Bachelor of Computer and Information Science in Application Development (BCAD313 and BCAD323)
INSY7315
Information Systems 3E
(WORK INTEGRATED LEARNING)
MODULE MANUAL 2025
(First Edition 2019)

The Independent Institute of Education (Pty) Ltd is registered with the Department of Higher Education and Training as a private higher education institution under the Higher Education Act, 1997 (reg. no. 2007/HE07/002). Company registration number: 1987/004754/07.

DID YOU KNOW?

Student Portal

The full-service Student Portal provides you with access to your academic administrative information, including:

- an online calendar,
- timetable,
- academic results,
- module content,
- financial account, and so much more!

Module Guides or Module Manuals

When you log into the Student Portal, the 'Module Information' page displays the 'Module Purpose' and 'Textbook Information' including the online 'Module Guides or 'Module Manuals' and assignments for each module for which you are registered.

Supplementary Materials

For certain modules, electronic supplementary material is available to you via the 'Supplementary Module Material' link.

Module Discussion Forum

The 'Module Discussion Forum' may be used by your lecturer to discuss any topics with you related to any supplementary materials and activities such as ICE, etc.

To view, print and annotate these related PDF documents, download Adobe Reader at following link below:
www.adobe.com/products/reader.html

IIE Library Online Databases

The following Library Online Databases are available. These links will prompt you for a username and password. Use the same username and password as for student portal. Please contact your librarian if you are unable to access any of these. Here are links to some of the databases:

Library Website	This library website gives access to various online resources and study support guides [Link]
LibraryConnect (OPAC)	The Online Public Access Catalogue. Here you will be able to search for books that are available in all the IIE campus libraries. [Link]
EBSCOhost	This database contains full text online articles. [Link]
EBSCO eBook Collection	This database contains full text online eBooks. [Link]
SABINET	This database will provide you with books available in other libraries across South Africa. [Link]
DOAJ	DOAJ is an online directory that indexes and provides access to high quality, open access, peer-reviewed journals. [Link]
DOAB	Directory of open access books. [Link]
IIESPACE	The IIE open access research repository [Link]
Emerald	Emerald Insight [Link]
HeinOnline	Law database [Link]
JutaStat	Law database [Link]

Table of Contents

Using this Manual

This manual has been developed to meet the specific objectives of the module, and uses a number of different sources. It functions as a stand-alone resource for this module and no prescribed textbook or material is therefore required. There may, however, be occasions when additional readings are also recommended to supplement the information provided. Where these are specified, please ensure that you engage with the reading as indicated.

Various activities and revision questions are included in the learning units of this manual. These are designed to help you to engage with the subject matter as well as to help you prepare for your assessments.

1. Introduction

An essential part of The Independent Institute of Education (The IIE) qualifications is to prepare students for a career in Information Technology.

This module gives you the opportunity to use all of your skills you have learned at The IIE; collaborate with your colleagues and develop a real-world application.

Your WIL project requires that you, in groups, create a complete Mobile App that tackle a serious issue that directly affects a disadvantaged community in South Africa. Examples include social housing, unemployment, education, crime, poverty, etc.

In approaching this problem, you will have to do research, conceptualise your ideas generate documentation for your mobile app.

Social responsibility is critical in our society, and each year your lecturer will select a Non-Profit Organisation (NPO) that has a critical need for a cutting-edge software solution. However, because the organisation is non-profit, they are not able to afford to pay a full-time development team. This also means that they are cost-conscious, and this should be taken into consideration when developing and building your solution.

Your goal is to develop a high-quality solution that the NPO can use sustainably into the future.

2. The Value of WIL

The purpose of having a WIL module in a qualification is to bring together all the knowledge and skills gained into one consolidated project thereby enabling you, the student, to integrate what you have learnt in several modules and demonstrate that you are able to work as a team and apply your knowledge to solve a real-world problem.

Through the WIL Modules additional attention can be given to what SAQA calls Critical Crossfield Outcomes (CCFOs) or what is now more generally known internationally as global competencies. These include:

CCFO1: Identify and solve problems in which responses demonstrate that responsible decisions using critical and creative thinking have been made.
CCFO2: Work effectively with others as a member of a team, group, organisation, community.
CCFO3: Organise and manage oneself and one's activities responsibly and effectively.
CCFO4: Collect, analyse, organise and critically evaluate information.
CCFO5: Communicate effectively using visual, mathematical and/or language skills in the modes of oral and/or written presentation.
CCFO6: Use science and technology effectively and critically, showing responsibility towards the environment and health of others.

CCFO7: Demonstrate an understanding of the world as a set of related systems by recognising that problem-solving contexts do not exist in isolation.

CCFO8: In order to contribute to the full personal development of each learner and the social and economic development of the society at large, it must be the underlying intention of any programme of learning to make an individual aware of the importance of:

- reflecting on and exploring a variety of strategies to learn more effectively.
- participating as responsible citizens in the life of local, national and global communities.
- **being culturally and aesthetically sensitive across a range of social** contexts.
- exploring education and career opportunities; and
- developing entrepreneurial opportunities.

The application of CCFOs or global competencies is largely context and discipline dependent.

INFORMATION FOR DEVELOPERS

Community Engagement:

Central to WIL is the need to simulate workplace activities to ensure that graduates can apply their learning in the world of work. Consequently, the development of the WIL should promote authentic learning activities. These activities should imitate what students will need to do in the workplace and promote students that are ready and willing to contribute to society and economy. This will ensure that our graduates are not only employable but global citizens. It is therefore paramount that our WIL reflects appropriate community engagement.

Although community engagement typically involves organised outreach, community research and service learning- it also encompasses important themes such as: 1) Citizenship 2) Values 3) Awareness/Consciousness- Raising 4) Skills and Knowledge (IIE019). It is not necessary nor advisable to include all these themes into one single WIL/POE. These should be appropriately crafted into the WIL to allow students to bridge the gap between theory and practice and to develop the skills needed to solve real-world problems, conduct themselves in an ethical and professional manner and to apply their knowledge in a context that can impact the well-being of others and the community.

In crafting these themes of community engagement, students should understand their role in the work group setting, how they would engage with colleagues and the impact that their work/career will have on the wider community. Students should also be given opportunities, where appropriate, to promote and practice respect and inclusion.

Here are some ideas for incorporating Community Engagement:

a) *Academic Content: It may be helpful to interrogate any links between the tasks, academic content, and any of the above themes. The intention here is not to talk about these themes but to empower the students with the skills and/or competencies to apply learning in the workplace and to appreciate their impact on the community.*

b) *Reciprocity: Seek opportunities where students need to work as a team and solve a problem. Emphasize the importance of communication, community voice (cultural understanding & involvement), equality and mutual impact to the successful completion of a task (Bandy, 2011).*

c) *Diversity: Expose students to diversity- this can be in groupwork activity or engaging/problem-solving in different contexts. This promotes identity development and challenges students to apply their skills in familiar and unfamiliar contexts (Bandy, 2011).*

d) *Ethical Dilemmas: Give students the opportunity to analyse a situation and gain practice in ethical decision making – this can be in a work environment, potential impact on the community or a legal issue where students must choose a course of action.*

e) *Underline Reflection: By highlighting the importance of reflection students are encouraged to think critically about the task/ exercise and its potential impact on the business, institution, colleagues and/or community. They are encouraged to challenge their attitudes, assumptions, beliefs and stereotypes. They are encouraged to transform a single task into broader issue awareness.*

f) *Creativity & Innovation: In a simulated assessment it is important to consider the type of assessment being utilised and how it relates to tasks students will perform in the world of work: directed writing (e.g., community experience/impact, community needs assessment, community problems- resources, processes and delivery, personal journals, lesson plans, letters, press releases, advocacy letter, policy review, photo essays and class presentations.*

3. WIL Role Players

WIL involves the following role players:

1. The **student** – the student is expected to attend all scheduled sessions (in person or in the case of distance students remotely), to meet deadlines, and collect and prepare evidence aligned to expectations as set out in the relevant WIL Module Manual.

2. The **WIL Coordinator** – takes responsibility for the overall operationalisation of WIL on a campus or for a group of students. The WIL Coordinator will also facilitate the initial engagement with the client.

3. IIE approved **lecturers** – designated to guide, mentor, assess and monitor students' academic progress in the WIL module.

4. The **client** is the customer who requires the software solution. The client will be responsible in evaluating the software solutions presented by each team.

A lecturer responsible for a WIL module may also be the designated WIL Coordinator.

4. Assessment of WIL

Assessment of Work-Integrated Learning (WIL) should be based on the design of the learning component of the programme expectations and stated outcomes. In addition, the assessment of WIL is governed by the principles in The IIE Assessment Strategy and Policy **(IIE009)** and the IIE Work-Integrated Learning Policy **(IIE006)**.

WIL modules are assessed primarily through:
- Portfolio of Evidence (PoE)
- GitHub repository
- Oral presentation and demonstration of the software product developed with your group
- Individual contribution must be evident in the PoE, GitHub and Oral presentation for marks to be allocated.

4.1 Portfolio of Evidence

A Portfolio of Evidence (PoE) is a collection of materials that illustrates a person's skills and capabilities. A PoE also typically includes reflecting on the learning process and is the place where documents are collected for one or both of the following purposes:
- To demonstrate student competence during a WIL process by putting together evidence of what they did, for example, analysis documentation, background research, design documents, evidence of testing, evidence of scrum processes, etc.
- Centralise all project documentation from WIL.

The student's submissions for their PoE should be in the form of a PDF document. This document should include all documentation relevant to your project.

The student will receive the WIL Module Manual halfway through the first semester, but the actual WIL runs in the second semester. The final submission of the portfolio will be after the 12th week in the second semester – normally in week 13. Alternatively, the WIL Module can be run in the second half of the year and start in the first week of the third semester, with a final submission in the fourth semester.

The PoE must minimally consist of the following:

1. Cover Page

    1. The Group name and members
    2. The Scrum Agile board URL, such as the Teamfu link (ensure that all markers have access)
    3. The Source code repository URL, such as the team's GitHub link (ensure that all markers have access)
    4. The Code Quality URL, such as the SonarQube project link (ensure that all markers have access)
    5. The Security report URL, such as the Snyk project link (ensure that all markers have access)
    6. The play store URL (if your app has been published)
    7. Any other URL relevant to your submission (E.g., website URLs should be included here)

2. Table of Contents

    This should minimally include the headings shown in this section.

    1. Introduction to the project
       Describe the project at a high-level and give the reader some background as to the problem you need to solve.

        - Work agreement
          Describe your team's work agreement, and team responsibilities for the project
        - Definition of ready (DoR)
          Describe your team's definition of ready.
        - Definition of Done (DoD)
          Describe your team's definition of done.
        - Roadmap (High-level plan)
          What was the high-level plan developed for your team. How long did you agree for sprints. Is everyone's availability planned for the duration of the project.

2. Requirements
   Include all requirement artifacts in this section. Include any research that your team did to better understand the organization, project and users of the software you will build.

   - User Roles
     Describe the various User Roles that your team has identified.
   - User Stories
     These should be in the standard format: As a (User role who wants to accomplish something) I want to (what they want to accomplish) so that (why they want to accomplish that thing)

     Include all User Stories in your team's backlog, with business priority and team estimation. Note which sprint they were included in and whether they have been successfully implemented or not.

   - User Experience Journey Map
     A User Experience (UX) Journey Map shows all the screens in your software application and how a user can navigate between these screens. Consider doing this for each application that you plan to build and remember to include different user roles.

3. Non-functional requirements
   This section should cover non-functional requirements gathered through interaction with your client.

4. Analysis artifacts
   This section contains information about how your team analysed the requirements to create domain model.

   - Domain modelling
     Show a diagram of the bounded contexts your team identified. Include a description of each bounded context that the team identified. It is recommended to use the Domain Driven Design approach to complete this section.

   - Design artifacts
     This section contains information about how your team analysed the domain model to create your implementation models.

5. Implementation Documentation
   This section should show your implementation document, such as UML sequence diagrams showing critical flows; deployment documentation; as well as any related documentation. Only important flows need to be documented.

6. Data Schema Documentation
   Document the data storage explaining your choices of technology, such as Entity Relationship Diagrams (ERD's). Include JSON schemas if using a document store technology (NoSQL). Describe and explain the diagrams and schema's that you include.

7. Architecture artifacts
   Describe which architecture patterns were used and why they were selected

   - Design Patterns
     Document your use of best-practice design patterns and explain your choices.
   - Architecture Patterns
     Documentation your choice of architecture pattern and explain your choices.
   - Cloud
     Describe your cloud architecture and discuss your technology decisions. Diagrams should include your cloud services and show network segregation and protocols used to communicate between services.

8. Security
   In this section, describe how security was addressed in the application. Address the standard security considerations, including networking, vulnerabilities and threats.

9. DevOps
   Rapid deployment of a high-quality application is pivotal to the success of a software product. This section should discuss how your team has addressed this non-functional requirement in your project.

   - GitHub Actions Pipeline
     Describe with use of a diagram the functionality in your GitHub pipeline.

10. Running costs
    Show predicted monthly costs of running the solution. Show growth projections for two (2) years that are based on input from client.

11. Change Management
    This section should describe your approach to managing the adoption of the software product that your team has built. Consider answering the following questions:
    - How and why will the organization adopt your software?
    - How and why will the users adopt your software?

- What is your strategy to gain adoption from both the organization and the users?

12. Appendices
    - Declaration of Authenticity
    - Scrum artifacts
      Evidence of using a Scrum board for collaboration; backlog prioritization; sprint planning; sprint reviews; and sprint retrospectives. Include an attendance roster for each Scrum event (ceremony).
    - Presentation rubric
      This should include lecturer's feedback from the final presentation.

4.2 GitHub repository

Your team's GitHub repository is the primary source of your project code. Your code will be evaluated with your PoE to check that there is alignment. Your GitHub repository is the evidence of everything you have documented in your PoE.

Your GitHub actions pipeline will also be evaluated.

Contributions to the GitHub repo will be considered as evidence of contribution to the project in the evaluation. Ideally everyone in the team should have contributed to your GitHub repository.

It is recommended that you create a single GitHub repository for your team at the start of the project and then give access to the rest of your team. Use this single repository for the duration of the project.

4.3 Oral Presentations

The WIL modules require students to deliver an oral presentation describing their project or activity to their client, WIL co-ordinator and lecturer(s). This will typically happen at the end of the project, i.e., towards the end of the WIL module. Each student in the group is to be evaluated according to the presentation rubric (Annexure E).

There are two main components of oral presentations, namely, a verbal component and a visual component. The **verbal** component focuses on the oral, or spoken, portion of the presentation during which aspects such as tone, delivery, language and audience engagement are assessed.

The **visual** component includes all other communication aids that are used during the presentation, for example, slides, video clips, posters, handouts, models, simulations, diagrams, websites, etc. The visual images created by the students themselves may be included here if they are relevant to the environment which is being represented. A typical example would be when a group of students is simulating the presentation of a proposal to a prospective client. In such instances, appropriate dress, posture and

body language are important. Visual aids used in presentations should be used effectively. For example, PowerPoint slides should **support** the presentation, but not **become** the presentation. Consequently, students need to think about both what they say, how they say it, what they use to support what they say, and how they are acting professionally and appropriately in a work-like environment.

5. Qualification Summary

Qualification Name: Bachelor of Computer and Information Science in Application Development (BCAD313 and BCAD323)

Qualification Code: BCAD313 and BCAD323

QUALIFICATION PURPOSE	EXIT LEVEL OUTCOMES
The purpose of this qualification is to qualify learners to follow a career in Software Application Development. The outcomes for this qualification were identified as critical to drive a successful career in software development providing competent desktop and mobile application developers in the private and the public sectors. By understanding business processes in the context of business rules, learners will be able to solve business problems and meet business needs through software application development. Learners will be suitably prepared to demonstrate competencies in application development, and to contribute to the economic well-being of their organisation in a responsible manner. The qualification design supports the logical progression in learning throughout the programme by introducing learners to the foundational and mathematical concepts, theories and fundamental knowledge in the first year to position them to master the more complex skills of analysing, interpreting and developing the principles and theories of desktop software development, mobile application development, cloud computing development and dynamic web development in the context of current IT trends and requirements. The design of the qualification will facilitate critical learning through the exposure to and application of specialised learning areas plus relevant support learning areas to enable the learner to manage the versatile and dynamic context of application development.	**ELO1**: Demonstrate the ability to apply key theories in the design and development of software applications. **ELO2**: Demonstrate the ability to provide software solutions ethically and professionally. . **ELO3**: Demonstrate the ability to design software applications on a variety of platforms. . **ELO4**: Communicate effectively and professionally as a member of a software design and development team **ELO5**: Demonstrate the ability to conduct research related to IT programming and Cloud Computing.

This qualification is designed to graduate learners with the ability to think and act strategically and professionally and to contribute meaningfully to the organisations that employ them. The programme design thus facilitates the development of a well-rounded software developer.	

6. MODULE SUMMARY

ITEM	DESCRIPTION
Faculty	Information Communications Technology
Qualification	Bachelor of Computer and Information Science in Application Development (BCAD313 and BCAD323)
Module Name	Information Systems 3E (Work Integrated Learning)
Module Code	INSY7315
Module Purpose	This module requires the students to integrate their knowledge and skills to develop software applications that meet specific given business requirements.
Module Outcomes	**MO1**: Identify software requirements for a new IT software system to meet given business requirements. **MO2**: Design the implementation plan to meet the pre-determined software requirements. **MO3**: Develop and implement the deliverables identified in the implementation plan. **MO4**: Create comprehensive documentation for each required deliverable for the development and implementation of the new IT software system. **MO5**: Collaborate as a group to produce all deliverables of the new IT software system.
Credits	15
Notional Hours	150 hrs (24 hrs contract)
Type of WIL	Project
Tools and Resources	Personal Computer Microsoft® Office 365 Internet Microsoft Teams (or Trello, Slack, Confluence) Project Management Software Microsoft Visual Studio, Android Studio (or any other IDE) GitHub Note additional tools recommendations in ANNEXURE C: Recommended tools
Additional Information	

7. Pacer

Students need to communicate with the lecturer on the **progression** of their project and need to take initiative to invite their lecturing team, the WIL Coordinator and your Client to appropriate Sprint Events (Ceremonies).

It is recommended that you review ANNEXURE F: A quick start to WIL if you are unsure where to start.

The roadmap that you create will be the guideline that the lecturing team and WIL Coordinator will use to monitor and assess your progress through this module.

8. Detailed WIL Requirements

While there are general principles for the assessment of WIL modules, each WIL module will have its own marking criteria and weightings.

MILESTONE (ASSESSMENT POINT)	TASK	MARK WEIGHTING BREAKDOWN
1. POE		37.5 %
2. GitHub repository		21.875 %
3. Contribution	Individual contribution to the project	6.25 %
4. Presentation	Group Presentation	15.625 %
5. Attendance		6.25%
6. Peer-Evaluation		12.5%
TOTAL WEIGHTING		**100**

1 ANNEXURE A: Cognitive Overload

Environments to avoid are where you, as a developer, experience extended cognitive overload, this is usually where the business domain model has diverse software models that one must interpolate and understand before being able to work on a system. Software systems that have a high extraneous mental load (unnecessary mappings between business and software models) become difficult and then nearly impossible to maintain. This difficulty to maintain the software leads to lengthy maintenance cycles. The extra time needed for maintenance means that the software product will inevitably loose it's commercial advantage, and at first will begin lagging in features when compared to its competitors but will ultimately lead to a loss in support from its customers. The inflection point in the life cycle of a software product usually spells the beginning of the end. Decisions to reduce complexity and focus on a maintainable code base made early in the life cycle of a software product can steer the product and the team away from this disaster, and towards a sustainable software product.

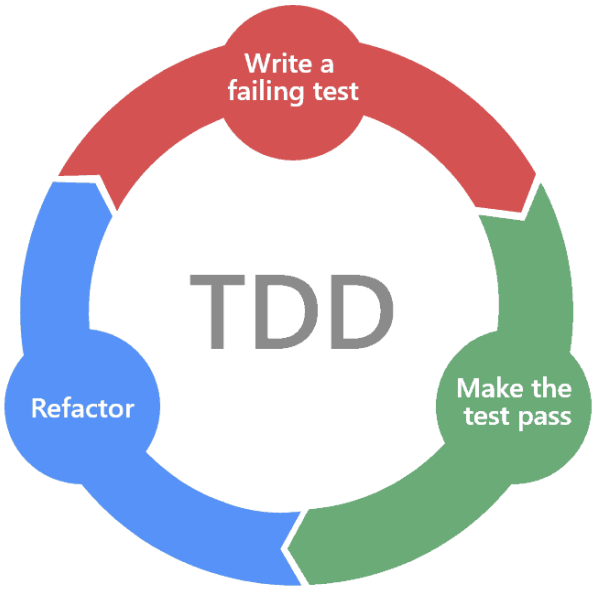
1.1 Domain Driven Design (DDD)

An excellent approach to managing cognitive load is to use Domain Driven Design (DDD) that aims to map the business domain model directly to the software model. Therefore, you as a developer, do not need intermediate interpolations, interpretations or mappings. DDD aims to give you a one-to-one mapping between the business understanding of a domain and the software services that you build in your application.

Further reading:
Eric Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software 1st Edition
https://en.wikipedia.org/wiki/Domain-driven_design [Accessed 08 February 2023].

1.2 Test Driven Development (TDD)

Another excellent tool for keeping a manageable level of cognitive load during development is to use Test Driven Development (TDD).
[1]



Using this approach, you write a failing unit test that describes a method you intend to build. This failing test is referred to as “red” stage, because when the unit test fails you see a red cross in your IDE. Then you write a method that implements the method to make the test pass. This passing test is referred to as a “green” stage because the test is now passing, and you see a green tick in your IDE. Next, you add complexity to your unit test (or another unit test for your user-story), typically by checking the next element of the user-story (or requirement), and this is the refactor stage. Refactoring the unit tests is expected to create a failing test, so back to the “red” phase of TDD. Next you need to implement the minimal additional code to ensure a passing test, to make the code green again.

This sequence of red->green->refactor->red->green->refactor->… is the development process for TDD. The major advantages of TDD are maintainable code because your suite of unit tests is actually your documentation of your code base, any new developer can quickly make changes with a minimal cognitive load; knowing that there is a safety net (of unit tests) to catch them if a mistake is made. There is a lot more to TDD, including a very useful naming conventions for tests that make each test “self-describing”, the name should contain:

- The name of the method being tested.
- The scenario under which it's being tested.
- The expected behaviour when the scenario is invoked.

[1] https://marsner.com/blog/why-test-driven-development-tdd/

For example, if you were unit testing a “login” method with incorrect credentials, you could use the following name: *loginWithIncorrectCredentialsExpectFailure*

You should also follow the AAA-pattern for each unit test that describes the structure of each unit test:

- Arrange: Prepare any prerequisites (service mocks, data dependencies, etc.)
- Act: Do the action that you want to test
- Assert: Was the action successful?

By way of example, consider writing a method that validates a password. The password needs to have a minimum length of twelve (12) characters and should include both upper and lowercase letters. There should also be at least one special character and one numeric character. The following can describe the process of using TDD to create a test suite for this method:

1. Write a test that expects minimum length of twelve (12) characters.
    a. The test case is that given a password of less than twelve characters expect failure.
    b. The test method name would then be: givenPasswordLessthanTwelve_expectFailure
2. Run the test and it should fail.
3. Create the method to check the password and add a validation on the length of the password. It must be twelve (12) characters or more.
4. Run the test and it should pass.
5. Can the code be refactored to be more efficient? If yes, then refactor the code. Make sure that the test still passes.
6. Write a second test that checks for at least one uppercase letter.
    a. The test case is given that there are no uppercase letters expect failure.
    b. The test method name would be givenPasswordHasNoUppercase_expectFailure
7. Run the test and it should fail.
8. Update the password validation method to check for at least one uppercase letter.
9. Run the test and it should pass.
10. Can the code be refactored to be more efficient? If yes, then refactor the code. Make sure that the tests still pass.
11. Write a third test method that checks for at least one lowercase letter.
    a. The test case is given that there are no lowercase letters expect failure.
    b. The test method name would be givenPasswordHasNoLowercase_expectFailure
12. Run the test and it should fail.
13. Update the password validation method to check for at least one lowercase letter.
14. Run the test and it should pass.
15. Can the code be refactored to be more efficient? If yes, then refactor the code. Make sure that the tests still pass.
16. Write a Fourth test method that checks for at least one special character.

   a. The test case is given that there are no special characters expect failure.
   b. The test method name would be givenPasswordHasNoSpecialCharacter_expectFailure
17. Run the test and it should fail.
18. Update the password validation method to check for at least one special character.
19. Run the test and it should pass.
20. Can the code be refactored to be more efficient? If yes, then refactor the code. Make sure that the tests still pass.

What other tests can you think of? Should the list of special characters be reduced to avoid SQL injection? You can easily extend the tests to include these types of requirements without breaking previous tests.

Admittedly this is a trivial example, however, it describes the process. TDD is about understanding the process of writing test case and then implementing the code so that the test case passes. Even with the most complex business case can be broken up into manageable test cases that can be implemented as unit tests in code.

Further reading on TDD:
https://en.wikipedia.org/wiki/Test-driven_development [Accessed 08 February 2023].
https://marsner.com/blog/why-test-driven-development-tdd/ [Accessed 08 February 2023].
http://www.tddbuddy.com/ [Accessed 08 February 2023].
https://www.codewars.com/ [Accessed 08 February 2023].
https://www.hackerrank.com/ [Accessed 08 February 2023].

It goes without saying that following good programming practices, like OOP, SOLID, and using design patterns all help in lowering cognitive load. You need to use all the tools in your tool-belt to create a maintainable application.

References worth reviewing:
A friendly introduction to cognitive load theory for the software developer
The theory behind cognitive load – Cognitive load during problem solving, J. Sweller 1988 [Accessed 08 February 2023].
Making better unit tests: part 1, the AAA unit test pattern, V. Khorikov 2020 (read the book if you can) [Accessed 08 February 2023].

2 ANNEXURE B: Declaration of Authenticity

Plagiarism occurs in a variety of forms. Ultimately though, it refers to the use of the words, ideas or images of another person without acknowledging the source using the required conventions. The IIE publishes a Quick Reference Guide (available on The IIE Library website) that provides more detailed guidance, but a brief description of plagiarism and referencing is included below for your reference. It is vital that you are familiar with this information and the Intellectual Integrity Policy before attempting any assignments.

The IIE respects the intellectual property of other people and requires its students to be familiar with the necessary referencing conventions. Please ensure that you seek assistance in this regard before submitting work if you are uncertain.

If you fail to acknowledge the work or ideas of others or do so inadequately this will be handled in terms of the Intellectual Integrity Policy (IIE023 – [available in the library]) and/or the Student Code of Conduct policy (IIE026)– depending on whether or not plagiarism and/or cheating (passing off the work of other people as your own by copying the work of other students or copying off the Internet or from another source) is suspected.

Your campus offers individual and group training on referencing conventions – please speak to your librarian or ADC/Campus Co-Navigator in this regard.

<u>Reiteration of the Declaration you have signed:</u>

1. I have been informed about the seriousness of acts of plagiarism.
2. I understand what plagiarism is.
3. I am aware that The Independent Institute of Education (IIE) has a policy regarding plagiarism and that it does not accept acts of plagiarism.
4. I am aware that the Intellectual Integrity Policy and the Student Code of Conduct prescribe the consequences of plagiarism.
5. I am aware that referencing guides are available in my student handbook or equivalent and in the library and that following them is a requirement for successful completion of my programme.
6. I am aware that should I require support or assistance in using referencing guides to avoid plagiarism I may speak to the lecturers, the librarian or the campus ADC/ Campus Co-Navigator.
7. I am aware of the consequences of plagiarism.

Please ask for assistance prior to submitting work if you are at all unsure.

Declaration of authenticity

I, _____ ID Number, _____

hereby declare that this portfolio, and any evidence included therein, contains my own independent work and that I have not received help from other groups.

I confirm that we have not committed plagiarism in the accomplishment of this work, nor have I falsified and/or invented experimental data.

I accept the academic penalties that may be imposed for violations of the above.

_____

_____

STUDENT SIGNATURE **DATE**

3 ANNEXURE C: Recommended tools

3.1 Diagramming tools

https://app.diagrams.net
Microsoft Visio

3.2 UML Diagrams

https://plantuml.com/
https://app.diagrams.net
Microsoft Visio

3.3 Scrum board

https://teamfu.tech
https://docs.github.com/en/issues

3.4 Retrospectives

https://reetro.io/

3.5 Source code

https://GitHub.com

3.6 Continuous Integration Pipeline

GitHub actions
https://docs.GitHub.com/en/actions

3.7 Security and static code analysis tools

Name/Link	Owner	License	Note
.NET Security Guard		Open Source or Free	.NET, C\#, VB.net
Agnitio		Open Source or Free	ASP, ASP.NET, C\#, Java, Javascript, Perl, PHP, Python, Ruby, VB.NET, XML
APIsecurity.io Security Audit		Open Source or Free	online tool for OpenAPI / Swagger file static security analysis
AppSweep	Guardsquare	Open Source or Free	Mobile application security testing tool for compiled Android apps with support of CI/CD integration
Automated Security Helper	AWS	Open Source or Free	ASH is a one stop shop for security scanners and does not require any installation. It will identify the different frameworks, and download the relevant, up to date tools. ASH is running on isolated Docker containers, keeping the user

Name/Link	Owner	License	Note
			environment clean, with a single aggregated report. The following frameworks are supported: Git, Python, Javascript, Cloudformation, Terraform and Jupyter.
Bandit		Open Source or Free	Bandit is a comprehensive source vulnerability scanner for Python
Betterscan CE (Community Edition)	Marcin Kozlowski	Open Source	Code Scanning/SAST/Static Analysis/Linting using many tools/Scanners with One Report. Currently supports: PHP, Java, Scala, Python, Ruby, Javascript, GO, Secret Scanning, Dependency Confusion, Trojan Source, Open Source and Proprietary Checks (total ca. 1000 checks). Supports also Differential analysis. Goal is to have one report using many tools/scanners
Brakeman		Open Source or Free	Brakeman is an open-source vulnerability scanner specifically designed for Ruby on Rails applications
clj-holmes	clj-holmes	Open Source	A CLI SAST (Static application security testing) tool which was built with the intent of finding vulnerable Clojure code via rules that use a simple pattern language.
Dawnscanner		Open Source or Free	Dawnscanner is an open-source security source code analyser for Ruby, supporting major MVC frameworks like Ruby on Rails, Padrino, and Sinatra. It also works on non-web applications written in Ruby.
Deep Dive		Open Source or Free	Byte code analysis tool for discovering vulnerabilities in Java deployments (EAR, WAR, JAR).
DevBug		Open Source or Free	PHP
Enlightn	Enlightn Software	Open Source	Enlightn is a vulnerability scanner specifically designed for Laravel PHP applications that combines SAST, DAST, IAST and configuration analysis techniques to detect vulnerabilities.

Name/Link	Owner	License	Note
Find Security Bugs		Open Source or Free	Java, Scala, Groovy
FindBugs		Open Source or Free	Find bugs (including a few security flaws) in Java programs [Legacy - NOT Maintained - Use SpotBugs (see other entry) instead]
FindSecBugs		Open Source or Free	A security specific plugin for SpotBugs that significantly improves SpotBugs's ability to find security vulnerabilities in Java programs. Works with the old FindBugs too.
Flawfinder		Open Source or Free	Scans C and C++.
Fluid Attack's Scanner	Fluid Attacks	Open Source	SAST, DAST and SCA vulnerability detection tool with perfect OWASP Benchmark score.
GitHub Advanced Security	GitHub	Open Source or Free	GitHub Advanced Security uses CodeQL for Static Code Analysis, and GitHub Secret Scanning for identifying tokens. GitHub code scanning can import SARIF from any other SAST tool
GolangCI-Lint		Open Source or Free	A Go Linters aggregator - One of the Linters is gosec (Go Security), which is off by default but can easily be enabled.
Google CodeSearchDiggity		Open Source or Free	Uses Google Code Search to identify vulnerabilities in open-source code projects hosted by Google Code, MS CodePlex, SourceForge, GitHub, and more. The tool comes with over 130 default searches that identify SQL injection, cross-site scripting (XSS), insecure remote and local file includes, hard-coded passwords, and much more. *Essentially, Google CodeSearchDiggity provides a source code security analysis of nearly every single open-source code project in existence – simultaneously. *
Graudit		Open Source or Free	Scans multiple languages for various security flaws. Basically, security enhanced code Grep.

Name/Link	Owner	License	Note
HCL AppScan CodeSweep - GitHub Action	HCL Software	Open Source or Free	Scan the new code on a push/pull request using a GitHub action. Findings are highlighted in the `Files Changed` view and details about the issue and mitigation steps can be found in the `Actions` page. Unrestricted usage allowed with a free trial account. The tool currently supports Python, Ruby, JS (Vue, React, Node, Angular, JQuery, etc), PHP, Perl, COBOL, APEX & a few more.
HCL AppScan CodeSweep - IDE	HCL Software	Open Source or Free	This is the first Community edition version of AppScan. It is delivered as a VS Code [https://hclsw.co/codesweep] and JetBrains [https://hclsw.co/codesweep-jetbrains] (IntelliJ IDEA, CLion, GoLand, PhpStorm, PyCharm, Rider, RubyMine, WebStorm) plugin and scans files upon saving them. The results show the location of a finding, type and remediation advice. The tool currently supports Java, .Net, Go, Python, Ruby, JS (Node, Angular, JQuery, etc), PHP, Perl, COBOL, APEX & a few more. Auto-fix for some of the issues is available with a free trial.
HCL AppScan on Cloud	HCL Software	Open Source or Free	Apex, ASP, C, C++, COBOL, ColdFusion, Go, Java, JavaScript (Client-side JavaScript, Kotlin, NodeJS, and AngularJS), .NET (C#, ASP.NET, VB.NET), .NET Core, Perl, PHP, PL/SQL, Python, Ruby, T-SQL, Swift, Visual Basic 6
Horusec		Open Source or Free	C#, Java, Kotlin, Python, Ruby, Golang, Terraform, Javascript, Typescript, Kubernetes, PHP, C, HTML, JSON, Dart, Elixir, Shell, Nginx, Swift

Name/Link	Owner	License	Note
HuskyCI		Open Source or Free	HuskyCI is an open-source tool that orchestrates security tests inside CI pipelines of multiple projects and centralizes all results into a database for further analysis and metrics. HuskyCI can perform static security analysis in Python (Bandit and Safety), Ruby (Brakeman), JavaScript (Npm Audit and Yarn Audit), Golang (Gosec), and Java (SpotBugs plus Find Sec Bugs)
Insider CLI	InsiderSec	Open Source or Free	An open-source Static Application Security Testing tool (SAST) written in GoLang for Java Maven and Android), Kotlin (Android), Swift (iOS), .NET Full Framework, C#, and Javascript (Node.js).
LGTM		Open Source or Free	A free for open-source static analysis service that automatically monitors commits to publicly accessible code in Bitbucket Cloud, GitHub, or GitLab. Supports C/C++, C\#, Go, Java, JavaScript/TypeScript, Python.
LunaTrace by LunaSec	LunaSec	Open Source or Free	Software Composition Analysis (SCA) tool to generate SBOMs, identify vulnerabilities in dependencies, and generate patches. Leverages Static Analysis to reduce false positives by filtering non-exploitable CVEs.
Microsoft FxCop		Open Source or Free	.NET
Microsoft PREFast		Open Source or Free	C, C++
MobSF		Open Source or Free	Mobile Security Framework (MobSF) is an automated, all-in-one mobile application (Android/iOS/Windows) pen-testing, malware analysis and security assessment framework capable of performing static and dynamic analysis.
MobSF		Open Source or Free	Android Java, Objective C, Swift

Name/Link	Owner	License	Note
nodejsscan		Open Source or Free	Node.js
OWASP ASST (Automated Software Security Toolkit)	Tarik Seyceri & OWASP	Open Source or Free	An Open Source, Source Code Scanning Tool, developed with JavaScript (Node.js framework), Scans for PHP & MySQL Security Vulnerabilities According to OWASP Top 10 and Some other OWASP's famous vulnerabilities, and it teaches developers of how to secure their codes after scan.
OWASP Code Crawler	OWASP	Open Source	.NET, Java
OWASP LAPSE Project	OWASP	Open Source	Java
OWASP Orizon Project	OWASP	Open Source	Java
OWASP WAP (Web Application Protection)	OWASP	Open Source	PHP
ParaSoft		Open Source or Free	C, C++, Java, .NET
phpcs-security-audit		Open Source or Free	A set of PHP_CodeSniffer rules to finds flaws or weaknesses related to security in PHP and its popular CMS or frameworks. It currently has core PHP rules as well as Drupal 7 specific rules.
PMD		Open Source or Free	PMD scans Java source code and looks for potential code problems (this is a code quality tool that does not focus on security issues).
PreFast	Microsoft	Open Source or Free	PREfast is a static analysis tool that identifies defects in C/C++ programs. Last update 2006.
Progpilot		Open Source or Free	Progpilot is a static analyzer tool for PHP that detects security vulnerabilities such as XSS and SQL Injection.
Psalm	Vimeo, Inc.	Open Source	Static code analysis for PHP projects, written in PHP.
Puma Scan Professional		Open Source or Free	.NET, C\#

Name/Link	Owner	License	Note
PVS-Studio		Open Source or Free	C, C++, C\#
Pyre		Open Source or Free	A performant type-checker for Python 3, that also has limited security/data flow analysis capabilities. [Accessed 08 February 2023].
Security Code Scan		Open Source or Free	Static code analyzer for .NET. It will find SQL injections, LDAP injections, XXE, cryptography weakness, XSS and more.
Semgrep		Open Source or Free	Lightweight static analysis for many languages. Find bug variants with patterns that look like source code. No compilation needed to scan source code. Supports Go, Java, JavaScript, JSON, Python, TypeScript, and more.
ShiftLeft Scan		Open Source or Free	A free open-source DevSecOps platform for detecting security issues in source code and dependencies. It supports a broad range of languages and CI/CD pipelines by bundling various open-source scanners into the pipeline.
Sink Tank		Open Source or Free	Java byte code static code analyzer for performing source/sink (taint) analysis.
Snyk Cloud	Snyk Limited	Commercial or Free	Detects cloud security issues as soon as developers start designing configurations, providing expert guidance to cloud, platform, and security teams in the tools and workflows they use every day.
Snyk Code	Snyk Limited	Commercial or Free	AI-powered code checker that analyzes your code for security issues, providing actionable advice directly from your IDE to help you fix vulnerabilities quickly
Snyk Container	Snyk Limited	Commercial or Free	Container and Kubernetes security that helps developers and DevOps find and fix vulnerabilities throughout the SDLC.
Snyk IaC	Snyk Limited	Commercial or Free	Reduce risk by automating Infrastructure as Code (IaC) security and compliance in development

Name/Link	Owner	License	Note
			workflows pre-deployment and detecting drifted and missing resources post-deployment.
Snyk Open Source	Snyk Limited	Commercial or Free	Software composition analysis (SCA) solution helping developers find, prioritize, and fix security vulnerabilities and license issues in open-source dependencies.
SonarCloud		Open Source or Free	ABAP, C, C++, Objective-C, COBOL, C\#, CSS, Flex, Go, HTML, Java, Javascript, Kotlin, PHP, PL/I, PL/SQL, Python, RPG, Ruby, Swift, T-SQL, TypeScript, VB6, VB, XML
SonarQube		Open Source or Free	Scans source code for 15 languages for Bugs, Vulnerabilities, and Code Smells. SonarQube IDE plugins for Eclipse, Visual Studio, and IntelliJ provided by [SonarLint](https://www.sonarlint.org/).
Spectral	SpectralOps	Open Source or Free	Discover, classify, and protect your codebases, logs, and other assets. Monitor and detect API keys, tokens, credentials, high-risk security misconfiguration and more.
Splint		Open Source or Free	C
SpotBugs		Open Source or Free	Java. This is the active fork replacement for FindBugs, which is not maintained anymore. Very little security. FindSecBugs plugin provides security rules.
Veracode		Open Source or Free	Android, ASP.NET, C\#, C, C++, Classic ASP, COBOL, ColdFusion/Java, Go, Groovy, iOS, Java, JavaScript, Perl, PhoneGap/Cordova, PHP, Python, React Native, RPG, Ruby on Rails, Scala, Titanium, TypeScript, VB.NET, Visual Basic 6, Xamarin
VisualCodeGrepper		Open Source or Free	C/C++, C\#, VB, PHP, Java, PL/SQL

Name/Link	Owner	License	Note
VisualCodeGrepper (VCG)		Open Source or Free	Scans C/C++, C\#, VB, PHP, Java, PL/SQL, and COBOL for security issues and for comments which may indicate defective code. The config files can be used to carry out additional checks for banned functions or functions which commonly cause security issues.
VS Code OpenAPI (Swagger) Editor extension		Open Source or Free	Plugin to Microsoft Visual Studio Code that enables rich editing capabilities for REST API contracts and also includes linting and Security Audit (static security analysis).

Reference: https://owasp.org/www-community/Source_Code_Analysis_Tools

3.8 Containerization

http://docker.io

3.9 Cloud hosting

https://azure.com

3.10 Documentation

Microsoft Word can be used and then a PDF generated when you are ready to submit. Alternatives include LaTeX tools, such as https://www.overleaf.com

3.11 User Experience Journey Mapping

https://miro.com/ag/ps/customer-journey-map/

4 ANNEXURE D: A Scrum Agile Primer

Scrum Agile is one of the most widely used approaches to developing commercial software today.

4.1 Some Success factors

There are several factors that have contributed to the adoption of this methodology, and here are some of the key factors:

- Collaboration with team and client is more important than following processes
- Responsive
- Continuous improvement
- Fail fast

4.1.1 Collaboration with team and client is more important than following processes

Working together with your team and your client in an open, honest way is more sustainable for a project and leads to more successful projects. When challenges are openly communicated with the team, it is more likely that a solution will be found. The team support each other when there is open and honest communication. Similarly, usually clients know when the facts are being hidden. Being honest with your client builds trust and becomes the foundation of a sustainable relationship. Being dishonest will quickly damage the relationship and will jeopardise the success of the project.

4.1.2 Responsive

The real-world evolves and changes continuously with time. In the same way, business requirements change to adapt to a changing environment. Iterations (or Sprints) allow clients to adapt their business priorities, add new requirements and remove requirements that are no longer needed.

Development is time-consuming and usually needs time to complete a user story. The balance between being responsive to business demand and the team needing enough time to implement a story is maintained by using Sprints. A Sprint is a fixed time period that is used for an iteration in the Scrum Agile model. An iteration can be any length of time but is commonly set to two (2) weeks. During a Sprint, the prioritization of backlog items and the items themselves are not allowed to change. Unless there is a critical change in the business and the client negotiates a change with the team. Such a negotiation means that if new items are added to the Sprint, then other items (which were originally planned for the sprint) must be removed.

This ability to respond to changing business requirements while maintaining stability for the Scrum team is sought after in many commercial environments.

4.1.3 Continuous improvement

Every iteration in the Scrum process includes the opportunity to reflect and improve. The improvements that the team agrees to implement should then be included in the next sprint. The team is looking continuously for ways to improve and work smarter.

4.1.4 Fail fast

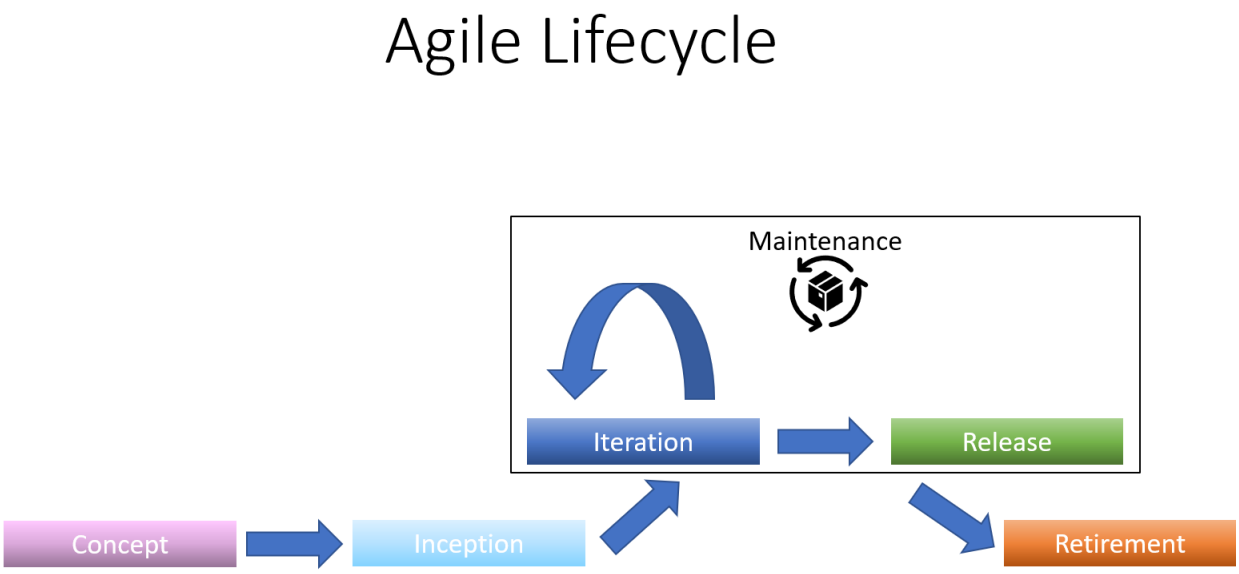
Not every technology is fit-for-purpose and not every idea is a good one. Spending too much time on a bad idea (wrong architecture) or the wrong technology can cause a project to fail.

The iterative nature of Scrum agile allows a review at the end of each iteration (Sprint) and this time it is the team's responsibility to be honest about these practices and ideas. If something is failing, then it should be stopped immediately. The team needs to adjust and find a new way forward.

The failing fast approach is the best way to test new technologies or new ideas before adoption. Test them and see if they can work before adopting for your final solution.

4.2 The Agile Lifecycle

The lifecycle in Agile refers to the software lifecycle and can be described by the following diagram:



4.2.1 Concept

The initial idea for the product is developed by the Product Owner (PO). The PO defines the core requirements for the product; considers product viability; competition in the market; and develops a business case for the product. Usually this includes phase includes gathering initial funding for the product.

4.2.2 Inception

The PO creates a team to work on the product. This initial work also includes refining the requirements into a Product Backlog.

4.2.3 Iteration

The iteration phase is where the construction of the product occurs. This is where the Scrum Agile iterations fit into the lifecycle (see the next section for details).

4.2.4 Release

Once the product has be quality assured (tested) and the PO is happy to release the features available in the product, then the team defines a product increment and releases the product.

4.2.5 Maintenance

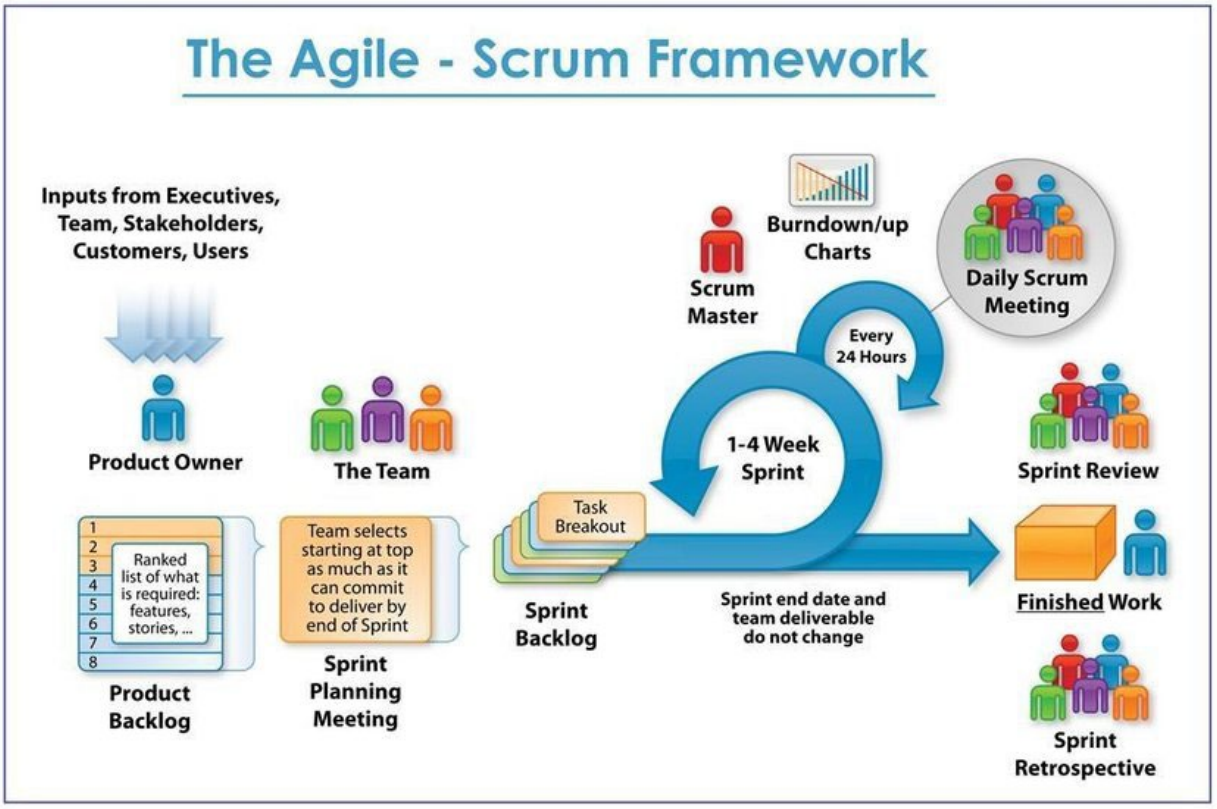
The product is available to customers and the team provide ongoing upgrades and customer support (bug fixing, and feature development). The maintenance period of the application includes regular iterations and releases.

4.2.6 Retirement

The product will eventually be retired (for many reasons) and this is the final phase of the software product. Typically, the customers are migrated to the replacement system and then the production system is shutdown. Finally, the code is archived.

4.3 The Scrum Agile Iterations

When your team enters the Iteration lifecycle phase, then the actual iterations (Sprints) are described in the following cycle:



[2]

4.3.1 Roles in the Scrum Framework

4.3.1.1 Product Owner

The Product Owner (PO) is responsible for the business value of the product being built. Typically, these are the people who have conceptualised a product, have sourced financial backing and have a passion to build the product.

The PO is responsible for defining the scope of the product and creating the Product Backlog. The Product Backlog is a ranked list of features that the PO needs in the product. The highest ranked feature is the most important.

4.3.1.2 Scrum Master

The Scrum Master is the co-ordinator of the Team, s/he is responsible for setting up all of the Scrum events (or ceremonies). The Scrum Master needs to live the Agile values and demonstrate to the team how to work with this methodology. The Scrum Master is also responsible for shielding the team from outside distractions (such as direct contact from Executives, Stakeholders, Customers, Users, etc).

[2] https://www.researchgate.net/figure/Agile-Scrum-Framework-20_fig6_340849598

4.3.1.3 The Team

These are the people responsible for building a high-quality software product. Usually these team members are cross-functional and able to work on more than one type of task.

4.3.2 Backlogs

There are different views of the items that the Team works on. Those that are future requirements are in the Product Backlog. Once an item has been committed to, it is moved from the Product Backlog into the Sprint Backlog. Once an item is in the Sprint Backlog it must be completed during that Sprint. Adding an item to a Sprint is a commitment to complete it within that sprint.

When a backlog item is too big to complete in a single Sprint, then it must be reduced in scope, so that it can be completed in a Sprint.

4.3.3 Events (Ceremonies)

It has been found that by using regular events and cycles, team develop a working rhythm that is referred to as the velocity of the team. This rhythm also helps the team develop a maintainable delivery rhythm. A maintainable cycle is the foundation of a healthy team with consistent high-quality delivery of working software.

4.3.3.1 Backlog Grooming

This event is where the team meets to discuss the Product Backlog. This ceremony is timeboxed, which means that there is a set amount of time allocated. When the time limit is reached, then the meeting ends. The goal is to work through as many backlog items as possible in the allocated time.

Typically, this meeting will be once a Sprint and would be 2 hours. Usually, there would be more time needed at the beginning of a project to groom the backlog items. As the project concludes, less time will be needed. Remember, that each team can adjust this to suit their requirements.

The prerequisite for this meeting is that the team have a Product Backlog and that the PO has ranked the items in the backlog. The most important Product Backlog Items (PBI’s) must be at the top of the list.

The purpose of this meeting is to discuss the first non-groomed item in the backlog and to identify what will be needed to fully implement the item. This is a team discussion, where each team member should ask questions to understand what needs to be done. Ideally, the PO is available to answer any questions about the PBI’s they have created.

Once each member of the team is confident that a PBI can be implemented, then the PBI should be marked as ready for implementation.

Note that the implementation time for a PBI should be considered. Ideally each PBI should not take more than two (2) days. Try to split the PBI's into parts if they are too big. Also try to focus on core functionality first. This means that sometimes you can implement the base of a feature before building on the "nice to have" part of the feature.

4.3.3.2 Sprint Planning

This is a Team meeting with the Scrum Master, the PO is not involved. The focus of this meeting is to commit to what is possible to be delivered in the Sprint.

The prerequisite for this meeting is that the Product Backlog has PBI's that have been groomed and are ready for implementation.

This meeting must occur before the start of the new Sprint. Before a Sprint can start, there must be PBI's added to the Sprint and the team must agree that the items in the Sprint Backlog can be accomplished during the Sprint.

The Scrum Master should first check each the availability of each team member and s/he must ensure that the team does not overcommit. However, the team also needs to complete the project, so it is important that there is a healthy balance when adding items into the sprint.

It is the responsibility of the team to create implementation tasks associated with each PBI, and these tasks form the Sprint Backlog for the upcoming Sprint.

The outcome of the Sprint Planning meeting is that each team member knows exactly what is expected of them in the upcoming sprint.

4.3.3.3 Daily Scrum Meeting

This is a short daily meeting where the team members gather to discuss what will be achieved in the coming day.

The team should raise with the Scrum Master if there are any issues that are blocking them from progressing. The Scrum Master will then take those issues offline with the person directly to understand the problem and to find ways to resolve the issue. The Scrum Master focus is usually on finding the right people to speak with, rather than trying to solve every problem themselves.

The typical format of this meeting is to briefly discuss what was accomplished yesterday, whether the person has any blockers; and what they hope to accomplish today.

Note that the frequency of the Scrum Meeting can be adjusted by the Team. For example, if the team are busy in an exam period and no-one has capacity to work on the WIL project, then a weekly Scrum meeting would make sense. Remember to adjust the meetings based on the unique requirements of your team.

4.3.3.4 Sprint Review

This is an opportunity to demonstrate your working software to your stakeholders. Ideally, the client (PO) must be present to review the software. With WIL, this is not always possible, so your WIL co-ordinator (and or lecturer) will act as a proxy product owner for your team.

This meeting occurs at the end of a Sprint and the purpose is to demonstrate the software that the team has developed.

There is no set amount of time for the meeting, discuss with your Scrum Master so that a meeting of appropriate time is scheduled with your PO (or proxy Product Owner).

If there is no working software to demonstrate at the end of a Sprint, then the Scrum Master must explain to the stakeholders why no. S/he must also negotiate with the stakeholders (PO and or proxy PO) regarding the next sprint's deliverables.

4.3.3.5 Sprint Retrospective

This meeting occurs at the end of a sprint and usually after the Sprint Review. Only the Team and the Scrum Master are included in this meeting and what happens in Retro, stays in Retro. This is a time for the team to honestly and openly review the Sprint that has just finished.

This meeting is typically timeboxed to 30min, but sometimes up to an hour. The meeting is chaired by the Scrum Master, but every team member is expected to contribute.

The structure of this meeting is to spend the first 5 minutes brainstorming on the following topics:

- What worked (what went well)?
- What did not work (what should we change)?
- Who did very good work (kudos)?

These items should be ranked by the team (remember that the meeting is timeboxed, so only the most important items will be discussed).

The remainder of the time should allow the team to talk through the items that have been raised.

Remember that this meeting is your chance to improve things in the next Sprint, so try to focus on what practical changes can be made.

The outcome of the meeting is a list of things that will be done differently in the next Sprint. Try to keep your list less than five (5) items, anything more will probably be too

much change for a single Sprint. Typically, teams find between one (1) and three (3) changes per Sprint to be manageable.

4.3.3.6 Sprint

A Sprint is a complete iteration that involves all the above events (ceremonies). The purpose of a Sprint is to achieve the goal described by the team at the beginning of the Sprint. The team works together to achieve the goal.

The Sprint is timeboxed into a few weeks that suits the product being developed and the technologies being used. The Scrum Agile guide recommends that a Sprint is not shorter than one (1) week and not longer than four (4) weeks. The industry norm is to use a Sprint of two (2) weeks, but it is the decision of the Scrum team to set the duration of a Sprint.

Our recommendation is that you consider the time your team need to produce demonstratable output to your client. We further recommend that you keep the duration of your Sprints consistent throughout this project, so that you can develop Velocity charts for your PoE. If you decide to change the duration of your Sprints, then your Velocity chart will be reset.

4.4 Scrum Charts

Charts are used to measure the progress and productivity of the team. These are a guide for the Scrum Master to understand when the team is not keeping pace with the commitment for the Sprint or when the team completes things faster than planned.

The charts that you should be aware of (remember this is just an introduction) are the Burndown chart, the Burnup chart and the Velocity chart.

4.4.1 Burndown chart

The burndown chart starts with the total amount of work planned for the sprint and reduces it each time a task is closed.

The goal is to reduce the work to zero before the end of the sprint.

4.4.2 Burnup chart

This is the inverse of the Burndown chart. This chart starts with nothing that has been completed (zero) and then increases the amount of work every time a task is closed. The goal is that the chart increases consistently over the course of the Sprint until the total amount of work planned is completed at the end of the sprint.

4.4.3 Velocity chart

This chart only starts to make sense after about three (3) sprints, and the objective is to measure the amount of work completed in each sprint. The duration of a Sprint must be consistent for this chart to make sense.

The horizontal axis plots the sprint, and the vertical axis plots the amount of work completed. The goal of this chart is to develop a maintainable rhythm for the team. The question that this chart should answer is “How much work can the team complete in a sprint?”

4.5 Further Reading

The art of software development is evolving all the time and this Scrum Agile Primer is only meant as a brief introduction. There is a lot more to learn and the reader is encouraged to continuously keep their knowledge up to date.

Here are a few links to get started:
Official Agile Manifesto
An Agile Guide
Scrum Guide
Scrum Resources
PMI Agile Practice Guide
Scrum Alliance Resources

5 ANNEXURE E: Sprint Attendance Record

SPRINT ATTENDANCE RECORD

GROUP NAME: _____

SPRINT: _____

Attendees	Backlog Grooming	Sprint Planning	Sprint Review	Sprint Retrospective	Daily Scrums
Client		n/a		n/a	n/a
WIL Coordinator					
Lecturer					

6 ANNEXURE F: A quick start to WIL and Dates

6.1 Choose your team

Friends do not always make the best work colleagues. Choose wisely. This module can only be successfully complete if every team member contributes and works towards a common goal.

Consider the agile principles when selecting your team:

- Ability to understand a problem
- Willingness to collaborate
- Ability to learn
- Willingness to stay flexible
- Focus on achieving high-performance results

A team works best when the people in the team can work hard together, be honest about the quality of the work.

Scrum teams should be cross functional, meaning that each team member can contribute to every part of the project. Practically, it makes sense to assign responsibilities in your team as early as possible. While everyone can contribute to every part of the project, there will be people better suited to gathering requirements; or developing software; or testing.

Be honest about what your commitments are for the semester and the level that you can realistically contribute to this project.

This should lead to your Work Agreement for your team.

6.1.1 Work Agreement

This is a list of principles that you as a team agree to follow. Spend between 30min to an hour as a team brainstorming what is important to you as a team and document what you agree.

Here is an example:
- We will meet every other day at 8am for 15min for a quick scrum.
- We will identify problems, dependencies early and communicate with the team.
- We will review the output of every sprint as if the client is participating (even if they are not)
- We will be honest but not aggressive in our feedback on other people's work.
- We will always help each other.

Remember that your Sprint Retrospective meeting is an opportunity to be honest with your teammates if someone is not following the Work Agreement.

6.1.2 Team Roles

Each person should ideally be cross-functional, contributing equally to every part of the project. In practice, this seldom happens, and it is more pragmatic to assign each team member a core area of responsibility and then to allow that team member to contribute to other areas once their primary responsibility is completed. The goal is to allow each team member to focus on a core area of responsibility and if blocked, then have another area that will allow each person to continue to contribute throughout the project.

Here is an example of how you could split your team roles:
1. Requirements and testing
2. Architecture and design
3. Development frontend lead
4. Development backend lead

There is not enough time in your semester to assign all responsibilities to a single person. Distribute the responsibilities evenly between your team so that everyone has an equal opportunity to contribute. Allow each team member to focus on their individual strength and when reviewing the overall project, each member should have a meaningful contribution to the success of the project.

6.2 Meet your client

Meet your client and understand their needs. Gathering requirements is important because you need to shape your product based on their needs. The client's needs are not always clearly communicated, and it is important to ask questions. Research their environment and understand as much as you can about the organisation before meeting your client. Preparation shows that you are serious about your work and your client; this demonstrates a level of professionalism that your client will appreciate.

Your time with your client is limited, so gather as much information as you can. Preparation is crucial, so research (Google) and find out as much as you can about the organisation before the meeting as you can. Use your research to prepare questions to ask the customer.

Also, remember to ask about non-functional requirements.

6.3 Plan your time

Create a roadmap for this module. In Scrum Agile terms, you can think of the key goals through the semester as being milestones.

Your team will not be successful with this module if you do not plan your time for this semester.

This module recommends Scrum Agile and to adhere to the Agile principles.
Plan how many sprints you want to use for each part of the project, how will you spend your time? Think high-level, what should the goal of these sprints be? At this point, you do not need

detailed tasks only the purpose of the sprint. Before you start a sprint, make sure that you have the detailed tasks.

Ideally you want to present at least a mock-up of the solution to the client for feedback as soon as you can. The more feedback you can get from the client the better your solution will align with the client needs.

This module is approximately 12 weeks, you need to split this time into sprints. A typical sprint is 2 weeks, but you can adjust this to suit your individual team. If you choose to work in 2-week sprints, then you have 6 sprints to complete this module.

During each sprint you should perform the Sprint events (ceremonies):

- Scrum (usually daily for 15min or less)
- Backlog prioritization (typically once a sprint for 1 hour, but usually more time is needed initially) The timing can be adjusted based on the duration of your sprint and your team's availability.
- Backlog grooming (typically 2 hours every two weeks, but often more frequently at the beginning of the sprint). The timing can be adjusted based on the duration of your sprint and your team's availability.
- Sprint Review (typically an hour at the end of the sprint). This is where you demonstrate working software to your client. If the client is not available, then your WIL Co-ordinator and Lecturers will fulfil this role.
- Sprint Retrospective (typically an hour after the Sprint Review). The timing can be adjusted based on the duration of your sprint and your team's availability.
- Sprint Planning (typically an hour once a sprint)

A sprint goal defines the high-level objective for the sprint. The actual tasks in the sprint can include any tasks that the team prioritise. It is important that the entire team agree the sprints and the tasks that are included.

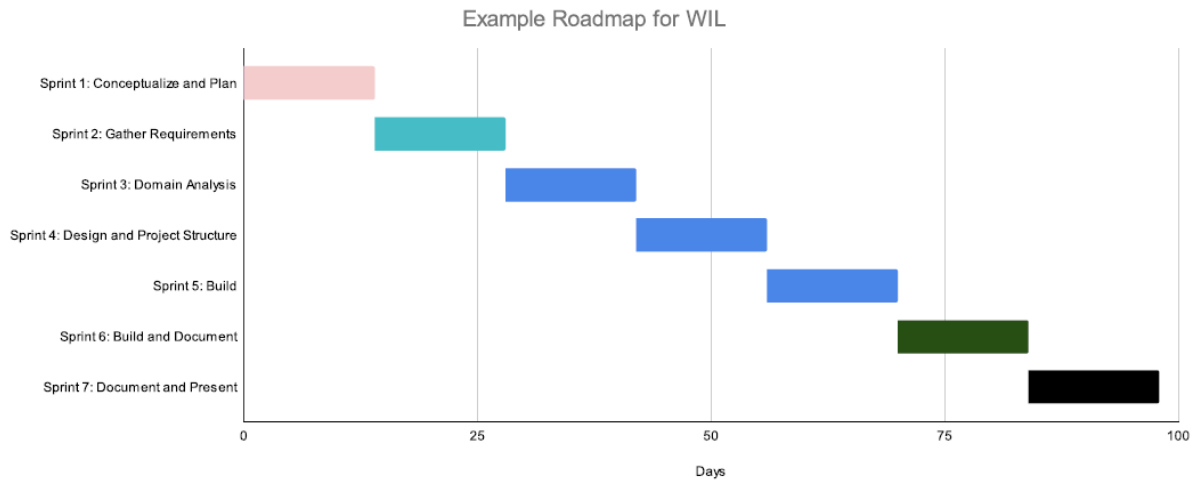
Here are a few things to keep in mind when creating your high-level plan:
Do your team members have commitments to other modules? Who is available in which sprint?

- What are your team commitments to assignments? When will they be unavailable for this project?
- When are your semester exams? Do any of these commitments clash with your planning. Who will be available?
- Have you planned time for the Scrum ceremonies? Who will schedule the meetings in your diaries?
- Have you considered contingencies? What if your lead developer contracts Covid? Identify risks and consider your approach to mitigating these risks as a team. A great way to mitigate this risk is to ensure pull request (PR) reviews by your team; so that no code is committed without at least one other person understanding the code.

The roadmap (high-level plan) needs to be something that all team members agree on and commit to. This is your plan, own it.

6.3.1 Example of a Roadmap (high-level plan)

Below is an example of a Roadmap for WIL. Note that the first Sprint is in the Concept Agile Lifecycle phase; the second Sprint is the Inception Agile Lifecycle phase; the third, fourth and fifth Sprints are in the Iteration Agile Lifecycle phase; then the sixth Sprint is in the Release Agile Lifecycle phase; then the last sprint is in the Maintain Agile Lifecycle phase. Refer to ANNEXURE D: A Scrum Agile Primer if you are unfamiliar with the lifecycle phases.



Here is an idea about how you might develop a high-level plan for this module:
Sprint 1: Conceptualize and Plan
Agile Lifecycle Phase: Concept
Sprint Goal: Create a Sprint team and create your Work Agreement, Definition of Ready (DoR), Definition of Done (DoD) and define your team roles. Also create a Roadmap (high-level plan)

Sprint 2: Gather requirements
Agile Lifecycle Phase: Inception
Sprint Goal: Meet with client and develop user stories, as well as a first draft user journey. (Remember the non-functional requirements.)
Considerations: Are any technology investigations needed?
Team availability: [List your team members and their availability for this sprint]

Example backlog items:
1. As a developer I need a prioritised backlog of requirements from the client so that I can develop an application that meets their needs. (Requirements responsibility)
2. As a developer, I need to understand how the screens and user roles will work together, so that I can build a UI that meets the client's needs. (Create first user journey – UX responsibility)
3. As a frontend developer I need to know which framework to use and which standards to follow so that I can develop high quality code. (Frontend responsibility)

4. As a DevOps team member, I need to understand how GitHub actions work so that I can build a pipeline to ensure a high-level of quality of the code that is committed. (DevOps responsibility)
5. As a backend developer I need to know which libraries (SDK) versions to use for backend development so that I can ensure my local environment is configured to match the rest of my team. (Backend responsibility)

Sprint 3: Domain analysis
Agile Lifecycle Phase: Iteration
Sprint Goal: Do the domain analysis. Complete the User Journey diagram. Agree the architecture and technology choices and document. Build mock-up and present to the client.
Team availability: [List your team members and their availability for this sprint]

Sprint 4: Design and project structure
Agile Lifecycle Phase: Iteration
Sprint Goal: Review any feedback from the client and update. Design core component interactions and document. Setup your GitHub repo. Design your pipeline and implement the initial version.
Team availability: [List your team members and their availability for this sprint]

Sprint 5: Build
Agile Lifecycle Phase: Iteration
Sprint Goal: Build the solution. Implement your DevSecOps pipeline. By this time the team members should all work in parallel using agreed design. Collaborate to ensure integrations between components are successful.
Team availability: [List your team members and their availability for this sprint]

Sprint 6: Build and Document
Agile Lifecycle Phase: Release
Sprint Goal: Complete the build and ensure that the components of the application work together. Continue working on your POE.
Team availability: [List your team members and their availability for this sprint]

Sprint 7: Presentation and Documentation
Agile Lifecycle Phase: Maintain
Sprint goal: Ensure that the application is stable for the presentation and give an outstanding presentation to the client. Complete the documentation of your project and finalize the POE.
Team availability: [List your team members and their availability for this sprint. Note that it important that everyone in the team is present at the final presentation.]

6.3.2 Example of time commitment expectations

The following is an example of the time commitment expected for a two-week Sprint, which consists of ten (10) working days. This assumes a light workload of two hours a day on WIL for each day in the Sprint. Only two hours are allocated because we assume the team has other study commitments. Adjust the task work time based on your team's availability, consider time commitments to other modules, exams, projects, etc.

Topic	Time commitment
Daily Scrum	10x15 = 150min
Backlog Grooming	120min
Sprint Planning	60min
Sprint Review	60min
Sprint Retrospective	30min
Task work	10x120 =1200min
TOTAL	27 hours

Over the course of seven (7) Sprints this would amount to 189hours time commitment to complete this module.

6.4 Create your backlog

Your product backlog contains the list of all User Stories that you plan to complete for your project. It is a list of all your requirements for the project. This list can grow (and shrink) as your project runs.

Create a backlog of tasks as early as possible. Assign tasks and ensure that everyone knows what they are responsible for.

We recommend using an online collaboration tool for this. Such as https://teamfu.tech to track User Stories and tasks.

6.4.1 Creating backlog stories

We recommend using the following template for creating stories for your team backlog:
As a (who wants to accomplish something)
I want to (what they want to accomplish)
So that (why they want to accomplish that thing)

This format helps everyone understand who wants the feature, why they want the feature and what they want to do. This also helps understand user journeys through the software and roles and rights that are needed per type of user.

6.4.2 Prioritize your backlog

Which User Stories in the backlog are the most important to the client? These should be your top priority. Rank the backlog from the most important to the least important.

Agile principles aim to deliver the most value to the client as early as possible. This means working software as early as possible.

Practically, you can nominate a team member to be a proxy-product owner. This person acts on behalf of your client and is given the responsibility to prioritize the backlog.

6.4.3 Backlog grooming

As a team you need to block a portion of time every sprint to discuss the top priority items for the next sprint.

Typically, in a two-week sprint cycle, you will probably want to meet at least once for two hours to discuss the backlog. Adjust this based on the length of your sprint. If you have a shorter cycle, then make your meetings shorter, or you won't have any time to complete the User Stories in your sprint.

The purpose of backlog grooming is to discuss the highest priority User Story that has not been groomed. Discussion points should include dependencies, implementation, testing and agreeing how you know it is complete. Do not leave a User Story open-ended. It needs to be clear to all team members what needs to be done, what the User Story is dependent on and how will it be tested.

As a team you should agree what your definition of ready is (DoR). This is your team's definition of when a User Story is ready for Sprint Planning. If a User Story has not been groomed, then it will not meet the DoR and cannot be added to a sprint.

You should also agree on a definition of done (DoD). This is your team's definition of when a User Story is successfully implemented. For example, the User Story is only done when the code builds successfully in the GitHub actions pipeline and is deployed to a shared environment; and another team member has tested the feature and confirms that it is fully implemented.

6.4.4 Sprint planning

Sprint planning must consider the team's time commitments. Remember that when you as a team agree to complete a user story it is a commitment. This means that the team does everything possible to achieve the commitment.

Only User Stories that have been groomed and meet the DoR can be considered for the Sprint. Start with the highest ranked User Story on the backlog and add to the sprint. Stop adding User Stories to the sprint when the team is fully committed for their available time.

Use your scrum board to track the User Stories and responsibilities. E.g., https://teamfu.tech

6.4.5 Scrum Event (Ceremony) Attendance

Use ANNEXURE E: Sprint Attendance Record to record the people who attended the events (ceremonies) in each Sprint. Add your team's names below the "Lecturer" entry.

Note the following:

- The Client needs to attend at least one Sprint Review. They do not need to attend every scrum meeting. Coordinate with your client about when they will be able to attend Sprint Reviews. Remember that there are other WIL teams that also need time from your client, so you may need to co-ordinate these meetings with your Lecturer and WIL Coordinator
- Your WIL Coordinator will attend as many Scrum Events (Ceremonies) as possible. However, you need to invite and coordinate with the WIL Coordinator as to what is practically possible.
- Your Lecturer will attend as many Scrum Events (Ceremonies) as possible. However, you need to invite and coordinate with your lecturing team as to what is practically possible.
- The Scrum Team is expected to attend all ceremonies.

For the Daily Scrum entries use a fraction, for example if you have ten (10) Scrums in a two-week Sprint cycle and you attend all ten (10) then use $^{10}/_{10}$

However, if a team member only attends three (3) Daily Scrums, then use $^{3}/_{10}$

6.5 Timebox

Timeboxing is restricting your meeting, work, investigation, or task to a specified time period. This is a good principle to help you and your team with time management. It is important to use your time productively or you will not complete the project.

For example, when busy with backlog grooming, you can timebox the meeting to two hours. During those two hours complete as many stories as possible. Complete the meeting promptly after two hours. In the next meeting you can continue the backlog grooming where your team finished.

Timeboxing can also help your team to deliver high value features to the customer early by only delivering the simplest implementation of the feature first. Subsequent sprints allow you to deepen your feature until it is completed.

6.6 Regularly review, reflect and adjust

Agile has the approach that teamwork is work in progress and leverages continuous improvement to keep bettering the team. Honest communication is important during Sprint retrospectives. If a team member did not deliver on work, they committed to do, then it is important to discuss the reasons why as a team.

Remember that you are working together to build a software solution for your client.

If someone was sick and was not able to deliver, then how can you as a team adjust and catch-up in the next sprint?

Did someone over-commit in the sprint? Help them by not assigning so much work in the next sprint and make sure that everyone in the team is contributing.

You goal is to adjust your next sprint so that you find a sustainable level of work for your team. Also, if you fall behind with your planning, then you will need to catch-up.

Catching up on missed work usually means that you will need to adjust the next sprint, and this will ripple through your project. Your team will probably find that they need to reduce the scope of the original work to a more manageable level.

On the other hand, if your team accomplishes more than expected then you can develop additional features.

Retrospectives (or Retro's) are also used to look at any changes that are needed. For example, is your selected technology stack working?

Everything that forms part of your Sprint cycle can be improved, including tools, processes, technologies, architecture, design patterns, etc.

6.7 Demonstrate working software often

Working software is the best measure of progress. Demonstrate your software as often as you can. Even if the client is not available, demonstrate your software to your colleagues, your family, anyone who will look and give you honest feedback.

6.8 GitHub

Your team needs a GitHub repository that all team members use. Create this as soon as possible and give access to all team members. The simplest approach is that one person in the team creates a repo and then gives contributor access to the rest of the team.

If you are not familiar with GitHub, then there is a useful introduction available here: https://github.com/Cyber-Mint/git-going

6.8.1 GitHub Actions

Your team will need a DevOps pipeline (GitHub Actions) to allow you to run your unit tests and build your project every time code is committed.

Once you have built a basic DevOps pipeline that runs the projects unit tests and builds the code consider the following enhancements:

- Add a static code analyser (see section below)
- Add a source code security analyser tool (see section below)
- Deploy your services into your cloud infrastructure. Note that if you do want to perform this step, then using containerisation (Docker) is usually a prerequisite.
- Publish your Android app to the Google Play Store

Further reading:
https://docs.github.com/en/actions/quickstart
https://docs.github.com/en/actions/automating-builds-and-tests
https://linuxhit.com/how-to-create-docker-images-with-github-actions/
https://learn.microsoft.com/en-us/azure/app-service/deploy-github-actions?tabs=applevel

6.9 Static Code Analysis and Security Scanners

A static code analysis tool will look for bugs, security flaws, code maintainability and similar code quality issues. Typically, these tools give you a dashboard page showing you the overall quality of your code. Such as where you need to increase unit testing coverage, the maintainability of your code and several other metrics that will help improve the quality of the code. The purpose is to scan your code and then remediate the findings so that you improve the score on your dashboard.

There is a list of code scanners available in ANNEXURE C: Recommended tools. Best practice is to integrate a static code analysis tool and a source code security analyser tool into your GitHub Actions pipeline so that every time you commit and push code the scanners can check the code quality and whether there are any security vulnerabilities. Ideally, the person committing the code will review these findings and fix them before merging their code into the main branch.

A source code security analyser tool will specifically focus on scanning your code for security vulnerabilities. These tools offer more comprehensive security scanning than a static code analysis tool. For any commercial project it is recommended to include a security scan of your source code. The purpose of this scan is to identify security concerns and to allow you to remediate the findings from the scan so that your security quality is improved.

Note that the source code security analyser tools are very language specific, so you may need to implement more than one depending on how many languages you are using.

The purpose of implementing these code scanners is to improve maintainability of your code, reduce bugs and improve security.

Dates for the module

Here's a breakdown of the Work Integrated Learning (WIL) project milestones, for a start date of 1st April and a completion date of 1st November, emphasising group work and role allocation. This timeline covers approximately seven months. The coordinator will confirm the dates with you, but these serve as a guide for them.

Milestones and deliverables

Overall Strategy

- Employ Scrum Agile methodology with two-week sprints.
- Ensure constant communication and collaboration among team members.
- Aim for continuous improvement and flexibility throughout the project.
- Maintain regular demonstrations of working software to the client.

Project Timeline and Milestones

Phase 1: April (Sprints 1 & 2)

- **Sprint 1: Team Formation and Initial Planning (1st – 14th April)**
  - **Goal**: Form the team, define team roles, establish communication protocols, and create initial project documentation.
  - **Tasks**:
    - Team selection and agreement on working principles.
    - Defining team roles (e.g., requirements and testing, architecture and design, front-end lead, back-end lead).
    - Setting up communication tools (Microsoft Teams, Slack, etc.).
    - Creating a project roadmap and initial sprint plan.
    - Establish "Definition of Ready" (DoR) and "Definition of Done" (DoD).
  - **Emphasis**: Importance of selecting team members with diverse skills and willingness to collaborate.
- **Sprint 2: Requirements Gathering (15th – 28th April)**
  - **Goal**: Meet with the client to gather and document requirements and create initial user stories, identify use cases and functional requirements.
  - **Tasks:**
    - Initial client meeting to understand their needs and expectations.
    - Researching the client's organisation and the problem they are trying to solve.
    - Creating initial user stories based on client requirements.
    - Developing initial use cases and functional requirements.
    - Documenting non-functional requirements.
  - **Emphasis**: Importance of preparation for client meetings and asking relevant questions.
  - **Responsibility**: The team member in charge of Requirements and Testing should take the lead in this Sprint.

Phase 2: May (Sprints 3 & 4)

- **Sprint 3: Domain Analysis and Design (29th April – 12th May)**
  - **Goal**: Analyze the gathered requirements to identify classes/objects, create a domain model, complete the functional requirements, and design the system architecture.

    - **Tasks**:
        - Conducting domain modelling using Domain Driven Design (DDD).
        - Completing the functional requirements.
        - Designing the system architecture and technology stack.
        - Presenting the mock-up to the client for feedback.
    - **Emphasis**: Importance of mapping the business domain model directly to the software model to manage complexity.
    - **Responsibility**: The team member in charge of Architecture and Design should take the lead in this Sprint.
- **Sprint 4: Project Structure and Initial Implementation (13th – 26th May)**
    - **Goal**: Set up the project infrastructure, including the GitHub repository and CI/CD pipeline, and begin initial implementation of core components.
    - **Tasks**:
        - Setting up the GitHub repository and granting access to team members.
        - Designing and implementing the initial version of the CI/CD pipeline using GitHub Actions.
        - Implementing core components of the system.
    - **Emphasis**: Importance of establishing a robust development environment early in the project.
    - **Responsibility**: The team member in charge of Architecture and Design **should** take the lead in setting up the infrastructure.

Phase 3: June - July (Sprints 5, 6 & 7)

- **Sprint 5: Core Feature Implementation (27th May – 9th June)**
    - **Goal**: Focus on implementing the core features of the application, ensuring that they meet the defined requirements and design specifications.
    - **Tasks**:
        - Implementing user stories/functional requirements related to core features.
        - Writing unit tests to ensure code quality and functionality.
        - Integrating implemented components and conducting integration testing.
    - **Emphasis**: The front-end and back-end developers work in collaboration during the sprint.
    - **Responsibility**: The front-end and back-end leads should take the lead, respectively.
- **Sprint 6: Integration and Testing (10th - 23rd June)**
    - **Goal**: Focus on integrating individual components into a cohesive system, conducting thorough testing, and addressing identified bugs.
    - **Tasks**:
        - Integrating front-end and back-end components.
        - Conducting various types of testing, such as unit, integration, and regression testing.
        - Documenting the testing process and results.
        - Addressing bugs and issues identified during testing.
    - **Emphasis**: Importance of continuous integration and testing to ensure a high-quality product.
    - **Responsibility**: The team member in charge of Requirements and Testing should take the lead in this Sprint.

- **Sprint 7: Secondary Feature Implementation (24th June - 7th July)**
  - **Goal**: Focus on implementing the secondary features of the application, ensuring that they meet the defined requirements and design specifications.
  - **Tasks**:
    - Implementing user stories related to secondary features.
    - Writing unit tests to ensure code quality and functionality.
    - Integrating implemented components and conducting integration testing.
  - **Emphasis**: The front-end and back-end developers work in collaboration during the sprint.
  - **Responsibility**: The front-end and back-end leads should take the lead, respectively.

Phase 4: August - September (Sprints 8, 9 & 10)

- **Sprint 8: Security and DevOps (8th - 21st July)**
  - **Goal**: Implement security measures, enhance the DevOps pipeline, and prepare the application for deployment.
  - **Tasks**:
    - Implementing security best practices throughout the application.
    - Enhancing the DevOps pipeline with static code analysis and security scanning tools.
    - Configuring the application for deployment to the cloud.
  - **Emphasis**: Ensuring the application is secure and can be deployed efficiently.
  - **Responsibility**: The team member in charge of Architecture and Design should take the lead.
- **Sprint 9: Performance and Scalability Enhancements (22nd July - 4th August)**
  - **Goal**: Optimise the application for performance and scalability, ensuring it can handle the expected user load.
  - **Tasks**:
    - Conducting performance testing and identifying bottlenecks.
    - Optimizing code and database queries for performance.
    - Implementing caching strategies to improve response times.
    - Configuring auto-scaling to handle increased traffic.
  - **Emphasis**: Focus on non-functional requirements to ensure a smooth user experience.
  - **Responsibility**: The team member in charge of Architecture and Design should take the lead.
- **Sprint 10: User Acceptance Testing and Bug Fixing (5th - 18th August)**
  - **Goal**: Conduct user acceptance testing with the client, gather feedback, and fix any remaining bugs.
  - **Tasks**:
    - Conducting user acceptance testing with the client or a proxy.
    - Gathering feedback and documenting issues.
    - Fixing bugs and making necessary adjustments.
    - Preparing the application for final deployment.
  - **Emphasis**: Ensuring the application meets the client's expectations and is ready for handover.
  - **Responsibility**: The team member in charge of Requirements and Testing should take the lead in coordinating with the client.

Phase 5: September - October (Sprints 11, 12 & 13)

- **Sprint 11: Documentation and Training (19th August - 1st September)**
  - **Goal**: Finalise all project documentation and create training materials for the client.
  - **Tasks**:
    - Completing the Portfolio of Evidence (PoE).
    - Creating user manuals and training materials.
    - Documenting the system architecture and design.
  - **Emphasis**: Ensuring the client has all the necessary information to maintain and use the application.
  - **All Team Members share responsibility** to share what they've worked on and to make the documentation as coherent as possible.
- **Sprint 12: Final Presentation Preparation (2nd - 15th September)**
  - **Goal**: Prepare for the final project presentation to the client and stakeholders.
  - **Tasks**:
    - Creating a compelling presentation that showcases the project's key features and benefits.
    - Practicing the presentation to ensure a smooth and engaging delivery.
    - Preparing for potential questions from the audience.
  - **Emphasis**: Delivering a professional and informative presentation.
  - **All Team Members share responsibility** and will be graded on their presentation skills.
- **Sprint 13: Contingency and Refinement (16th - 29th September)**
  - **Goal**: Address any remaining issues, refine the application based on final feedback, and prepare for handover.
  - **Tasks**:
    - Addressing any bugs or issues identified during the final review.
    - Making final refinements to the application.
    - Ensuring all documentation is complete and accurate.
  - **Emphasis**: Delivering a polished and well-documented application to the client.
  - **All Team Members share responsibility**. Use ANNEXURE E: Sprint Attendance Record to record the people who attended the events (ceremonies) in each Sprint.

Phase 6: October (Sprint 14)

- **Sprint 14: Handover and Support Transition (30th September - 13th October)**
  - **Goal**: Hand over the completed project to the client and ensure a smooth transition for ongoing support.
  - **Tasks**:
    - Delivering the final application and documentation to the client.
    - Providing training to the client on how to use and maintain the application.
    - Establishing a process for ongoing support and maintenance.
  - **Emphasis**: Ensuring the client is self-sufficient and can effectively use the application.
  - **All Team Members share responsibility** to train the client and ensure the client understands the application.

Final Handover and presentation (1st week of November)

- **Goal**: Officially hand over the completed project and all related documentation to the client and present your final project to the Coordinator.

Importance of Group Work

- **Collaboration**: Regular communication and collaboration are essential for success.
- **Shared Responsibility**: Distribute responsibilities evenly and allow each team member to focus on their strengths.
- **Conflict Resolution**: Establish a process for resolving conflicts and addressing issues constructively.
- **Peer Review**: Implement code reviews to ensure code quality and knowledge sharing.

Note: The lecturer, WIL coordinator, and client must be invited to attend appropriate Sprint Events (Ceremonies).

Further reading:
https://owasp.org/www-community/controls/Static_Code_Analysis
https://www.nist.gov/itl/ssd/software-quality-group/source-code-security-analyzers
https://owasp.org/www-community/Source_Code_Analysis_Tools
https://github.com/analysis-tools-dev/static-analysis
https://github.com/marketplace/actions/sonarqube-scan
https://github.com/marketplace/actions/securitycodescan

7 ANNEXURE G: PoE Rubric

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
Business Communication (hint APCT5111, IPMA6212, INRS7321)					
Demonstrate effective small group (team) communication team demonstrates ability to: • Deal with conflict • Communicate about challenges • Allocation of work • Meet regularly (even with other modules and being under exam pressure)	• No evidence of regular communication. • No documentation of retrospectives. • No evidence of regularly meeting. • No evidence of distributing work evenly. • No evidence of conflict resolution in the team. • No evidence of team collaboration to solve problems.	• Limited evidence of regular communication. • Limited documentation of retrospectives. • Limited evidence of regularly meeting. Some Sprints have the ANNEXURE E: Sprint Attendance Record included. • Limited or no evidence of distributing work evenly. • No evidence of conflict resolution in the team. • Limited or no evidence of team collaboration to solve problems.	• Evidence of regular communication. • Documentation of retrospectives. • Evidence of regularly meeting. Most Sprints have the ANNEXURE E: Sprint Attendance Record included. • Limited or no evidence of distributing work evenly. • Limited or no evidence of conflict resolution in the team. • Limited evidence of team collaboration to solve problems.	• Clear evidence of regular communication. • Clear documentation of retrospectives. • Clear evidence of regularly meeting. Every Sprint has the ANNEXURE E: Sprint Attendance Record included. • Clear evidence of distributing work evenly between team members. • Clear evidence of conflict resolution in the team. • Clear evidence of team collaborating to solve problems.	/6

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
Project introduction clear and research logical	• No introduction to the project. • No research on the client organisation is included. • No Work agreement. • No definition of Ready (DoR) is included. • No definition of Done (DoD) is included. • No High-level plan of the team's sprints is included. • No actual sprints with user stories are included.	• An introduction to the project. • A work agreement is included. • A definition of Ready (DoR) is included. • A definition of Done (DoD) is included. • A High-level plan of the team's sprints is included. • None of the above has been copied from the manual or another team.	• Well-constructed introduction to the project is included. • Ethical and privacy concerns are highlighted in the context of the project. • Well described research into the client organisation is included and is logical. • Work agreement is included. • Definition of Ready (DoR) is included and is logical and easy to understand. • Definition of Done (DoD) is included and is logical and easy to understand. • High-level plan of the team's sprints is included.	• Well-constructed introduction to the project is included. • Ethical and privacy concerns are highlighted in the context of the project. • Well described research into the client organisation is included and is logical. • Work agreement that is unique and was followed throughout the project. • Definition of Ready (DoR) is included and is logical and easy to understand. • Definition of Done (DoD) is included and is logical and easy to understand. • High-level plan of the team's sprints is included. • Project risks with relevant mitigations were documented.	/10

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
			• Some Sprints with User Stories were included. • None of the above has been copied from the manual or another team.	• The actual sprints with User Stories are included to show how the team adjusted during the project. The documentation explains the challenges and how the team adjusted each sprint to reach the project objectives. • None of the above has been copied from the manual or another team.	
Systems Analysis and Design (hint SAND6221)					
Demonstrate effective requirements elicitation	• No evidence of requirements elicitation.	• User Roles are documented. • Use Stories are documented in the required format.	• User Roles are documented. • User Stories are all in the required format and documented clearly. • User stories show attempted prioritization and some estimations.	• User Roles are documented. • User Stories are all in the required format and documented clearly. • User Stories were prioritized throughout the project (every Sprint). • User Stories were groomed every Sprint and re-prioritization is evident.	/5

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
				• User Stories were groomed every Sprint and team estimations are evident. • Release notes for each Sprint are included. E.g., which User Stories were completed, and the features and bugfixes in the release are clearly documented.	
Demonstrate understanding of non-functional requirements	No evidence of any non-functional requirements: • Performance • Scalability • Reliability • Maintainability • Security • Usability • Interoperability • internationalisation / localisation	• Some Performance requirements included in user-stories (backlog). • Some Scalability requirements included in user-stories (backlog). • Some Reliability requirements included in user-stories (backlog).	• Performance requirements are included in user-stories (backlog). • Scalability requirements included in user-stories (backlog). • Reliability requirements are included in user-stories (backlog). • Maintainability requirements are included in user-stories (backlog). • Security requirements are	• Clear definition of Performance requirements included in user-stories (backlog). • Clear definition of Scalability requirements included in user-stories (backlog). • Clear definition of Reliability requirements included in user-stories (backlog). • Clear definition of Maintainability requirements included	/8

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
			included in user-stories (backlog).	in user-stories (backlog). • Clear definition of Security requirements included in user-stories (backlog). • Clear definition of Usability requirements included in user-stories (backlog). • Clear definition of Interoperability requirements included in user-stories (backlog). • Clear definition of internationalisation / localisation requirements included in user-stories (backlog).	
Demonstrate understanding of User Experience (UX)	• No User Experience Journey Map was included.	• User Experience Journey Map available for the normal user role.	• User Experience Journey Map available for review. • User Experience Journey Map includes paths for each User Role.	• User Experience Journey Map available for review. • User Experience Journey Map includes paths for each User Role and clearly shows which screens	/5

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
				and functions are accessible by only that Role. • User roles have limited overlap between screens. Administrators should not have access to features that regular users have access to.	
Demonstrate understanding of analysis	• No bounded contexts identified. • No UML documentation of the bounded contexts.	• Bounded context identified and documented.	• Bounded contexts identified and documented. • UML documentation of the bounded context (but no clear explanation of the mapping between the bounded context and the UML packages).	• Bounded contexts identified and clearly defined that match the problem domain clearly. • UML documentation of the identified bounded context as services. UML Package Diagram is expected as evidence; this needs to map to the bounded contexts described previously.	/4
Demonstrate understanding of software implementation documentation	• No documentation of software implementation. • Tool used to generate UML for all class objects	• UML object diagram showing class relationships for entities identified from analysis.	• UML object diagram showing class relationships for entities identified from analysis.	• UML object diagram showing class relationships for entities identified from analysis. • UML sequence diagrams show	/4

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
	after code was written.		• UML sequence diagrams show interactions between components for important complex flows	interactions between components for important complex flows. • UML State diagrams are included for important state transitions within your project.	
Architecture (hint CLDV6211/CLDV6212, PROG7311, PROG7312, OPSC7312)					
Demonstrate understanding of architecture	• No evidence of cloud architecture. • No evidence of decisions regarding cloud architecture. • No evidence of cloud networking. • No evidence of communication protocols. • No evidence of cloud security considerations.	• Cloud architecture clearly documented • Cloud architecture decisions clearly documented.	• Cloud architecture clearly documented • Cloud architecture decisions clearly documented. • Cloud networking clearly documented. • Protocols used for communication clearly documented and justified.	• Cloud architecture clearly documented • Cloud architecture decisions clearly documented. • Cloud networking clearly documented. • Protocols used for communication clearly documented and justified. • Cloud security clearly documented and justified.	/5
Demonstrate understanding of Design and Architecture patterns	• No documented Data Structures nor any use of well-known algorithms. For example, are there	• Some documentation of Data Structures and well-known algorithms being used. For example,	• Documented Data Structures and well-known algorithms being used. For example, is there	• Documented Data Structures and well-known algorithms being used. For example, is there any use of stacks, queues,	/4

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
	any use of stacks, queues, dictionaries, sets, trees, graphs, etc? • No documentation of data classes used for all data transfer between services in the code. • No documentation of architectural and design patterns. • No documentation of industry standard communication between frontend (UI) and backend services. • No documentation of integration with third-party API (webservices).	is there any use of stacks, queues, dictionaries, sets, trees, graphs, etc? The use of the structure must be appropriate to the problem being solved. • Some documentation of data classes used for all data transfer between services in the code. • Some documentation of architectural and design patterns. These patterns must exist in your GitHub repository (code). • Some documentation of industry standard communication between frontend (UI) and backend services. E.g., Webservices /	any use of stacks, queues, dictionaries, sets, trees, graphs, etc? The use of the structure must be appropriate to the problem being solved. An attempt at implementing this in code can be found in the GitHub repository. • Documentation of data classes used for all data transfer between services in the code. An attempt at implementing this in code can be found in the GitHub repository. • Documentation of architectural and design patterns. These patterns must exist in your GitHub repository (code). An	dictionaries, sets, trees, graphs, etc? The use of the structure must be appropriate to the problem being solved. Well-implemented code that is easy to read and understand can be found in the GitHub code repository exists. • Documentation of data classes used for all data transfer between services in the code. Well-implemented code that is easy to read and understand can be found in the GitHub code repository exists. • Documentation of architectural and design patterns. These patterns must exist in your GitHub repository (code). Well-implemented code that is easy to	

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
		APIs were used as an integration layer and did the team use the repository pattern for data access? • Some documentation of integration with third-party API (webservices). For example, using geolocation services, social media integration, etc.	attempt at implementing this in code can be found in the GitHub repository. • Documentation of industry standard communication between frontend (UI) and backend services. E.g., Webservices / APIs were used as an integration layer and did the team use the repository pattern for data access? Limited evidence in the GitHub code repository must exist before these marks can be given. • Documentation of integration with third-party API (webservices). For example, using geolocation services, social	read and understand can be found in the GitHub code repository exists. • Documentation of industry standard communication between frontend (UI) and backend services. E.g., Webservices / APIs were used as an integration layer and did the team use the repository pattern for data access? Well-implemented code that is easy to read and understand can be found in the GitHub code repository exists. • Documentation of integration with third-party API (webservices). Example, using geolocation services, social media integration, etc. Well-implemented code that is easy to read	

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
			media integration, etc. A clear attempt to implement this in the GitHub code repository must exist before these marks can be given.	and understand can be found in the GitHub code repository exists.	
Demonstrate understanding of data schemas	• No data storage description. • No entity relationship diagram (ERD). • The Data entities do not match objects described in design documentation (domain models). • There is a single data entity that has been defined.	• Data storage is clearly described. • The data storage explanation is supported by a logical entity relationship diagram (ERD).	• Data storage is clearly described. • The data storage explanation has an entity relationship diagram (ERD) that closely matches the description. • The data entities match objects described in design documentation (domain models).	• Data storage is clearly described. • The data storage explanation is well supported by a logical entity relationship (ERD) diagram. • Data entities match objects described in design documentation (domain models). • Multiple data entities are defined and their relationships are well defined.	/4
• **Digital Law and Ethics (hint DILE5111)**					
Demonstrate respect for profession and colleagues	• No declaration of original work (Annexure B) is included in submissions.	• Declaration of original work - Annexure B is completed.	• Declaration of original work - Annexure B is completed.	• Declaration of original work - Annexure B is completed.	/3

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
	• No references are included for work not originating from the team. • No use of a software license for the source code.		• References are included for all work that is not original from the team.	• References are included for all work that is not original from the team. • Use of a software license with justification about its selection.	
Application security (hint PROG7311)					
Demonstrate an understanding of application security	• No potential threat actors were documented. • No potential threat vectors were documented. • No mitigations for threats were documented: ◦ Economy of mechanism was not considered. • No discussion on balancing security with usability. • No Security was considered when accessing every data object:	• Potential threat actors documented. • Potential threat vectors documented.	• Potential threat actors documented. • Potential threat vectors documented. • Mitigations for threats were documented: ◦ Economy of mechanism was considered with each mitigation. • Logical discussion on balancing security with usability.	• Potential threat actors documented. • Potential threat vectors documented. • Mitigations for threats were documented: ◦ Economy of mechanism was considered with each mitigation. • Logical discussion on balancing security with usability. • Security was considered when accessing every data object: ◦ Complete mediation is logically discussed in the	/4

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
	◦ Complete mediation was not discussed.			context of the application.	
Operations					
Demonstrate an understanding of DevOps	• No flow chart describing the various steps in the GitHub pipeline was included. • No description of each step in the pipeline was included. • Testing output from the pipeline is available. E.g., A unit test report • No static code analysis using a third-party tool is included. • No security testing report is included.	• A clear description of the use of DevOps in the project is included and each step in the pipeline is well described. • A flow chart describing the various steps in the GitHub pipeline.	• A clear description of the use of DevOps in the project is included and each step in the pipeline is described. • A flow chart describing the various steps in the GitHub pipeline. • Testing output from the pipeline is available. E.g., A unit test report is included.	• A clear description of the use of DevOps in the project is included and each step in the pipeline is well described. • A flow chart describing the various steps in the GitHub pipeline. • Testing output from the pipeline is available. E.g., A unit test report • Static code analysis using a third-party tool is shown. E.g., Reports from the external tool are included. • A security testing report using a third-party tool from the GitHub pipeline is included.	/5

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
Demonstrate an understanding of running costs	• No predicted user growth is documented. • No technology scaling points are documented. • No predictive modelling for the next two years • No description planned technology replacements due to scaling have been documented.	• The predicted user growth is documented. • The scaling points of each technology used in the architecture has been related back to the number of users. E.g. How many users will cause the Azure compute services to scale?	• The predicted user growth is well documented. • The scaling points of each technology used in the architecture has been related back to the number of users. E.g. How many users will cause the Azure compute services to scale? • Create predictive models that cover three scenarios over two years on a monthly basis: ◦ Best case of user growth ◦ Worst case of user growth ◦ Mean (Avg) user growth	• The predicted user growth is well documented. • The scaling points of each technology used in the architecture has been related back to the number of users. E.g. How many users will cause the Azure compute services to scale? • Create predictive models that cover three scenarios over two years on a monthly basis: ◦ Best case of user growth ◦ Worst case of user growth ◦ Mean (Avg) user growth • A description of where adopting alternative technologies will prove better with scale. Document the costs of the replacement technology. E.g., when will a technology	/4

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
				limit be reached and will need to be replaced?	
Demonstrate an understanding of change management	• No argument for the organisational adoption of your software is included. • No argument for the user adoption of your software is included. • No description of your strategy to gain adoption from both the organisation and the users. • No strategy to maintain and support the software for the client is included. Or the strategy is unrealistic.	• An argument for the organisational adoption of your software is included. • An argument for the user adoption of your software is included.	• A well-formed argument for the organisational adoption of your software is included. • A well-formed argument for the user adoption of your software is included. • A clear description of your strategy to gain adoption from both the organisation and he users.	• A well-formed argument for the organisational adoption of your software is included. • A well-formed argument for the user adoption of your software is included. • A clear description of your strategy to gain adoption from both the organisation and the users. • A realistic strategy to maintain and support the software for the client is included.	/4
				TOTAL:	**Mark/75*100**

8 ANNEXURE H: GitHub Rubric

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
Basic Programming Skills (hint PROGf5111, PROG6112, PRLD5111)					
Demonstrate basic programming skills	• No evidence of basic concepts including use of methods; classes; loops; arrays; and appropriate use of variables. • No evidence of exception handling. • No evidence of good application structure that shows logical separation of code. This separation should map to the Domains identified and documented in the POE. • No evidence of following good programming practices.	• Evidence of basic concepts including use of methods; classes; loops; arrays; and appropriate use of variables. • Evidence of exception handling.	• Evidence of basic concepts including use of methods; classes; loops; arrays; and appropriate use of variables. • Evidence of exception handling. • Evidence of good application structure that shows logical separation of code. This separation should map to the Domains identified and documented in the POE.	• Evidence of basic concepts including use of methods; classes; loops; arrays; and appropriate use of variables. • Evidence of exception handling. • Evidence of good application structure that shows logical separation of code. This separation should map to the Domains identified and documented in the POE. • Evidence of following good programming practices. The results of a static code analysis tool are included and show level A (best level) for the code.	/4

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
Intermediate Programming Skills (hint PROG6211, PROG6212, OPSC7311)					
Demonstrate intermediate programming skills	• No evidence of Object Orientated Programming, such as: encapsulation, inheritance and polymorphism; and usage of interfaces. • No evidence of using the chosen programming language framework to accelerate development. Such as the use of an ORM (E.g., Entity Framework in C#) • No evidence of using parallel programming. Such as multithreading, async, etc. • No evidence of testing. Not limited to: Unit testing, integration testing, regression testing and functional testing. • No evidence of Internationalization	• Evidence of Object Orientated Programming, such as: encapsulation, inheritance and polymorphism; and usage of interfaces. • Evidence of using the chosen programming language framework to accelerate development. Such as the use of an ORM (E.g., Entity Framework in C#). • Evidence of testing. Not limited to: Unit testing, integration testing, regression testing and functional testing.	• Evidence of Object Orientated Programming, such as: encapsulation, inheritance and polymorphism; and usage of interfaces. • Evidence of using the chosen programming language framework to accelerate development. Such as the use of an ORM (E.g., Entity Framework in C#). • Evidence of using parallel programming. Such as multithreading, async, etc Evidence of testing. Not limited to: Unit testing, integration	• Evidence of Object Orientated Programming, such as: encapsulation, inheritance and polymorphism; and usage of interfaces. • Evidence of using the chosen programming language framework to accelerate development. Such as the use of an ORM (E.g., Entity Framework in C#). • Evidence of using parallel programming. Such as multithreading, async, etc. • Evidence of testing. Not limited to: Unit testing, integration testing, regression testing and functional testing. • Evidence of Internationalization or localization in the code base. E.g., resource files were used for all	/4

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
	or localization in the code base. E.g., resource files were used for all text displayed in the application. • No evidence of an effective user interface. Compare the User interface design with the implemented design. Are features available with minimal clicks in the application? Is there appropriate use of design - audio, images, fonts, text colour, video, etc. Does the choice of design improve usability or detract from it? Is the UI responsive under "real-world" conditions. For example, if the expected users have limited internet speeds (such as		testing, regression testing and functional testing	text displayed in the application. • Evidence of an effective user interface. Compare the User interface design with the implemented design. Are features available with minimal clicks in the application? Is there appropriate use of design - audio, images, fonts, text colour, video, etc. Does the choice of design improve usability or detract from it? Is the UI responsive under "real-world" conditions. For example, if the expected users have limited internet speeds (such as EDGE), will the app still work?	

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
	EDGE), will the app still work?				

Advanced Programming Skills (hint PROG7311, PROG7312, OPSC7312)					
Demonstrate advanced programming skills	• No Evidence of Data Structures and well-known algorithms being used. For example, is there any use of stacks, queues, dictionaries, sets, trees, graphs, etc? The use of the structure must be appropriate to the problem being solved. • No evidence of data classes used for all data transfer between services in the code. • No evidence of logical grouping of code in the structure of the folders. • No evidence of architectural and design patterns implemented. Check the POE, which architectural patterns were referenced and is there evidence that	• Some evidence of Data Structures and well-known algorithms being used. For example, is there any use of stacks, queues, dictionaries, sets, trees, graphs, etc? The use of the structure must be appropriate to the problem being solved. • Some evidence of data classes used for all data transfer between services in the code. • Some evidence of logical grouping of code in the structure of the folders.	• Evidence of Data Structures and well-known algorithms being used. For example, is there any use of stacks, queues, dictionaries, sets, trees, graphs, etc? The use of the structure must be appropriate to the problem being solved. • Evidence of data classes used for all data transfer between services in the code. • Evidence of logical grouping of code in the structure of the folders. • Evidence of architectural and design patterns implemented. Check the POE, which architectural patterns were referenced and is there evidence that	• Evidence of Data Structures and well-known algorithms being used. For example, is there any use of stacks, queues, dictionaries, sets, trees, graphs, etc? The use of the structure must be appropriate to the problem being solved. • Evidence of data classes used for all data transfer between services in the code. • Evidence of logical grouping of code in the structure of the folders. • Evidence of architectural and design patterns implemented. Check the POE, which architectural patterns were referenced and is there evidence that these were applied in the code? • Evidence of industry standard communication between frontend (UI) and backend services. E.g.,	/4

	these were applied in the code? • No evidence of industry standard communication between frontend (UI) and backend services. E.g., Webservices / APIs were used as an integration layer and did the team use the repository pattern for data access? • No evidence of integration with third-party API (webservices). Example, using geolocation services, social media integration, etc. • The application was not published to the Google Play store (nor any similar commercial app store) • No evidence of implementation of non-functional requirements. Use		these were applied in the code? • Evidence of industry standard communication between frontend (UI) and backend services. E.g., Webservices / APIs were used as an integration layer and did the team use the repository pattern for data access?	Webservices / APIs were used as an integration layer and did the team use the repository pattern for data access? • Evidence of integration with third-party API (webservices). Example, using geolocation services, social media integration, etc • The application was published to the Google Play store (or similar commercial app store) • Evidence of implementation of non-functional requirements. Use POE to cross-reference expected implementation in the GitHub repo. Check for evidence of performance; scalability; reliability; maintainability; usability; interoperability.	

	POE to cross-reference expected implementation in the GitHub repo. Check for evidence of performance; scalability; reliability; maintainability; usability; interoperability.				

Secure application development (hint PROG7311)					
Demonstrate understanding of secure application development	• No evidence of security considerations in the GitHub repo. Nothing in the code to confirm statements made in the POE regarding security. • No evidence of user inputs being sanitised. • No evidence of using secure protocols with all communication. • No evidence of masking used to protect sensitive information when logging. • No evidence of the principle of least privilege was followed. E.g., No user roles; or user roles have the same access. • No evidence of the principle of defence in depth was followed. E.g., No	• Compare code implementation with POE discussion on security; and use the GitHub repo to confirm statements made in the POE regarding security. E.g., Is there evidence of Complete Mediation in the code? Is there evidence of a balance of security with usability? Is there evidence of economy of mechanism? • Evidence of all user inputs sanitised. • Evidence of using secure protocols (e.g., TLS 1.3) with all communication. The use of secure protocols must align with the POE security discussion on protocols.	• Compare code implementation with POE discussion on security; and use the GitHub repo to confirm statements made in the POE regarding security. E.g., Is there evidence of Complete Mediation in the code? Is there evidence of a balance of security with usability? Is there evidence of economy of mechanism? • Evidence of all user inputs sanitised. • Evidence of using secure protocols (e.g., TLS 1.3) with all communication. The use of secure protocols must align with the POE security discussion on protocols.	• Compare code implementation with POE discussion on security; and use the GitHub repo to confirm statements made in the POE regarding security. E.g., Is there evidence of Complete Mediation in the code? Is there evidence of a balance of security with usability? Is there evidence of economy of mechanism? • Evidence of all user inputs sanitised. • Evidence of using secure protocols (e.g., TLS 1.3) with all communication between application frontend (UI) and backend. The use of secure protocols must align with the POE security discussion on protocols. • Evidence of masking used to protect sensitive information when logging.	/4

	strategies were used to provide multiple defensive measures in the event of the failure of a security control (or if a vulnerability is exploited)		• Evidence of masking used to protect sensitive information when logging.	• Evidence of the principle of least privilege was followed. E.g., Users have different roles to Administrators and these roles have appropriate levels of access in the application. • Evidence of the principle of defence in depth was followed. E.g., what strategies were used to provide multiple defensive measures in the event of the failure of a security control (or if a vulnerability is exploited).	
DevOps					
Demonstrate understanding of a DevOps pipeline	• No evidence of a working GitHub pipeline that automatically builds the application. • No evidence of running unit tests implemented in the GitHub pipeline. • No evidence of static code analysis	• Evidence of a working GitHub pipeline that automatically builds the application. • Evidence of running unit tests implemented in the GitHub pipeline. Unit tests should be valid tests that	• Evidence of a working GitHub pipeline that automatically builds the application. • Evidence of running unit tests implemented in the GitHub pipeline. Unit tests should	• Evidence of a working GitHub pipeline that automatically builds the application. • Evidence of running unit tests implemented in the GitHub pipeline. Unit tests should be valid tests that appropriately exercise the code under test.	/4

	in the GitHub pipeline. • No evidence of security testing in the GitHub pipeline.	appropriately exercise the code under test.	be valid tests that appropriately exercise the code under test. • Evidence of static code analysis in the GitHub pipeline.	• Evidence of static code analysis using a third-party tool in the GitHub pipeline. E.g., Reports from the external tool are available as part of the POE and match the output from the pipeline. • Evidence of security testing using a third-party tool in the GitHub pipeline. E.g., Reports from the external (third-party) tool are available in the POE and match the output from the pipeline.	
				TOTAL:	**/20**

9 ANNEXURE I: Presentation Rubric

This is the rubric your lecturer will mark your group and individual presentation against. Please refer to this when preparing your presentation.

PRESENTATION RUBRIC

NAME OF STUDENT GROUP ..... **MODULE: .....**

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
VISUAL AIDS & TIMING					
Physical & Electronic e.g., demonstration of working application, PowerPoint slides, etc	• Unrelated to presentation.	• Poor, distracts audience, adds nothing to presentation.	• Commercially available visual aids, relevant to topic, enhance understanding and explanation.	• Original visual aids, relevant to topic, support and enhance understanding and explanation.	/5
GROUP DYNAMIC					
Present as a team	• Only one person presents.	• The whole team are involved in the presentation.	• The whole team are engaged with the audience while presenting.	• Presentation flows naturally between all team members without any prompting from other team members.	/5
GENERAL LECTURER FEEDBACK:					
				TOTAL:	**/10**

INDIVIDUAL PRESENTATION RUBRIC

NAME OF STUDENT ..... **MODULE: .....**

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
NON-VERBAL SKILLS (CCFO5)					
Audience Engagement	• Makes no attempt to interact with the audience.	• Sometimes interacts with one or two members of the audience.	• Has frequent interaction with the same people in the audience.	• Holds attention through direct interaction with various members of audience.	/5
VERBAL SKILLS					
Delivery	• Shows no interest in topic or activity/does not participate in oral part of presentation.	• Mumbles, appears distracted or unfocused, reads notes word for word.	• Thoughts well-articulated, uses own words, but unable to keep audience engaged throughout presentation.	• Enthusiastic, relaxed, self-confident, seldom refers to notes, maintains interest of audience throughout presentation.	/5
SUBJECT KNOWLEDGE					
Understanding of subject	• Demonstrates no understanding of concepts and unable to answer questions	• Demonstrates limited understanding of concepts and has difficulty answering questions.	• Demonstrates adequate understanding of concepts and able to answer most questions.	• Demonstrates deep understanding of concepts and is able to provide in-depth explanations in response to all questions.	/5
GENERAL LECTURER FEEDBACK:					
				TOTAL:	**/15**

10 ANNEXURE J: Individual Rubric

CRITERIA	0-1 Does not meet the required standards	2 – Meets the required standards	3 – Partially exceeds the required standards	4 – Exceeds the required standards	TOTAL
GITHUB CONTRIBUTION					
Evidence of contribution using GitHub contribution page on team repo website	• No evidence of contribution.	• Some evidence of contribution can be found at the end of the project timeline.	• Regular contribution throughout the timeline of the project.	• Regular contribution throughout the timeline of the project. • Contribution shows code added and removed (refactoring).	/5
POE CONTRIBUTION					
Evidence of contribution to the PoE	• No evidence of contribution.	• Some evidence can be found of contribution to the PoE.	• Clear evidence of contribution to the project can be found. • Contribution aligns with team role.	• Clear evidence of contribution to the project can be found. • Contribution aligns with team role. • Contribution is valuable and of a high quality.	/5
				TOTAL:	**/10**

11 ANNEXURE K: LECTURER FEEDBACK TO STUDENT

NAME OF STUDENT: ..... STUDENT NUMBER: ..... MODULE: .....

	Task	Student Submission	Maximum Mark	Weighting	Student Mark	Comments
PoE	Consolidation of all evidence supporting the submission for this module.		60	37.5%		
GitHub	All code used to create your final application. This should include a GitHub Actions pipeline.		35	21.875%		
Contribution	GitHub contributions as well as PoE document contributions will be used to determine this mark.		10	6.25%		
Presentation	Final Project Presentation		25	15.625%		
Attendance			10	6.25%		
Peer - Evaluation			20	12.5%		
TOTAL			**160**			
			Mark %			

Comments_____

_____

_____

12 ANNEXURE L SELF and Peer – EVALUATION

REFLECTIVE REPORT

Please complete and include this self-reflective report when submitting your final PoE.

Using a reporting structure complete the following:

Introduction

Write an introductory paragraph in which you briefly outline your understanding of the purpose and value of WIL.

Skills Learnt

Identify the skills you have learnt. State how you used/were expected to use each skill during your WIL. Consider skills under each of the following three categories and report on each:

Industry specific practices, e.g. media monitoring, compiling media kits, writing articles for the staff newsletter, etc.

- Interpersonal communication skills, e.g. brainstorming sessions, feedback sessions, staff meetings or briefing and debriefing sessions, etc.
- Management skills, e.g. time management to meet deadlines, crisis management to solve unexpected problems, etc.

Role in the team

Describe the team dynamic during your WIL and whom you reported to and with whom you were in a team with. Comment on your role in the team with regard to all of the following points:
Leadership responsibilities and being provided instruction.

- Your contribution to team success.
- The group dynamic and your contribution to the group/team as a whole.
- Dealing with concerns, complaints, queries and conflict.

Research, technology and the presentation of information

Finding information that is both relevant and useful is a much-needed skill in WIL.

- Describe one (1) or two (2) scenarios in which you were expected to find information for a task or duty that you had to complete. This can be related to online research, finding client or supplier contact information, or looking through files and databases to find relevant data.
- Where did you find the information you needed to do this work?
- What technology did you use?
- How did you have to present the information you found?

Personal strengths (strong points) and weaknesses (areas to do better in) Comment on the elements, tasks or duties during your WIL that you found yourself excel in, as well as the ones you found difficult to master. • List and describe the tasks that you did really well in. • Identify at least five strengths that you realised you have. • List and describe the tasks that you did not do well in. • Why in your opinion, did you not perform well in these tasks? • Comment on how you think you can improve on the weaknesses that you identified.
Stakeholder relationship Describe your relationship with the WIL Coordinator or in the case of placement, the mentor in the workplace by focusing on the following areas: • Part of this relationship that worked well for you and parts that did not. • Explain how you think you could have made the relationship better or stronger.
Impact This refers to your contributions to the organisation during your placement there. • Comment on how you think others (if placed, management, fellow staff members, team members, clients, suppliers and others you worked with during your placement) benefitted from you being there and the work you did. • Describe how you have made a better/greater/more positive impact.
Conclusion Write a summary whereby a clear overall impression of your WIL experience is provided.

The lecturer will use the rubric below to mark your self-reflection report. Consider the criteria when compiling your report.

REFLECTION REPORT CRITERIA		0-1 Does not meet the required standard	2- Meets the required standard	3- Exceeds the required standard	TOTAL
Introduction (CCFO8) Write an introductory paragraph in which you briefly outline your understanding of the purpose and value of WIL		• Lack of understanding of the purpose and value of WIL. • Did not refer to preparation for the world of work. • Did not mention concepts from any modules.	• Some understanding of the purpose and value of WIL. • Could relate to the world of work, but did not mention concepts from modules.	• Clear understanding of the purpose and value of WIL. • Explained the relationship between the world of work and the concepts from a range of modules.	/3
	0-1 Does not meet the required standard	**2- Meets the required standard**	**3- Partially exceeds the required standard**	**4- Greatly exceeds the required standard**	
Skills Learnt (CCFO1; CCFO2; CCFO3; CCFO4; CCFO8) Identify the skills you have learnt. State how you used/were expected to use each skill during your WIL.	• The student did not reflect on the skills they learnt. • Limited to no details or examples were provided.	• The student thought about some skills that they learnt during the WIL. • Some examples were provided as per brief	• The student clearly considered and reflected some understanding of the skills they learnt.	• The student fully understands and can explain to others what skills they learnt in the WIL module.	

		.	• Detailed examples were provided, however there was no or limited reflection on the skills learnt.	• Detailed examples for what and how the student learnt were provided. • Reflection on the skills learnt is complete and done well in line with the brief.	/4
	0-1 Does not meet the required standard	**2- Meets the required standard**	**3- Partially exceeds the required standard**	**4- Greatly exceeds the required standard**	
Role in the team (CCFO2; CCFO8) Describe the team dynamic during your WIL. Who you reported to and who you were on a team with. Comment on your role in the team.	• The student did not clearly reflect on the team dynamic. • The role of the student concerning their role in the team is not clear.	• The student reflected on the team dynamic and some key issues concerning their role in the team were described.	• The student clearly reflected on the team dynamic and underlined all key issues. • The student reflected on how	• The team dynamic is clearly described. • The role of the student concerning their role in the team is clear and in line with the brief. • The student reflected on their	

			they contributed to team success.	contribution to the team success and how they addressed concerns and/or complaints.	/4
Research, technology and the presentation of information (CCFO5; CCFO6; CCFO8) Finding information that is both relevant and useful is a much-needed skill in WIL. • Describe one or two scenarios in which you were expected to find information for a task or duty that you had to complete. This can be related to online research, finding client or supplier contact information, or looking through files and databases to find relevant data. • Where did you find the information you needed to do this work? • What technology did you use? • How did you have to present the information you found?	• The student did not clearly reflect on research, technology and the presentation of information.	• The student reflected on a limited number of key issues concerning research, technology and the presentation of information. • The student attempted to share how the information was found, used and presented.	• The student clearly reflected on the key issues concerning research, technology and presentation of information. • The student clearly addressed how and where information was found and what technology was used.	• Most aspects relating to research, technology and the presentation of information is clearly described as per brief. • The student presented clear scenarios that they had to complete. • The student clearly described where the information was found and what technology was used.	

					/4
	0-1 Does not meet the required standard	**2- Meets the required standard**	**3- Partially exceeds the required standard**	**4- Greatly exceeds the required standard**	
Personal strengths (strong points) and weaknesses (areas to do better in) (CCFO8) Comment on the elements, tasks or duties during your WIL that you found yourself excel in, as well as the ones you found difficult to master. • List and describe the tasks that you did really well in. • Identify at least five strengths that you realised you have. • List and describe the tasks that you did not do well in. • Why in your opinion, did you not perform well in these tasks? • Comment on how you think you can improve on the weaknesses that you identified.	• The student did not accurately reflect on their personal strengths and weaknesses. • Limited to no details were provided as per brief and the reflection lacks insight on how weaknesses can be improved.	• The student displayed some understanding of their personal strengths and weaknesses. • Some details were provided as per brief and the reflection included a satisfactory description of how weaknesses can be improved.	• The student displayed a good understanding of their personal strengths and weaknesses. • Most of the details were provided as per the brief and the reflection included a more that satisfactory description of how weaknesses can be improved.	• The student fully recognises their personal strengths and weaknesses. • Details and examples were provided as per brief and the student clearly understands how to improve on their weaknesses.	

					/4
	0-1 Does not meet the required standard	**2- Meets the required standard**	**3- Partially exceeds the required standard**	**4- Greatly exceeds the required standard**	
Stakeholder relationship (CCFO2; CCFO8) Describe your relationship with the WIL Coordinator or in the case of placement, the mentor in the workplace by focusing on the following areas: • Part of this relationship that worked well for you and parts that did not. • Explain how you think you could have made the relationship better or stronger.	• The student did not adequately describe their relationship with the WIL Coordinator and/or mentor. • Limited to no understanding was shown on how the quality of the relationship could have been enhanced.	• The student displayed satisfactory understanding of their relationship with the WIL Coordinator and/or mentor. • The student provided some details on how the relationship could have been improved.	• The student displayed a more than satisfactory understanding of the relationship with the WIL Coordinator and/or mentor. • Details were provided on which part of the relationship worked well and parts that did not. • The student provided clear details on how the relationship could have been improved.	• The student fully understands their relationship with the WIL Coordinator and/or mentor. • Details were provided on which part of the relationship that worked well and parts that did not. • The student is also able to comprehensively explain how the relationship could have been made stronger or better. • Overall, the reflection on the stakeholder	

				relationship is complete and done well in line with the brief.	/4
	0-1 Does not meet the required standard	**2- Meets the required standard**	**3- Partially exceeds the required standard**	**4- Greatly exceeds the required standard**	
Impact (CCFO8) This refers to your contributions to the organisation during your placement there. • Comment on how you think others (management, fellow staff members, team members, clients, suppliers and others you worked with during your placement) benefitted from you being there and the work you did.	• The student did not think about their contributions during WIL.	• The student provided limited insight into their contributions during WIL.	• The student provided clearly articulated insight into their contributions during WIL. • It was evident how the student made a positive impact to the organisation.	• The student's reflection on their contribution during WIL is complete and done well in line with the brief. • It was well evidenced how the student made a positive impact to the organisation.	

• Describe how you have made a better/greater/more positive impact.					/4
		0-1 Does not meet the required standard	**2- Meets the required standard**	**3- Exceeds the required standard**	
Conclusion (CCFO4; CCFO8) Write a summary whereby a clear overall impression of your WIL experience is provided.		• The student did not provide a clear summary of their overall impression of their WIL experience	• The student provided an adequate summary of their overall impression of their WIL experience. More details could have been included in this regard.	• The student provided a clear and detailed summary of their overall impression of their WIL experience.	/3
TOTAL					**/30**

Peer – EVALUATION Form

Peer Review Evaluation Form
Module Code: INSY7315
Group Name: [Provide Group Name]
Evaluator's Name: [Provide Your Name]

Instructions
For each member (excluding yourself), rate their performance in the listed criteria on a scale of 1 (Poor) to 5 (Excellent). Add comments where necessary.

Criteria	Member 1 Name	Member 2 Name	Member 3 Name	Member 4 Name	Member 5 Name
Contribution to the Teams Work (1-5)					
Comments					
Team Collaboration and Communication (1-5)					
Comments					
Adherence to deadlines (1-5)					
Initiative and Problem-Solving (1-5)					
Comments					

Overall Feedback
Please provide any additional comments on each member’s overall performance and teamwork:

Intellectual Property

Plagiarism occurs in a variety of forms. Ultimately though, it refers to the use of the words, ideas or images of another person without acknowledging the source using the required conventions. The IIE publishes a Quick Reference Guide that provides more detailed guidance, but a brief description of plagiarism and referencing is included below for your reference. It is vital that you are familiar with this information and the Intellectual Integrity Policy before attempting any assignments.

Introduction to Referencing and Plagiarism

What is 'Plagiarism'?

'Plagiarism' is the act of taking someone's words or ideas and presenting them as your own.

What is 'Referencing'?

'Referencing' is the act of citing or giving credit to the authors of any work that you have referred to or consulted. A 'reference' then refers to a citation (a credit) or the actual information from a publication that is referred to.

Referencing is the acknowledgment of any work that is not your own, but is used by you in an academic document. It is simply a way of giving credit to and acknowledging the ideas and words of others.

When writing assignments, students are required to acknowledge the work, words or ideas of others through the technique of referencing. Referencing occurs in the text at the place where the work of others is being cited, and at the end of the document, in the bibliography.

The bibliography is a list of all the work (published and unpublished) that a writer has read in the course of preparing a piece of writing. This includes items that are not directly cited in the work.

A reference is required when you:

- Quote directly: when you use the exact words as they appear in the source;
- Copy directly: when you copy data, figures, tables, images, music, videos or frameworks;
- Summarise: when you write a short account of what is in the source;
- Paraphrase: when you state the work, words and ideas of someone else in your own words.

It is standard practice in the academic world to recognise and respect the ownership of ideas, known as intellectual property, through good referencing techniques. However, there are other reasons why referencing is useful.

Good Reasons for Referencing

It is good academic practice to reference because:

- It enhances the quality of your writing;
- It demonstrates the scope, depth and breadth of your research;
- It gives structure and strength to the aims of your article or paper;
- It endorses your arguments;
- It allows readers to access source documents relating to your work, quickly and easily.

Sources

The following would count as 'sources':

- Books,
- Chapters from books,
- Encyclopaedias,
- Articles,
- Journals,
- Magazines,
- Periodicals,
- Newspaper articles,
- Items from the Internet (images, videos, etc.),
- Pictures,
- Unpublished notes, articles, papers, books, manuscripts, dissertations, theses, etc.,
- Diagrams,
- Videos,
- Films,
- Music,
- Works of fiction (novels, short stories or poetry).

What You Need to Document from the Hard Copy Source You are Using

(Not every detail will be applicable in every case. However, the following lists provide a guide to what information is needed.)

You need to acknowledge:

- The words or work of the author(s),
- The author(s)'s or editor(s)'s full names,
- If your source is a group/ organisation/ body, you need all the details,
- Name of the journal, periodical, magazine, book, etc.,
- Edition,
- Publisher's name,
- Place of publication (i.e. the city of publication),
- Year of publication,
- Volume number,
- Issue number,
- Page numbers.

What You Need to Document if you are Citing Electronic Sources

- Author(s)'s/ editor(s)'s name,
- Title of the page,
- Title of the site,
- Copyright date, or the date that the page was last updated,
- Full Internet address of page(s),
- Date you accessed/ viewed the source,
- Any other relevant information pertaining to the web page or website.

Referencing Systems

There are a number of referencing systems in use and each has its own consistent rules. While these may differ from system-to-system, the referencing system followed needs to be used consistently, throughout the text. Different referencing systems cannot be mixed in the same piece of work!

A detailed guide to referencing, entitled Referencing and Plagiarism Guide is available from your library. Please refer to it if you require further assistance.

When is Referencing Not Necessary?

This is a difficult question to answer – usually when something is 'common knowledge'. However, it is not always clear what 'common knowledge' is.

Examples of 'common knowledge' are:

- Nelson Mandela was released from prison in 1990;
- The world's largest diamond was found in South Africa;
- South Africa is divided into nine (9) provinces;
- The lion is also known as 'The King of the Jungle'.
- $E = mc^2$
- The sky is blue.

Usually, all of the above examples would not be referenced. The equation $E = mc^2$ is Einstein's famous equation for calculations of total energy and has become so familiar that it is not referenced to Einstein.

Sometimes what we think is 'common knowledge', is not. For example, the above statement about the sky being blue is only partly true. The light from the sun looks white, but it is actually made up of all the colours of the rainbow. Sunlight reaches the Earth's atmosphere and is scattered in all directions by all the gases and particles in the air. The smallest particles are by coincidence the same length as the wavelength of blue light. Blue is scattered more than the other colours because it travels as shorter, smaller waves. It is not entirely accurate then to claim that the sky is blue. It is thus generally safer to always check your facts and try to find a reputable source for your claim.

Important Plagiarism Reminders

The IIE respects the intellectual property of other people and requires its students to be familiar with the necessary referencing conventions. Please ensure that you seek assistance in this regard before submitting work if you are uncertain.

If you fail to acknowledge the work or ideas of others or do so inadequately this will be handled in terms of the Intellectual Integrity Policy (available in the library) and/ or the Student Code of Conduct – depending on whether or not plagiarism and/ or cheating (passing off the work of other people as your own by copying the work of other students or copying off the Internet or from another source) is suspected.

Your campus offers individual and group training on referencing conventions – please speak to your librarian or ADC/ Campus Co-Navigator in this regard.

Reiteration of the Declaration you have signed:

1. I have been informed about the seriousness of acts of plagiarism.
2. I understand what plagiarism is.
3. I am aware that The Independent Institute of Education (IIE) has a policy regarding plagiarism and that it does not accept acts of plagiarism.
4. I am aware that the Intellectual Integrity Policy and the Student Code of Conduct prescribe the consequences of plagiarism.

5. I am aware that referencing guides are available in my student handbook or equivalent and in the library and that following them is a requirement for successful completion of my programme.
6. I am aware that should I require support or assistance in using referencing guides to avoid plagiarism I may speak to the lecturers, the librarian or the campus ADC/ Campus Co-Navigator.
7. I am aware of the consequences of plagiarism.

Please ask for assistance prior to submitting work if you are at all unsure.