Innovate: Coding coursework
Student Name: Nigel Martin

This document details the written coursework to be completed alongside your projects or as a retrospective for the Code Nation Innovate: Coding course.

Learners studying at level 3 are expected to evidence excellent English skills and complete at least the equivalent of two paragraphs (over 100 words) per question.

You can use experience of the any of the project lifecycles or projects completed during this course to answer these questions, as well as any research where appropriate (please include any sources you use).

Collaborative technologies

1. Discuss how different collaborative technologies can be used and integrated and explore any potential compatibility issues. (Describe any potential access issues from tools or devices, and how to overcome them).

2. Identify risks associated with using collaborative technologies and describe how you can prevent and manage them. (How can a developer ensure the security of a project for both client and server side?)

3. Discuss the effectiveness of collaborative technologies used in this project (Describe the features, benefits, limitations and potential issues of using these tools and how this affected your project).

Project development, testing and management

4. Explain the benefits and drawbacks of using your chosen development environment and file management. (How would you ensure an efficient development environment for a future project? Reflect on any different principles, tools or technologies you could use).

5. Describe why a developer needs to test their projects and explain potential methods of testing through the lifecycle of development project. (Explain the effectiveness of testing and discuss any potential boundaries or corners when it comes to testing).

6. Describe project management techniques that you have used throughout this project and explain any specific methodologies you have utilised. (Explain the effectiveness and efficiency of these techniques or methods, and how you would adapt to improve for future projects).

Collaborative technologies

1. Discuss how different collaborative technologies can be used and integrated and explore any potential compatibility issues. (Describe any potential access issues from tools or devices, and how to overcome them).

Collaboration tools allow modern work environments to achieve a heterogeneous workflow between all business and trading entities, regardless of that entity's location. Team members, teams, departments, business units, stakeholders, security entities and clients can all participate on the provisor they have suitable computing and internet resources.

In the main, most of these technologies are easily accessed via an internet web browser and exist as Soft-as-a-Solution (SaaS) suites and/or applications. Vendors tend to provide a free tier of access to their SaaS and indeed a thriving market exists. However, even though the options are many and the vendor landscape is vast, vendor lock-in, and inter suite incompatibility are commonplace.

On the recent Innovate Coding 030522 training course, the SaaS application Zoom eas used for video conferenced delivery of training. Slack was used for communications, scheduling, file resource sharing, but also some training through code snippets, internet references etc. These solutions though not integrated performed their functions without issue, fault or major tinkering.

Conversely, Visual Studio Code (Code) was utilized as the core development environment. With wrappers to the external python suite, extensions and plug-ins, Code became an Integrated Development Environment (IDE). Unfortunately, even though it functioned very well it felt less robust, as additional parts had to be "plugged in" to make our environment complete. Further, Code and the plug-ins received regular updates (mostly automated installs), which left this experienced computer user with a sense of trepidation… What was wrong with the last version? Will I encounter issues or incompatibilities with this version? How do I roll back versions if there is an issue? Will lose my code? How long will an issue like this take to fix?

2. Identify risks associated with using collaborative technologies and describe how you can prevent and manage them. (How can a developer ensure the security of a project for both client and server side?)

The author admits to being insufficiently experienced with possible security issues specifically related to the area of collaboration and collaborative technologies beyond the usual suspects: unencrypted embedded communications, network transit-related transfer capture or transfer corruption, server-side spoofing and disingenuous clients/stakeholders/colleagues stealing code/work product.

In using the Github resource for this course, a network-based link was created between the local machine and a Github repository at Microsoft, but the author is unable to speak about the safety or efficacy of this configuration/arrangement. Can an outside entity use the link to "push" changes to either the local machine or remote repository? Git was created by Linus Torvalds as a full replacement for the Bitkeeper "commercial version control software" used in the early days of the Linux kernel project. It was not built for security; it was built for distributed collaboration. So, we can assume it has some security issues.
https://www.linuxjournal.com/content/git-origin-story
https://spectralops.io/blog/8-top-git-security-issues-what-to-do-about-them/

All this being said, Github does provide public, private and shared repositories. The latter two are considered development domains, used prior to the licencing and offering of completed release candidates.

Moving away from Github, Zoom, Slack and pretty much any piece of collaborative network software may be abused if the local machine has been hacked. Also, all network communication can be captured, copied and manipulated enroute, which is why encryption should be employed when required.

3. Discuss the effectiveness of collaborative technologies used in this project (Describe the features, benefits, limitations and potential issues of using these tools and how this affected your project).

The list of collaborative technologies used in this project includes Visual Studio Code (Code), Zoom, Slack, Github (mostly via the Git Command Line Interface) and additionally for the Kanban element of the final project, Github Projects or Trello (the author chose Trello). These tools were key to “Innovate Coding” as the course was delivered remotely via the internet.

Zoom is a SaaS meeting and collaboration tool that came to prominence during the period of the COVID lockdown. Its combination of simplistic usage, a generous free tier duration, larger free tier attendee limit and very affordable upper tiers, quickly made it the ideal choice, especially when compared to the expensive Cisco and HP options, the lower quality Microsoft option, the less well-known Google option, plus other third-party solutions. Although it should be noted that the software has not been without its controversies.

Slack, like other start-up businesses in the SaaS arena, provides a free tier and as such a low cost of entry. Its messaging system is functional, extremely flexible and has firmly outpaced mobile phone dedicated messaging options.

GitHub and Trello where invaluable tools for the production and delivery of project work.

All in all, I cannot fault the software we were supplied with as tools for this course.

Project development, testing and management

4. Explain the benefits and drawbacks of using your chosen development environment and file management. (How would you ensure an efficient development environment for a future project? Reflect on any different principles, tools or technologies you could use).

The use of Visual Studio Code (Code) as a development environment was the choice of the trainers. However, I found it to be intuitive, easy to understand and therefore easy to use. As mentioned elsewhere in this document my main issue was the volume of updates that I received and the fear that an update would break Code's functionality, leaving me without a working development environment.

Code extensions were available to enhance the HTML, JavaScript and Python programming workflow with auto-complete and IntelliSense. However, it seemed nothing existed to recognise/auto-complete the Jinger2 syntax within the flask HTML templates. So that was a drawback.

It's difficult in these situations to know whether to go it alone and create a file structure or wait for the trainer to instruct how that file structure should be formatted. In the end, a file structure that encapsulated the activities/challenges with the code attempts was employed.

When it came to the Flask micro-framework, file structure was very important to prevent code from breaking. The HTML templates were held in a "templates" folder (child to root) and fixed website elements that do not change were held in the "static" folder (also child to root). Web site elements that are not expected to be changed by the website include CSS files, JavaScript files, images, audio, video etc. If multiples of any particular type existed (I.e. CSS documents) an appropriate folder was created to hold them. The top-level only had an "__init__.py" or "app.py" (depending on which Flask tradition was being followed).

```
FSP-MOVIE
|   app.py
|   views.py
|
+---static
|   +---css
|   |       style.css
|   |
|   +---img
|   \---js
|
+---templates
        404.html
        500.html
        base.html
```

5. Describe why a developer needs to test their projects, and explain potential methods of testing through the lifecycle of development project. (Explain the effectiveness of testing and discuss any potential boundaries or corners when it comes to testing).

The trainers encouraged the use of Kanban, an Agile derivative. Put simply, small functional segments of code created and tested, solve a small aspect of the software solution being developed. Once a segment is complete, another is allocated from a ToDo list. Complete next, complete next, over and over.

Sometimes a segment cannot be completed because it depends on one or more other segments, that have not been created yet. That segment becomes “blocked” and work continues with some other segment until the block has been removed. Segments are created and tested, later segments are integrated and tested. Eventually, we have a final product. If memory serves, the time window in which a segment is supposed to be created and then tested is called a sprint.

Test-driven development (TDD) differs from the above in that the focus of the developer is writing valid test code first. That test should fail in the absence of actual production code. Then the programmer may write only as much production code as is necessary to make the test pass. This cycle of “test, fail, production, pass” can continue within Kanban sprints and ultimately generates a necessary suite of tests required for future solution development or changes.

The author did not feel confident enough to attempt TDD and simply stuck with the first method described, ‘small segments of code’. Also, there were no sprints. A Trello board was employed to note the small segments/steps of the project.

Team Boundary Management describes the ability of a team to establish and maintain psychological boundaries to the outside world. Conclusion: Teams with a pronounced team boundary management are much less likely to fall into the acceleration trap. This is shown by a study by Dr. Ulrich Leicht-Deobald (IWE-HSG), which he conducted with Prof. Dr. Heike Bruch from the Institute for Leadership and Human Resource Management (IfPM-HSG) and Jakob Mainert from the COSA Institute Luxembourg.

Ref: https://www.youtube.com/watch?v=PFKrTZ0U2JQ

6. Describe project management techniques that you have used throughout this project and explain any specific methodologies you have utilised. (Explain the effectiveness and efficiency of these techniques or methods, and how you would adapt to improve for future projects).

The author applied a Kanban methodology to the development process, taking a section of the problem domain and creating code to solve it. Over and over until the basic skeleton of the Flask website was ready. After two days and a sleepless night, it became evident very quickly that there was not sufficient time to create the first choice of website. Even though the assets and JavaScript resources were already available, database implementation and AJAX or Websocket IO were non-trivial requirements that needed adding to this Heroku implementation. In the end, the author had to revise the project and design something simpler.

In the author's humble opinion, there is something to be said about keeping the analysis and design phases of the Waterfall methodology and following through with an Agile development cycle. Issues, blockers and change requests would then be fed back into the Waterfall's analysis and design phases.

Taking elements from different methodologies and merging them where it is possible could lead to the best of both worlds: the niffy Agile creation/test cycle at the coal face and the 1000ft view from above visualising the coal mines. What an analogy.

Level 3 Unit Tracker			
Q1	3.2.3, 3.2.4	Q4	3.3.1, 33.1.1, 33.1.2, 33.1.3
Q2	3.1.1, 3.1.3, 3.1.5, 3.1.6	Q5	33.1.4, 33.3.4
Q3	3.2.1, 3.4.5	Q6	33.2.1, 33.2.4, 33.3.1, 33.3.2
FOR INTERNAL REFERENCE ONLY			