WEEK 1 – PRESENTATION AND DISCUSSION OF RESULTS (OVERVIEW)

The following will be covered this week:

A. Presentation and Discussion of Results

- Presentation of results based on project type
- Analysis of results based on project type
- The structure of the project report

PRESENTATION AND DISCUSSION OF RESULTS

PRESENTATION OF RESULTS BASED ON PROJECT TYPE

The approach you use to present the results of your research is dependent on the type of project you have chosen. Ultimately, you must select a suitable approach that adequately illustrates the results or findings that you have gathered.

Approach for presenting qualitative results

- Present results that support your literature review
- Develop methods to creatively and visually present results that are not in any numeric form

Approach for presenting quantitative results

- Present results that support your literature review
- Display results in the form of graphs, tables or charts

Approach for presenting results of successful software or project development

- Annotated screenshots of the different parts of the application or software (including sample input).
- A user manual that describes how the application can be used to show its interface and other visual aspects.

ANALYSIS OF RESULTS BASED ON PROJECT TYPE

Once you have decided the most suitable approach to present the results of your research, you must perform an analysis to inform and draw conclusions about the outputs.

Approach for analysing qualitative results

- Explain how the results support or challenge your original problem statement, project goals, objectives and learning outcomes.

Approach for analysing qualitative results

- Explain how the results support or challenge your original problem statement, project goals, objectives and learning outcomes.

Approach for presenting results of successful software or project development

- Explain if you have managed to meet the requirements that you specified in your research proposal.
- Describe if there were any problems that occurred during the implementation phase of the project.
- Describe the approach you used to address problems that occurred during the implementation phase of the project.
- Describe any workarounds or new techniques that were discovered when program code performed unintended actions.
- Describe the core functionality of the application and the different features that were implemented.
- Provide code snippets and or screenshots to explain and illustrate the functionality of the core elements of the application.

THE STRUCTURE OF THE PROJECT REPORT

In Week 2 of the COM-4005 module, you were given an outline of the structure for the final project report. This section will explain in detail the expected content that should be included in each section. Information about structuring your report will also be made available on the VLE in a separate document on the submission page for the project report and presentation.

Each section of the report should be consistently styled and include page numbers, headers and footers in accordance with APA referencing style.

A. Title page

- Title of project
- University logo
- Author
- Name of university degree programme and module code
- Date/Year

B. Acknowledgements

- A note from the author
  - Information about contributing parties or inspirations for project

C. Abstract

- A short but concise summary of the project
  - The word count of the abstract should not exceed 250 words

D. Table of Contents

- The sections of the table of contents page(s) should be labelled and numbered to match the sections in the report that follows

E. Introduction

- Provide a brief introduction to the project

- This information can be obtained from the approved project proposal, but must outline the subject area, the research problem (problem statement), and the goals, objectives and learning outcomes for the project.

F. Executive Summary

- A concise summary of the entire project

G. Literature Review

- An overview of the relevant information about the research conducted in the projects subject area including;
  - Acquired literature on theoretical aspects of the subject area using primary and secondary resources
  - Collection of qualitative and quantitative data that supports the research problem

H. Methodology

- The method and resources used to conduct research including;
  - The different tools used and how they were used
  - Participant involvement and its level of contribution to the project
  - Project management methodologies or tools used

I. Presentation of results

- Illustrate the raw results of the research using;
  - Graphs, charts and tables that display numerical based data
  - Visualisations that display non numerical based data
  - Screenshots and illustrations of output from software applications

J. Discussion of results

- Describe and interpret the results obtained and if they address the problem statement, research goals, objectives and learning outcomes

K. Conclusion

- Provide an overview of the success of the project and make suggestions on improvements that can be made in the future.

L. References/Bibliography

- List of cited sources in APA referencing format
- Reading list of referenced works in APA referencing format

M. Appendices/Tables/Figures

- Information that does not need to be included directly in the project report such as;
  - User guides
  - Programming code snippets
  - Other relevant illustrations (Graziano & Raulin, 2014; Leedy & Ormrod, 2015)

APPENDIX A – GUIDE TO PRESENTING RESEARCH RESULTS

The following section will describe in detail the different ways that you can present your research results based on the project type you have chosen. The project type, as described in the COM-4005 module can be heavily focused either **on research or on software engineering**. This section will guide you through approaches that can be used in each scenario.

***Note** – this Guide may overlap some of the content that was covered in the earlier sections of the COM-4005 module, nevertheless it is important to know how to document the development and completion of your final project appropriately and any future projects you wish to undertake.*

SOFTWARE ENGINEERING OR WEB DEVELOPMENT PROJECTS

If you have chosen a project that involves any type of software engineering or web development, you will need to add technical documentation that can be used to describe the full development process of your software or application from planning to implementation and usage. As you may have already been documenting your project work, you should already be able to place descriptions of your work in the appropriate sections of this technical document.

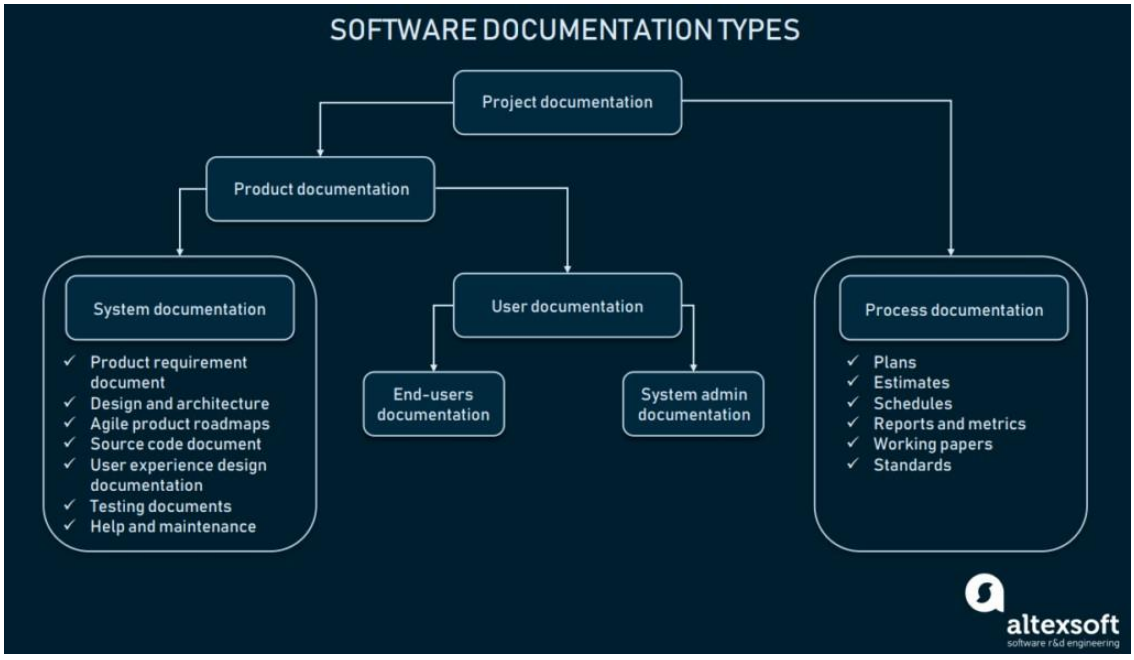


Figure 1 - Software documentation types most commonly used in Agile projects (Altexsoft.com)

The following is a list of recommended sections that you may wish to include in your documentation or as separate sections of documentation.

System documentation

- **README file**: overview of software project and additional relevant information
- **Contributing:** list of additional developers or other contributors to the project
- **License:** software licensing and usage details (if any)
- **Requirements:** business requirements and description of software functionality
- **Design and Architecture**: software architectural design diagrams and descriptions
- **Source Code**: code definitions, explanations and examples using snippets
- **UX and UI design:** wireframes, mock-ups or sample use case to for demonstrating user experience (UX) and user interaction (UI) design.
- **Testing documentation:** specific results analysis from user testing phase

User documentation (UX and UI)

- User manuals, help and troubleshooting guides for **end users**
- User manuals, help and troubleshooting guides for **system or software**

administratorsThe importance of documenting your project

- Effectively document your process of software development from planning to implementation
- Keep track of project milestones and difficulties faced during different phases of the project
- Allow potential contribution or future collaboration with other developers (making use of project hosting on platforms such as GitHub)
- As your project will address a gap in research, an improvement in a particular process or software implementation, documenting your project will illustrate a unique contribution to the field of computer science and the development community

Hosting your project and documentation on GitHub

"*GitHub is the developer company. We make it easier for developers to be developers: to work together, to solve challenging problems, to create the world's most important technologies. We foster a collaborative community that can come together – as individuals and in teams – to create the future of software and make a difference in the world."* (GitHub.com)

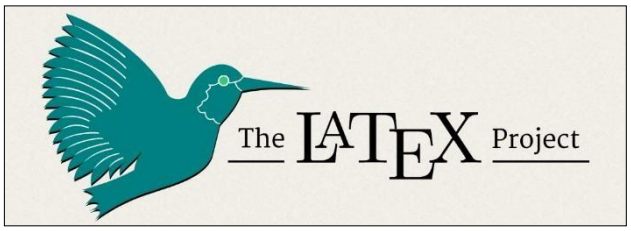
- GitHub is an online platform that can be used to develop and maintain your projects source code
- The free GitHub plan will allow you to create unlimited public and private repositories for different types of projects
- You can publish your project on GitHub privately or publically
    - Allow other members of the development community to comment, report issues, make suggestions on improvements, request new features
    - Allow other developers to collaborate and contribute to your project.
    - Create documentation and customize pages
    - Store and host project files for distribution and as personal backups
- GitHub also provides a range of useful features for project management and creating documentation for your project

Essential software tools for writing technical documentation

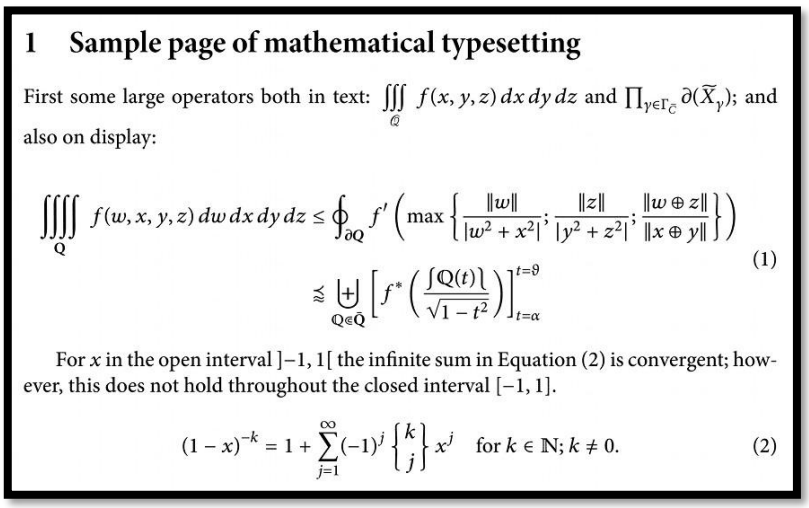
The following section will describe in detail some software applications and tools that were created specifically to aid in the documentation of software engineering or development projects. You will find several different software applications and their associated programming language and from here you can choose to use one which you prefer.

Note – software such as LaTex may have a steep learning curve. If you find yourself spending more time learning how to use it rather than documenting, choose a language specific alternative.

LaTex – General-purpose software for technical and scientific documentation



LaTex is a popular documentation software for writing scientific research papers. In comparison to a word processor such as Microsoft Word or LibreOffice Writer, LaTex specializes in dealing with text formatting for scientific calculations and source code for software development using TypeSetting.



1 Sample page of mathematical typesetting

First some large operators both in text: $\iiint_{\mathbb{Q}} f(x,y,z)\,dx\,dy\,dz$ and $\prod_{\gamma\in\Gamma_{\widetilde{C}}}\partial(\widetilde{X}_\gamma)$; and also on display:

$$\iiint_{\mathrm{Q}} f(w,x,y,z)\,dw\,dx\,dy\,dz \leq \oint_{\partial\mathrm{Q}} f'\left(\max\left\{\frac{\|w\|}{|w^2+x^2|};\frac{\|z\|}{|y^2+z^2|};\frac{\|w\oplus z\|}{\|x\oplus y\|}\right\}\right) \lessapprox \biguplus_{\mathbb{Q}\in\bar{\mathbb{Q}}}\left[f^*\left(\frac{\int\mathbb{Q}(t)\rfloor}{\sqrt{1-t^2}}\right)\right]_{t=\alpha}^{t=\vartheta} \tag{1}$$

For x in the open interval $]-1,1[$ the infinite sum in Equation (2) is convergent; however, this does not hold throughout the closed interval $[-1,1]$.

$$(1-x)^{-k} = 1+\sum_{j=1}^{\infty}(-1)^j \left\{ {k \atop j} \right\} x^j \quad \text{for } k\in\mathbb{N}; k\neq 0. \tag{2}$$

Figure 2 - LaTex mathematical typesetting example

Achieving the same results in the previous image on a word processor would be very tedious. You can use the LaTex software to document the whole project or simple use it to copy and paste source code snippets into your documents.

With verbatimbox, you can tell the macro to make the code "footnotesize."

```
Here is my input code text
Line two of my $%& code text
Done now.
```

If it is from a file, I can do that, too, this time \scriptsize:

```
\documentclass{article}
\usepackage{verbatimbox}
\usepackage{graphicx}
\parskip 1ex
\parindent 0in
\begin{document}
With verbatimbox, you can tell the macro to make the code ‘‘footnotesize.’’\par
\begin{verbbox}[\footnotesize]
Here is my input code text
Line two of my $%& code text
Done now.
\end{verbbox}
\fbox{\theverbbox}\par
If it is from a file, I can do that, too, this time \verb|\scriptsize|:\par
\verbfilebox[\scriptsize]{junk.tex}
\fbox{\theverbbox}\par
Alternately, you could shrink it further with a \verb|\scalebox|:\par
\fbox{\scalebox{.5}{\theverbbox}}\par
If you have need to break across pages, the package offers ‘‘nobox’’ versions
of some of its macros.
\end{document}
```

Figure 3 - Source code formatting options in LaTex

3.3 Formatting source code

minted Using minted is straightforward. For example, to highlight some Python source code we might use the following code snippet (result on the right):

```
\begin{minted}{python}
def boring(args = None):
    pass
\end{minted}
```

```
def boring(args = None):
    pass
```

Optionally, the environment accepts a number of options in key=value notation, which are described in more detail below.

\mint For a single line of source code, you can alternatively use a shorthand notation:

```
\mint{python}|import this|
```

```
import this
```

This typesets a single line of code using a command rather than an environment, so it saves a little typing, but its output is equivalent to that of the minted environment.

7

Figure 4 - Documentation for minted package used for syntax highlighting in LaTex

Documentation software for the Python programming language

- **Sphinx**

  - Written in python and first created for Python documentation
  - Can also be used for documenting projects in other languages
  - Compatible with Linux, Windows and macOS

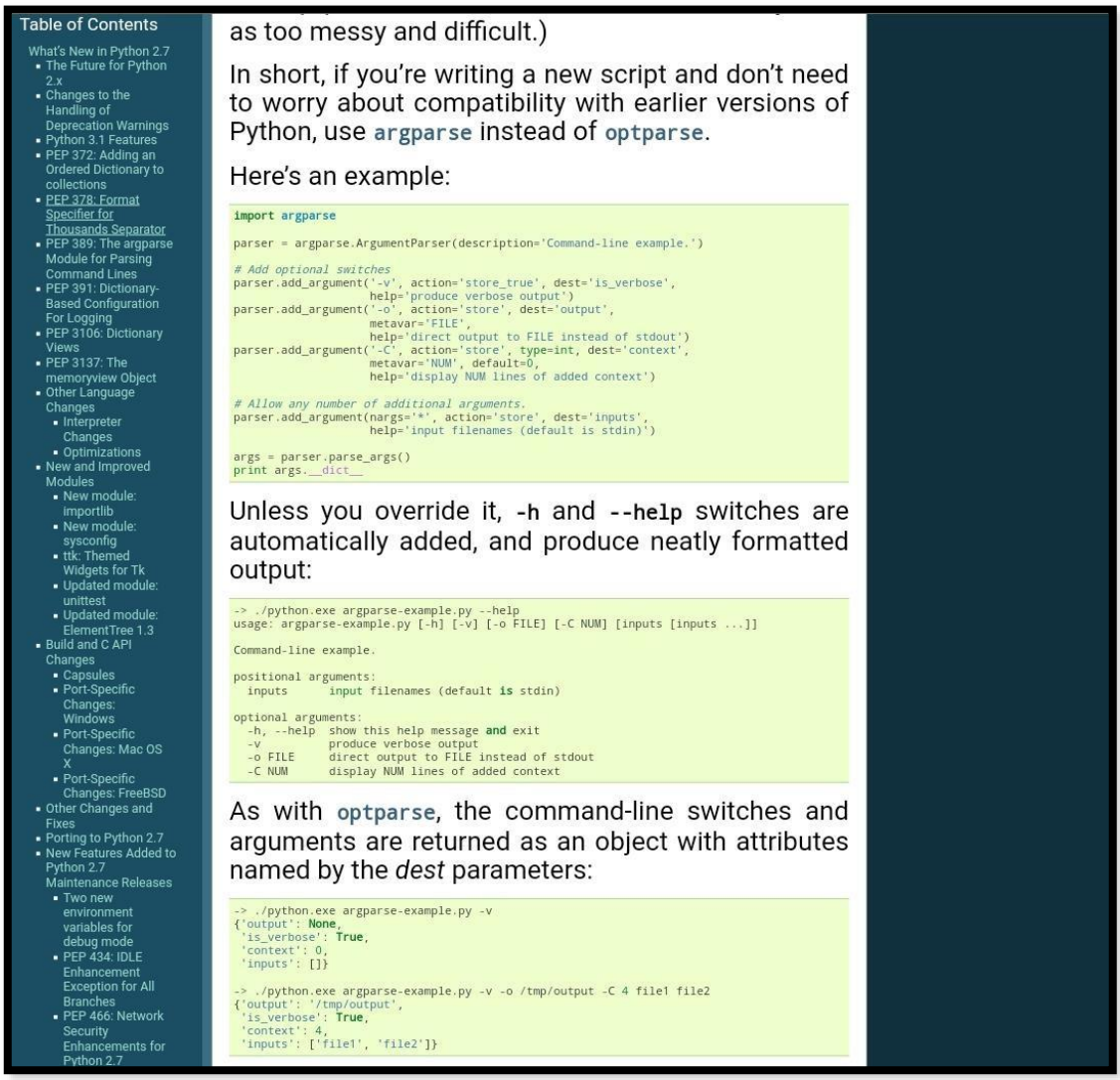


Figure 5 - Python 2.7 documentation created using Sphinx

Documentation software for the R programming language

- **R Markdown**

  - Official documentation software for R projects developed by R Studio
  - Can be used to document non-R projects including Python, SQL, Bash, Rcpp, Stan, JavaScript and CSS

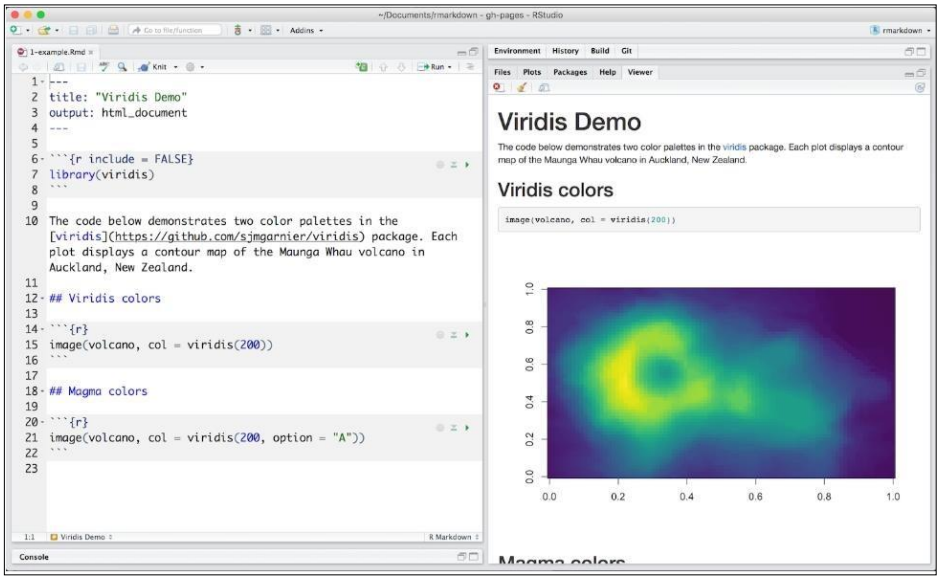


Figure 6 - Creating documentation for R projects

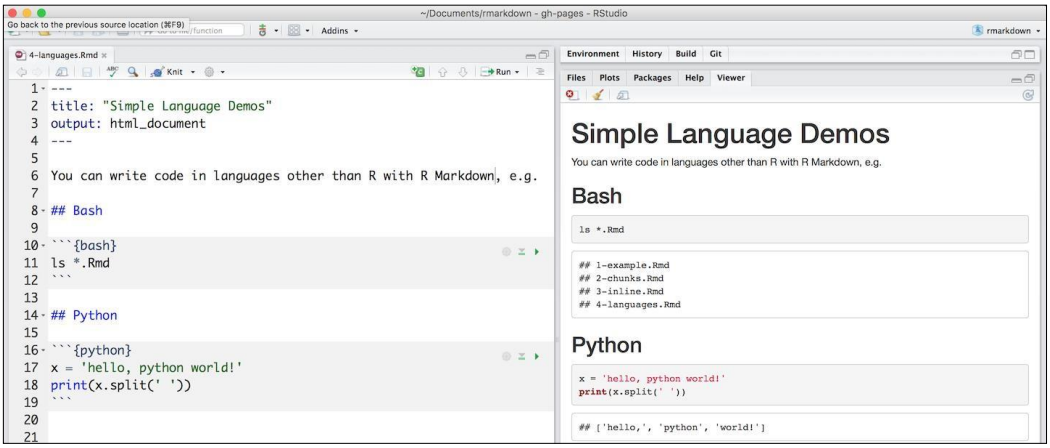
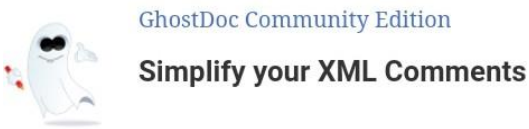


Figure 7 - R Markdown documentation using Python and Bash notation

Documentation software for the C++ programming language

- **GhostDoc**



  - Documentation extension for the **Visual studio IDE**
  - Can also be used to document C#, Visual Basic and JavaScript projects
  - **GhostDoc Pro** is a paid version with comprehensive and advanced documentation features not available in the free Community Edition
  - **GhostDoc Community Edition** is free but contains less documentation features than the GhostDoc Pro version

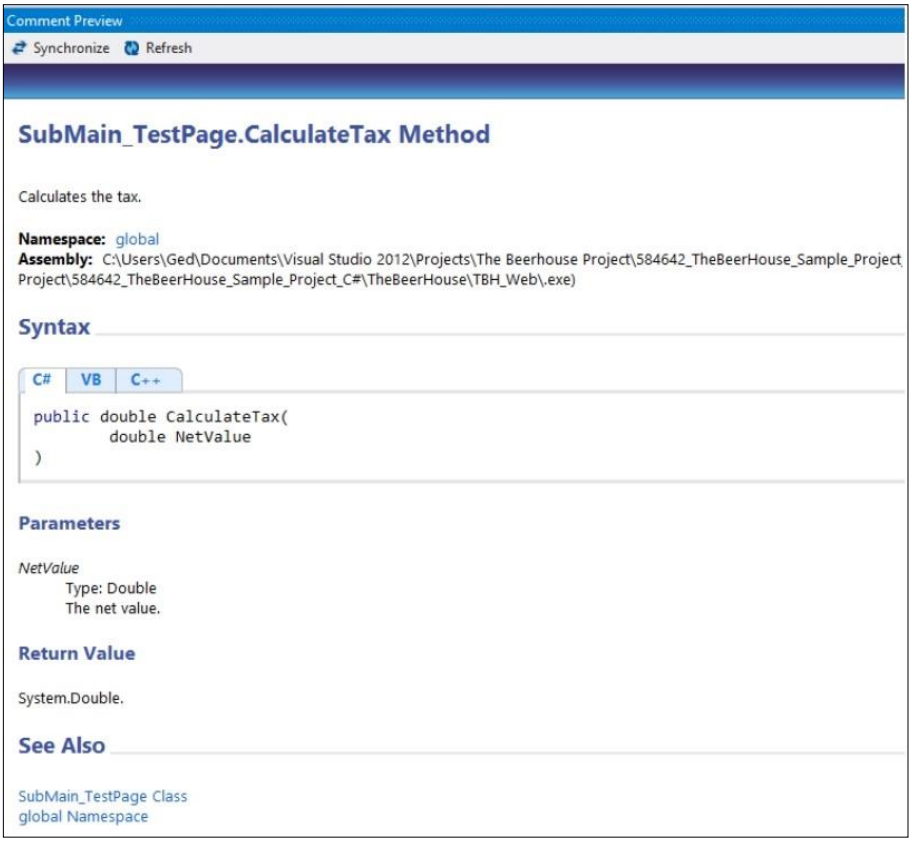
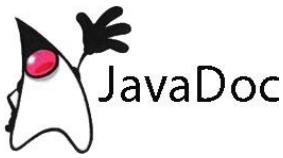


Figure 8 - Comment Preview with Submain GhostDoc

Documentation software for the JAVA programming language

- **Javadoc**



  - Official documentation generator for Oracle's Java programming language
  - Included as a tool in the JDK
  - Generates Java documentation in HTML format

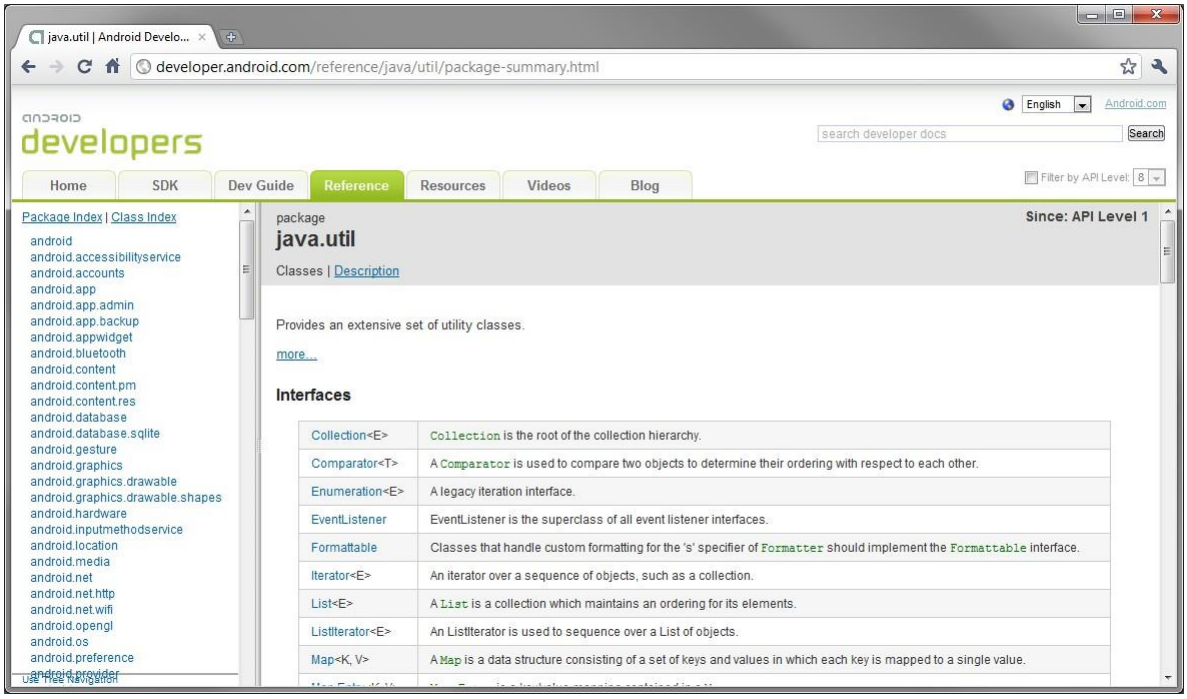


Figure 9 - Generated Java Documentation sample (Android Developers)

Documentation software for Web Development (HTML, CSS, JavaScript)

- **iAWriter**

  - Document editor with features such as syntax highlighting and Markdown formatting for code snippets and TeX math expressions

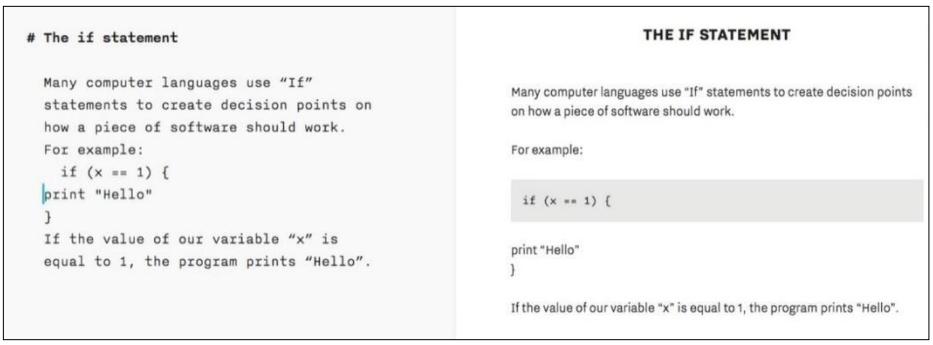


Figure 10 - iA Writer Code block output (right)

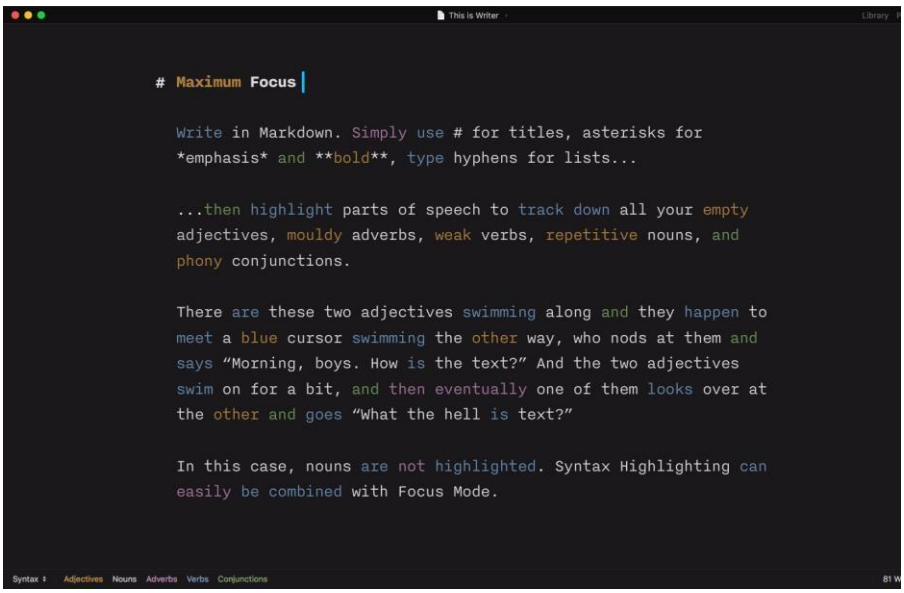


Figure 11 - Custom syntax highlighting in iA Writer

PRESENTING RESULTS FOR RESEARCH HEAVY PROJECTS

In general, you should make use of the Microsoft Office Suite specifically **MS Word** and **MS Excel** for presenting research results. Similarly, you may use **LibreOffice** alternatives **Writer** and **Calc**.

Guidelines for formatting tables, graphs and mathematic formulas

- If you are **tallying information such as survey results or other statistics**, you should create a table in **MS Excel** then generate the relevant graph or chart

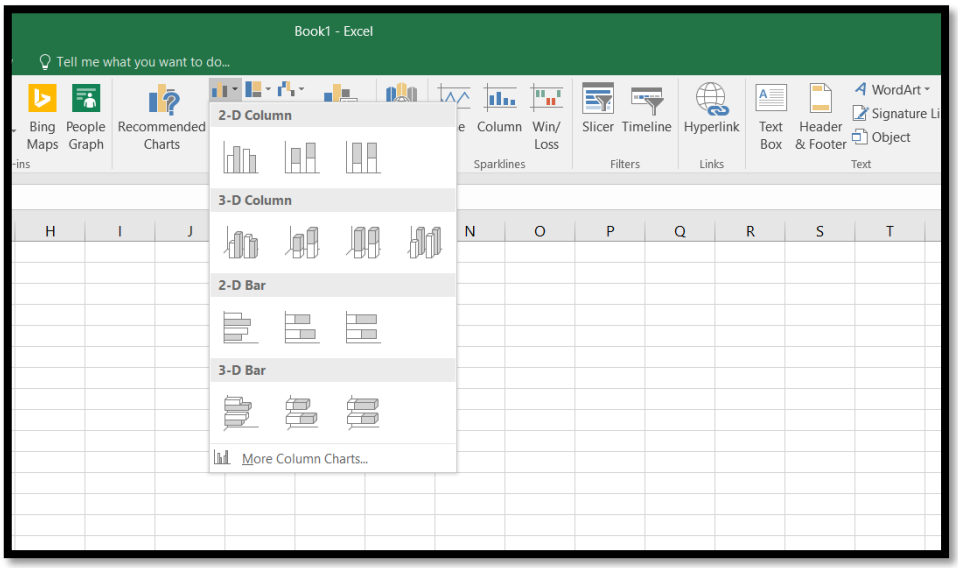
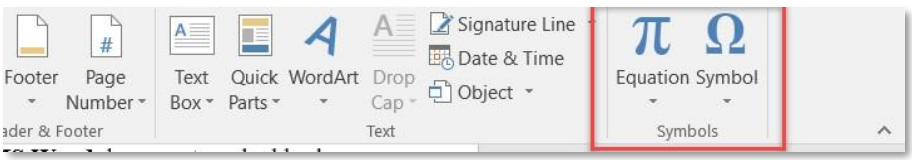


Figure 12 - Visualisation of Results in MS Excel using graphs and charts

  - You will be able to copy and paste the graph or chart directly in the **MS Word** document then add relevant annotations or descriptions.

- If you need to display any mathematical formulas make use of the **Equation** and **Symbol** menus located on the **Insert** Ribbon in MS Word



- Alternatively you may use **LibreOffice Math** which is an advanced formula editor

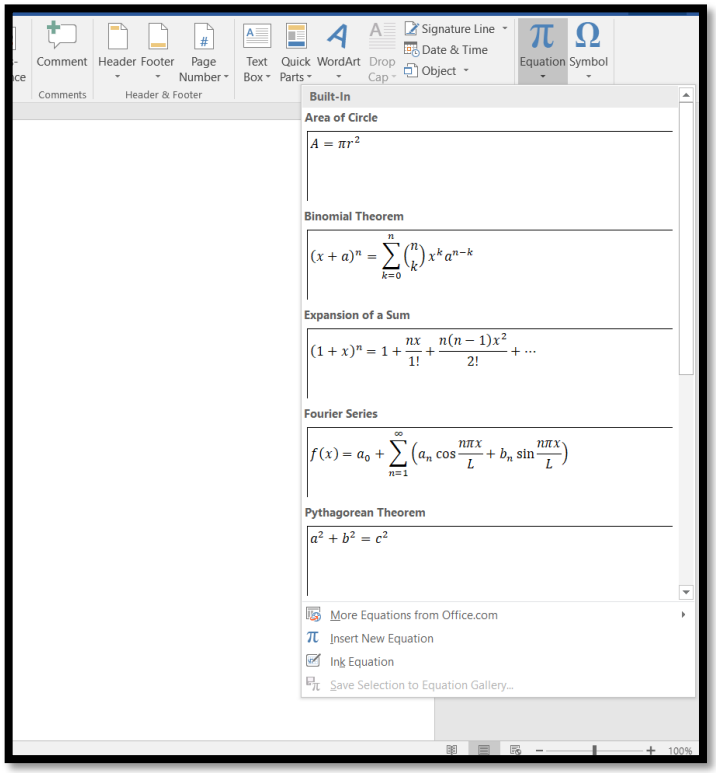


Figure 13 - Inserting Equations and Symbols in MS Word

GUIDELINES FOR FORMATTING CODE SNIPPETS AND SOFTWARE SCREENSHOTS IN MS WORD

If you have decided not to use documentation software for your project, you will still be able to include code snippets and images of a working software application by following the guidelines below.

Make use of Screenshot software or tools included on your operating system

- On a **Windows** system
  - Use the **Print Screen or prt sc button** on your keyboard to take a screenshot
    - Paste the screenshot directly from the clipboard in your document using the **Ctrl-V** keyboard shortcut.
    - You should then crop the image, resize it and add a descriptive caption
  - Use the **Snipping Tool** program to take screenshots
    - Crop the image and add descriptions and annotations directly on them

- Save the images then copy and paste them in your Word document or alternative word processing software

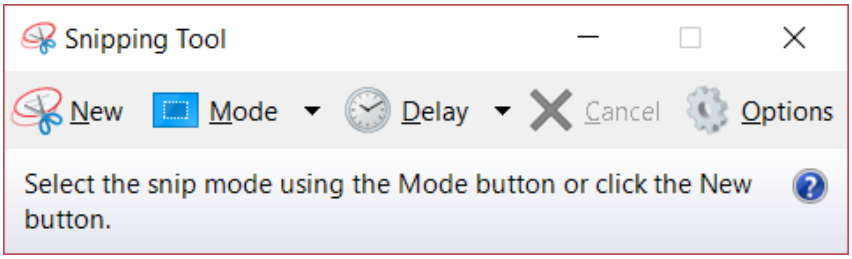


Figure 14 - Windows Snipping Tool (for taking screenshots or screen grabs)

- On a **macOS** system
  - Use the **Shift-Command-4** shortcut to start using the screenshot application*
    - Follow similar steps detailed for Windows based systems to edit, save and paste images into your document
      - ***Note** - keyboard shortcut may differ depending on OS version*
- On a **Linux** based system (Ubuntu)
  - Make use of the screenshot software called **Screenshot** in Ubuntu 18.04 LTS
    - Follow similar steps detailed for Windows based operating system to edit, save and paste images into your document

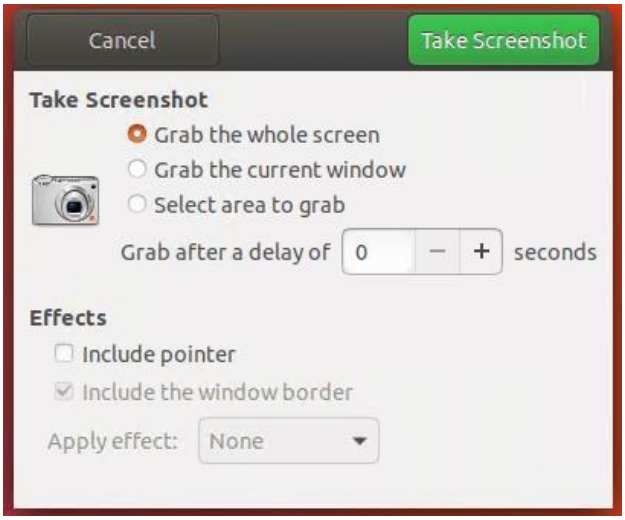


Figure 15 - Screenshot in Ubuntu (Bionic Beaver - 18.04 LTS)

- On **Android** and **iOS** systems
    - Make use of the shortcuts on Android and iOS systems to take screenshots of running mobile application
        - For **Android** – hold down the **Power button** and the **Volume Down** button*
            - Edit, save and transfer the screenshot to your document
        - For **iPhones –** hold down the **Power** and press the **Home buttons** together*
            - Edit, save and transfer the screenshot to your document
        - **** Note*** *– instructions may differ depending on OS version*

Additional guidelines

- Avoid including screenshots that are not relevant
- Ensure that images you will use are in their highest resolution or quality possible and scale down when possible
- Ensure any resized images have been resized without affecting their aspect ratios
- Where possible, use annotations directly on an image or as a detailed caption under an image if the relevant parts of code is not clear

APPENDIX B – BMC CLOUD LIFECYCLE MANAGEMENT SOFTWARE (CASE STUDY)

In this example, we will take an in depth look at how documentation was setup for the **BMC Cloud LifeCycle Management 4.6** software. By the end of this case study, you should be able to develop an awareness of how documentation can be set up to compile and present the results of a software engineering project.

The documentation for BMC's Lifecycle Management software is set up using the structure shown on the left side of the image below.

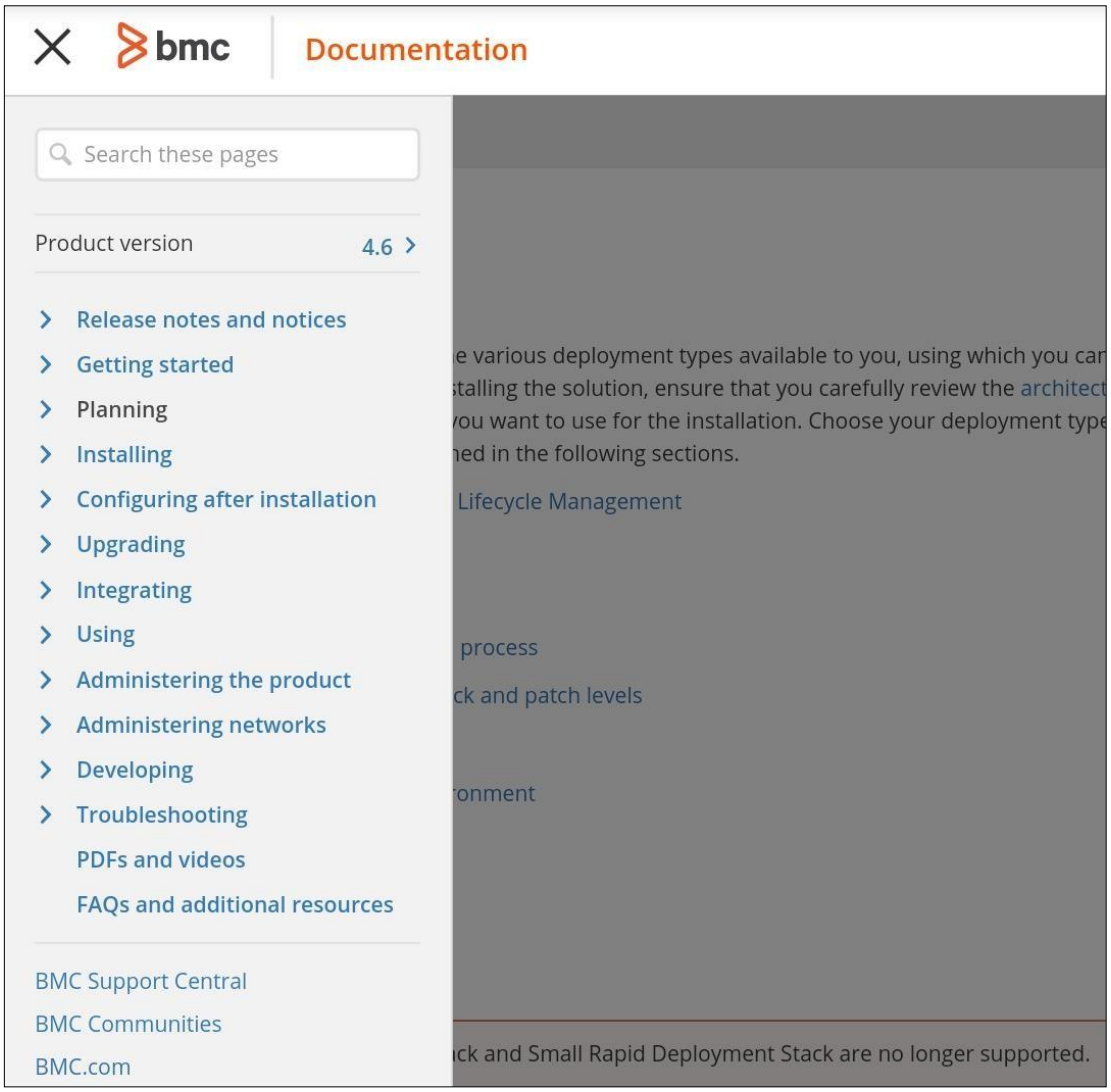


Figure 16 - BMC Cloud Lifecycle Management Software documentation (Table of Contents)

As can be seen in the previous image, the structure of the documentation follows the general guidelines outlined in **Appendix A**. We can now look inside each section to get an overview of what the company has decided to include in their documentation. You will then be given, recommendations on how you can use a similar structure and approach for writing your own software documentation.

Release notes and notices

- This section gives information about the current version (release) of the software. This can either be information about a final version of the software or beta and alpha versions. The information included is usually related to fixes or patches included in current versions, a list of features, instructions for the user and general technical information and installation guidelines.

Getting started

- This section provides a very brief guide for users who wish to use the software. It is important that a getting started guide provide non-detailed descriptions of how to use the software. This is important to help new users or users unfamiliar with the version of software get an overview of its core functionality. If a user requires detailed information about specific functionality, the getting started guide should lead them to a separate area in the help or troubleshooting documentation.

Planning

- In this section, BMC described deployment instructions for their software. This section is of key importance as it describes deployment guidelines for operating system types, deployment scale, language and licensing requirements and customized configuration instructions.

- ***NOTE -*** *For the COM 4005 and COM 2006 projects, you will just need to focus on providing instructions for installation as described below*

Installing

- This section will provide installation instructions for the user, which is essential for any software or application
- Prior to providing the installation instructions, you will need to include a list of compatible operating systems and recommended hardware specifications needed to run the software. In addition, you will need to include installation instructions for all operating systems or versions that your software can run on. The following is a general list of the types of information;
    - Supported operating systems and versions (i.e. Windows ,7-10, macOS Mavericks-High Sierra)
    - Compatible operating system processor architecture (i.e. 32-bit, 64-bit)
    - Minimum hardware requirements (disk space, processor speed and type, graphic and input devices)
- After all the specifications have been listed, create an installation guide for each compatible operating system using a variety of bullet points and screenshots for user friendliness

Configuration after installation

- This section will provide information on how the user can configure the software after it has been installed. This information may include setting up environmental variables, file paths, user directories or simply instructions on how to customize and tweak the user interface and experience (i.e. layouts, colours, and control and navigations schemes).

Upgrading

- This section is useful if you have an older working version of the software that has been updated or patched. The upgrading instructions may be specific to the operating system or

could be generic instructions on how to update an old installation to the latest version. Whichever the case may be, insure that the instructions are clear for the user.

Integrating

- This section is specific to software that can be integrated with third party tools or libraries. Ensure that you have highlighted the different ways in which the software can be integrated with other tools, libraries or third party software. In addition, ensure that you provide a list of compatible tools and provide some practical examples.

Using

- In this section, you will provide the full details on how to use the software. Ensure to provide separate guides for user roles where necessary (i.e. admin guide and general user guide). The written documentation should be detailed enough and have concise language and formatting. It is essential that you also provide visual documentation in the form of screenshots or video for a more interactive experience for potential users.

Administering the product

- This section will provide specific information to users that are given administration access to manage roles and permissions as well as access to modify user restricted controls and features of a software program.

Administering networks

- In this section, BMC has described specific information concerning the administering of their software over a network using custom tools and automation features. As this case is highly specific to BMC, documentation you write that is similar should address your software's specific use case or scenario.

Developing

- In this section, BMC has described information for developers that wish to integrate or customize their system using the BMC Cloud Lifecycle Management software's API (Application Programming Interface) and SDK's (Software Development Kits). If your software has implemented an API or you have developed an SDK, this could be useful documentation to describe how other developers can implement components of your software into their projects.

Troubleshooting

- This section is one of the most important ones to include which requires concise documentation for the **end-user** and potential **software administrators**. During the testing phase, you will likely have encountered bugs or issues that may affect certain systems or hardware configurations. Since you cannot account for full fixes for all types of systems or hardware configurations, the troubleshooting documentation can be used to provide guidelines on dealing with common issues that can be fixed by the user themselves. Such information can include;
    - Providing clear error messages
        - This can help user self-diagnose issues
    - Providing a method of logging or generating a log file
        - This can help users submit error logs to help software provider or community members help understand and fix common issues
    - Providing answers to common errors using a FAQ (Frequently Asked Questions)
        - Keep track of commonly occurring errors you faced in your software development and implementation processes
        - Address common installation issues and solutions (i.e. being unable to install or modify software due to permissions or privileges)
    - Provide users a way to contact support

- Include additional documentation, videos and links to additional third party resources or communities where relevant troubleshooting help can be found.

Additional recommendations based on the BMC case study

- The documentation BMC has provided is for a fully functional software program so they have omitted any specific requirements, planning, implementation and testing documentation, as it is not very useful for the end-user. Be sure to include this documentation for your project to outline all the relevant stages.

FURTHER READING

Graziano, A.M., Raulin, M.L. (2014). *Research Methods A Process of Inquiry*. Essex: Pearson Education Limited

Leedy, P.D. & Ormrod, J.E. (2015). *Practical Research Planning and Design (11th Edition)*. Essex: Pearson Education Limited

Technical Documentation in Software Development: Types, Best Practices, and Tools. (n.d.). Retrieved April 12, 2019 from https://www.altexsoft.com/blog/business/technical-documentation-in-software-development-types-best-practices-and-tools/