proposed improvements are specific, practical, and show depth of thought for future testing.	Reflections are valid and practical, showing good understanding of testing effectiveness and providing reasonable lessons learned and improvements. Some points may lack depth or detail.	Reflections are valid but basic, identifying a few lessons or improvements with limited analysis of testing effectiveness	Reflections are vague or incomplete, showing minimal understanding of testing effectiveness or providing only general statements without clear lessons or improvements.	
	Additional Features (10 marks)	Writeup demonstrates clear understanding of the additional features and rationales provided are relevant and valid. Multiple additional features related to testing are implemented meaningfully with no bugs and errors and well-documented with evidence(s) and reference(s).	Writeup demonstrates good understanding of the additional features and rationales provided are somewhat relevant and valid. At least two additional features related to testing is implemented meaningfully with no bugs and errors and well-documented with evidence(s) and reference(s).	Writeup demonstrates some understanding of the additional features and rationales provided are weak. At least one additional feature related to testing is implemented with no bugs and errors and documented with evidence(s) and reference(s).	Writeup demonstrates some understanding of the additional features. At least one additional feature related to testing is implemented with no bugs and errors and documented with evidence(s) and reference(s).	No attempts of any DevOps-related additional feature, or Clear evidence of plagiarism detected (disciplinable offence)
Presentation and Demonstration (10%)	**Q&A (10 marks)**	Answers all technical/code-related questions confidently, including reasoning behind implementation choices; explains key parts of the code accurately and in detail Present within the given time-limit.	Answers most technical/code-related questions confidently; gives reasonable explanation of code and logic. Present within the given time-limit.	Struggles with some code-related questions; explanations lack depth or clarity.	Unable to respond to most code or technical questions; shows limited understanding of own code.	Student was absent for project presentation and demonstration without a valid reason Clear evidence of plagiarism detected (disciplinable offence)

The grading criteria for **Part 3: Continuous Integration/Continuous Deployment (CI/CD) (40%)** will be based on the following:

Criteria	In Context	Performance Level				
		Excellent	**Good**	**Average**	**Poor**	**Below Standard**
Pipeline Implementation (15%)	**CI/CD Pipeline - Build (15 marks)**	Writeup is clear and comprehensive. CI/CD pipeline for build is implemented successfully with no bugs or errors. Writeup includes good set of evidence.	Writeup is mostly clear. CI/CD pipeline for build is implemented successfully with 1 or 2 bugs or errors. Writeup includes good set of evidence.	Writeup is mostly clear. CI/CD pipeline for build is implemented successfully with some bugs or errors. Writeup includes some evidence.	Writeup is vague. CI/CD pipeline for build is implemented successfully with mostly bugs or errors. Writeup includes few evidence.	Section is largely incomplete, non-submission, or Clear evidence of plagiarism detected (disciplinable offence).
	CI/CD Pipeline - Test (15 marks)	Writeup is clear and comprehensive. CI/CD pipeline for test is implemented successfully with no bugs or errors. Writeup includes good set of evidence.	Writeup is mostly clear. CI/CD pipeline for test is implemented successfully with 1 or 2 bugs or errors. Writeup includes good set of evidence.	Writeup is mostly clear. CI/CD pipeline for test is implemented successfully with some bugs or errors. Writeup includes some evidence.	Writeup is vague. CI/CD pipeline for test is implemented successfully with mostly bugs or errors. Writeup includes few evidence.	
	CI/CD Pipeline - Deployment (15 marks)	Writeup is clear and comprehensive. CI/CD pipeline for deployment is implemented successfully with no bugs or errors. Writeup includes good set of evidence.	Writeup is mostly clear. CI/CD pipeline for deployment is implemented successfully with 1 or 2 bugs or errors. Writeup includes good set of evidence.	Writeup is mostly clear. CI/CD pipeline for deployment is implemented successfully with some bugs or errors. Writeup includes some evidence.	Writeup is vague. CI/CD pipelines for deployment is implemented successfully with mostly bugs or errors. Writeup includes few evidence.	
	Reflection (10 marks)	Reflections on challenge(s) faced when working on CI/CD pipeline setup is relevant and insightful.	Reflections on challenge(s) faced when working on CI/CD pipeline setup is relevant.	Reflections on challenge(s) faced when working on CI/CD pipeline setup are vague.	Reflections on challenge(s) faced when working on CI/CD pipeline setup is brief.	

	Additional Features (20 marks)	Writeup demonstrates clear understanding of the additional features and reasons provided are relevant and valid. Multiple additional features are implemented meaningfully with no bugs and errors and is well-documented with evidence(s) and reference(s).	Writeup demonstrates good understanding of the additional features and reasons provided are somewhat relevant and valid. Some additional features are implemented meaningfully with no bugs and errors and is well-documented with evidence(s) and reference(s).	Writeup demonstrates some understanding of the additional features and reasons provided are weak. At least one unguided additional feature is implemented meaningfully with no bugs and errors and is well-documented with evidence(s) and reference(s).	Writeup demonstrates some understanding of the additional features. At least one additional feature is implemented meaningfully with no bugs and errors and is well-documented with evidence(s) and reference(s).	No attempts of any DevOps-related additional feature, or Clear evidence of plagiarism detected (disciplinable offence)
SDL Report (10%)	**SDL Process (50 marks)**	Shows great initiative in managing the tasks on hand and able to deliver valuable work on time throughout. Report is clear, well-structured and organised. Report demonstrates in-depth research and analysis.	Shows some initiative in managing the tasks on hand and able to deliver valuable work on time throughout. Report is well-structured and organised. Report demonstrates evidence of research and analysis.	Show minimal initiative in managing the tasks on hand and able to deliver valuable work on time throughout. Report has some structure and organisation. Report adequately addresses concepts from selected domain.	Show minimal initiative in managing the tasks on hand and able to deliver valuable work on time throughout. Report lacks structure and organisation. Report fails to adequately cover concepts from selected domain.	Non-submission or clear evidence of plagiarism detected (disciplinable offence)
Presentation and Demonstration (15%)	**Live Demonstration (5 marks)**	Committing the minor code successfully automates all phases (build, test, deploy) of the CI/CD pipeline with no bugs and errors.	Committing the minor code successfully automates at least 2 phases (build, test, deploy) of the CI/CD pipeline with no bugs and errors.	Committing the minor code successfully automates at least 1 phase (build, test, deploy) of the CI/CD pipeline with no bugs and errors.	Committing the minor code successfully automates at least 1 phase (build, test, deploy) of the CI/CD pipeline with some bugs and errors.	Student was absent for project presentation and demonstration without a valid reason Clear evidence of plagiarism detected (disciplinable offence)
	Q&A (10 marks)	Answers all technical/code-related questions confidently, including reasoning behind implementation choices; explains key parts of the code accurately and in detail Present within the given time-limit.	Answers most technical/code-related questions confidently; gives reasonable explanation of code and logic. Present within the given time-limit.	Struggles with some code-related questions; explanations lack depth or clarity.	Unable to respond to most code or technical questions; shows limited understanding of own code.	