Project: Terminal Application

Overview

Title	Plan and Implement a Terminal Application
ID	T1A2
Type	Assessment - Individual - Industry simulated project
Release Date	Week 3, Term 1 - Tuesday
Due Date	Week 3, Term 1 - Friday
Course	Diploma of Information Technology - Bootcamp Delivery Mode (Fast Track)
Subject	CMP1041 - Foundation Programming PRG1002 - Programming I
Weighting	71% of CMP1041 45% of PRG1002
Marks	Total = **130.5** marks - 90 marks for CMP1041 - 40.5 marks for PRG1002
Assessment Policies	Coder Academy utilises the Academy of Information Technology's policies relating to assessment, academic matters and student welfare. If you have any questions about these policies, please speak to your educator, or Coder Academy's student services team. A penalty of 10% of the total mark per day may apply for late submissions, inclusive of weekends.

Introduction

Terminal applications, also known as command line applications (or **apps** for short), are run from the shell and accept input from stdin and arguments such as flags and options, and produce an output that is useful to the user.

You will create your own terminal app because:

- terminal applications are an essential part of learning to utilise the functions of all operating systems efficiently and in a way that promotes your understanding of the inner workings and processes of operating systems and the hardware itself;
- many applications used across IT functions utilise terminal applications as a standard, rather than applications with a graphical user interface;
- developing your own terminal application will increase your technical sophistication and ability to use other terminal applications.

Brief

You are to design, implement and test a terminal application and throughout the process demonstrate that you are able to use a range of developer tools.

The mandatory requirements/constraints for your application are:

- accept user input in the form of a file or text input
- produce printed output or interact with the file system

> You must get your Educators approval for your *app* idea prior to working on the **software development plan** or implementing the application itself

Requirements

Requirements for this project are divided into three major parts,

1. Documentation
2. Presentation
3. Code

Documentation Requirements

Documentation for this project must be supplied as part of the following documents,

1. **README** document
2. Slide deck

README document

Project documentation is to be compiled as a single markdown file named `README.md`. This file should contain,

No.	Requirement	Word Count
R1	Answers to all the *documentation requirements* below.	

No.	Requirement	Word Count
R2	Your `README.md` should have a separate heading for each *documentation requirement* and answers organised under the appropriate headings.	
R3	Provide full attribution to referenced sources (where applicable).	
R4	Provide a link to your source control repository	
	Design a **Software Development Plan** for a terminal application. The following requirements provide details of what needs to be included in this plan,	
R5	**Develop** a statement of **purpose** and **scope** for your application. It must include: - **describe** at a high level what the application will do - **identify** the problem it will solve and **explain** why you are developing it - **identify** the target audience - **explain** how a member of the target audience will use it	300 - 500 words
R6	**Develop** a list of features that will be included in the application. It must include: - at least THREE **features** - **describe** each feature **Note: Ensure** that your features above allow you to demonstrate your understanding of the following language elements and concepts: - use of variables and the concept of variable scope - loops and conditional control structures - error handling **Consult with your educator** to check your features are sufficient .	300 words (approx. 100 words per feature)

No.	Requirement	Word Count
R7	**Develop** an **outline** of the user interaction and experience for the application. Your outline must include: - how the user will find out how to interact with / use each feature - how the user will interact with / use each feature - how errors will be handled by the application and displayed to the user	
R8	**Develop** a diagram which describes the control flow of your application. Your diagram must: - show the workflow/logic and/or integration of the features in your application for each feature. - utilise a recognised format or set of conventions for a control flow diagram, such as UML.	
R9	**Develop** an implementation plan which: - **outlines** how each feature will be implemented and a checklist of tasks for each feature - prioritise the implementation of different features, or checklist items within a feature - provide a deadline, duration or other time indicator for each feature or checklist/checklist-item Utilise a suitable project management platform to track this implementation plan > Your checklists for each feature should have at least 5 items.	
R10	**Design** help documentation which includes a set of instructions which accurately **describe** how to use and install the application. You must include: - steps to install the application - any dependencies required by the application to operate - any system/hardware requirements	

Slide deck

You must present your Terminal application and it's code - which will form the basis of a professional report.

The presentation and associated Slide deck should include,

Present your Terminal application to the class. You must provide a walk-through of the logic of your application and how the application is used.

- A walk-through of your Terminal application, its features and how it used
- A walk-through of the logic of your Terminal application and code
- A review of your development/build process including challenges, ethical issues, favourite parts, etc

Make sure to incorporate the following into your presentation Slide deck,

No.	Slides must include...	You must explain...
R11	An overview of your Terminal application	The main features and overall structure of your *app*
R12	An overview of your code	An explanation of the important parts of your code

Presentation Requirements

Your presentation must:

- be no longer than **10** minutes (or the time limit specified by your Educators)
- utilise the submitted Slide deck

> You may also be required to answer any questions from your Educators and the audience

Code Requirements

No.	Requirement
R13	**Implement** features in the software development plan you have designed. You must utilise a range of programming concepts and structures using Ruby such as: - variables and variable scope - loops and conditional control structures - write and utilise simple functions - error handling - input and output - importing a Ruby Gem - using functions from a Ruby Gem
R14	**Apply** DRY (Don’t Repeat Yourself) coding principles to all code produced.

No.	Requirement
R15	**Apply** all style and conventions for the programming language consistently to all code produced.
R16	**Creates** an application which runs without error and has features that are consistent with the development plan.
R17	**Design** TWO tests which check that the application is running as expected. Each test should: - cover a different feature of the application - state what is being tested - provide at least TWO test cases and the expected results for each test case > An **outline** of the testing procedure and cases should be included with the source code of the application
R18	**Utilise** source control throughout the development of the application by: - making regular commits (at least 20 commits) with a commit message that summarises the changes - pushing all commits to a remote repository
R19	**Utilise** developer tools to facilitate the execution of the application: For example, - writing a script which turns the application into an executable; OR - packaging the application for use as a module or dependency

Submission

- This project must be submitted via **Canvas**
- Your submission is to be a single zip file (organised as described in the Deliverables section below)

File

Follow the steps below to correctly prepare file(s) to submit,

1. Create a **directory** called,

    ```
    {Fullname}_T1A2
    ```

    > Where,
    >
    > - `{Fullname}` is to be replaced by your Fullname

- T1A2 is this Assignment's ID.

2. **Organize** all the files for submission in this directory
   Follow the guidelines provided in the *Deliverables* section below
3. Create a **zip** file of this directory called,

   ```
   {Fullname}_T1A2.zip
   ```

   A sample command that can be run in Mac OS X/Linux terminal to achieve creation of the zip file is as follows,

   ```
   zip -r {Fullname}_T1A2.zip {Fullname}_T1A2
   ```

4. Submit this zip file in Canvas

Example

If your name is **`Luke Skywalker`**,

1. The directory you create will be called **`LukeSkywalker_T1A2`**

   ```
   mkdir LukeSkywalker_T1A2
   ```

2. Organize all the files for submission in this directory
3. Create a zip file of this directory. The zip file will be named **`LukeSkywalker_T1A2.zip`**

   ```
   zip -r LukeSkywalker_T1A2.zip LukeSkywalker_T1A2
   ```

4. Submit **`LukeSkywalker_T1A2.zip`** in Canvas

Deliverables

Following is a list of deliverables (which should cover all the above requirements) to include in your submission. The table also details how the information should be organized.

Deliverable	Description	Location/Folder (in Zip file)
`README.md`	General project documentation is to be compiled as a single markdown file named `README.md`.	/ (Root folder of your zip file)

Deliverable	Description	Location/Folder (in Zip file)
Resources	All files linked by the `README.md` file must be included in a folder named `docs`. All resources included in this folder must be in either `png`, `jpeg`, `pdf`, or markdown (`md`) format	**docs/**
Presentation	Slide deck of your presentation in pdf format	**ppt/**
Source Code	Source code for your entire project	**src/**

> Note: **All links to online material should have corresponding screenshots included in submission.**

Deadline

You are responsible for submitting this assessment before the deadline date/time. Your assessment `submission-time` is set and logged by Canvas once your entire submission is uploaded and submitted in the system. This `submission-time` is used to verify on-time submission or determine if you are liable for any late submission penalties.

Slow internet speeds, long submission/upload times, delay in uploading, etc are NOT grounds for special consideration (i.e. for waiving of any applicable late penalty).

> Remember for any valid special consideration request, relevant and full support documentation MUST be provided. The outcome of the special consideration request will be determined by Academic management and not your Educators.

Learning Outcomes and Criteria

Subject	Learning Outcome	Criterias
CMP1041 - Foundation Programming	CMP1041-1	`CMP1041-1.1` `CMP1041-1.2` `CMP1041-1.3`
	CMP1041-2	`CMP1041-2.1` `CMP1041-2.2` `CMP1041-2.3`
	CMP1041-3	`CMP1041-3.1` `CMP1041-3.2` `CMP1041-3.3`
	CMP1041-4	`CMP1041-4.1` `CMP1041-4.2` `CMP1041-4.3` `CMP1041-4.4`
	CMP1041-6	`CMP1041-6.1` `CMP1041-6.2` `CMP1041-6.3`

Subject	Learning Outcome	Criterias
PRG1002 - Programming I	PRG1002-3	`PRG1002-3.1` `PRG1002-3.2` `PRG1002-3.3`
	PRG1002-5	`PRG1002-5.1` `PRG1002-5.2` `PRG1002-5.3`

> Refer to the Term Academic overview for further details

Marks

Marks and/or results for this Assessment (as released in Canvas) are only raw marks and may not necessarily reflect final grades on transcripts. Grades are only finalized after review by the Academic Board and applicable processing (moderation, etc).

> The `*` symbol in the *rubric* below indicates only part of the specified requirement is being assessed in the applicable criteria

Glossary

Describe means to provide detail.
Identify means to make a statement or name something.
Explain means to provide detailed reasons/information relating to why or how something works

Key	Value
Version	2.1
Keywords	FastTrack; ftb; assessment