INFO1111: Computing 1A Professionalism

2024 Semester 1

Skills: Team Project Report

Submission number: 3

Github link: https://github.sydney.edu.au/INFO1111-2024/CC25-2

Team Members:

Name	Student ID	Target * Foundation	Target * Advanced	Selected Major
RANN, Callum	540698143	A	NA	Computer Science
MAI, ZIXI	540246807	A	NA	Data Science
ZHONG, QILONG	540238846	S	A	SW Development
Li, Senhao	540220209	A	NA	Cyber Security

* Use the following codes:

- NA = Not attempting in this submission
- A = Attempting (not previously attempting)
- AW = Attempting (achieved weak in a previous submission)
- AG = Attempting (achieved good in a previous submission)
- S = Already achieved strong in a previous submission

Contents

1. Task 1 (Foundation): Core Skills 2
    - 1.1. Skills for Computer Science: RANN, Callum 3
    - 1.2. Skills for Data Science: MAI, ZIXI 5
    - 1.3. Skills for SW Development: ZHONG, QILONG 6
    - 1.4. Skills for Cyber Security: Li, Senhao 8
2. Task 2 (Advanced): Advanced Skills 11
    - 2.1. Tools and Skills for Computer Science: RANN, Callum 14
    - 2.2. Tools and Skills for Data Science: MAI, ZIXI 23
    - 2.3. Tools and Skills for SW Development: ZHONG, QILONG 34
    - 2.4. Tools and Skills for Cyber Security: Li, Senhao 49
3. Submission contribution overview 50
    - 3.1. Submission 1 contribution overview 50
    - 3.2. Submission 2 contribution overview 50
    - 3.3. Submission 3 contribution overview 50

1. Task 1 (Foundation): Core Skills

Throughout your Computing degree we will help you learn a range of new skills. Once you graduate however you will need to continue to learn new languages, new tools, new applications, etc. Task 1 focuses on core technical skills (related to LaTeX and Git) and the key technical skills used in different computing jobs. Each member of the team should individually complete their subsection below. You should begin by allocating to each team member a different major to focus on (i.e. one of: Computer Science; Data Science; Software Development; Cyber Security). If you have a fifth member, then your tutor will suggest a fifth topic to cover. This allocation should be specified above (see lines 37-56 in the LaTeX file).

For this section each member of your team needs to select one of the majors provided and identify 3 key technical skills that you would need to be able to work in the industry of your allocated major. You should then put these in order from most required to least required, and for each one explain why it is a key skill required for the industry of your major. You must use the skills framework for the information age ”SFIA” to identify at least 2 out of the 3 key tech skills. (Target = ∼100 words per skill = ∼300 words total, per student).

Begin by looking at the list of skills identified within SFIA (Skills Framework for the Information Age) [1]. Then select two skills from the complete list. The skills you select should be skills you believe are the most required key technical skills relevant to the major you have selected. You should explain why each skill is a key technical skill and necessary for that major.

You will need to integrate your information into the shared collaborative LaTeX document and compile the result.

OVERALL REQUIREMENTS:
To achieve an ”OK” rating for this task you must individually accomplish the following:

- Each member of your team **has been** allocated a different major (Computer Science, Data Science, Software Development, Cyber Security).
- Each member of your team **has identified** 3 key technical skills that you would need to be able to work in the industry of your allocated major.
  - These must be in order from most required to least required.
  - Each skill must have an explanation on why it is a key skill required for the industry of the major (∼100 words per skill).
  - At least 2 out of the 3 key tech skills must be identified from the skills framework for the information age SFIA.
- Github, LaTeX & LaTeX
  - Your team has created a team repository on Github for the project and put a copy of the LaTeX template, bib file, and image file into the team repository (only needs to be done by one member of your team).
  - The information for ‘Task 1’ has been compiled into the shared collaborative LaTeX document using the template provided on Canvas with your team members sections - you have edited the LaTeX template to include your chosen major and the 3 key tech skills for the major.
  - You have cloned the team repository to your local machine.

  - Provide evidence that you can compile from the command line (provide screenshots of the command entered and output).
  - Provide evidence that you can commit to your local repo (provide screenshots of the steps taken to commit to their local repo).

- Referencing
  - You have provided in-text references (IEEE) to support your claims or where they gathered the information from.
  - You have a reference list following the IEEE referencing guidelines.
  - Some common things to look for to see whether your have correctly followed the referencing guide are:
    - Sources are listed in alphabetical order
    - The sources you have listed are only the sources that are present in-text.
    - All sources seen in-text are included in the reference list.
    - You followed the correct convention for references that don't have author's details or multiple sources have the same author and year of publication
    - You have included the required information for the source type as outlined in the guide.
    - Sources are not a list (i.e. dotpoints)

To achieve a "STRONG" rating, you must individually accomplish all of the above in addition to the following:
Demonstrated the following to your tutor during the tutorial:

- You are able to retrieve your team's shared repo
- You are able to make changes, recompile, commit changes, and push back to repo.
- Note: you should also provide screen-shots of relevant actions taken to make changes, recompile etc. does not require you to provide evidence of detailing conflicts.

1.1. Skills for Computer Science: RANN, Callum

Three key technical skills that are needed to work in the computer science industry:

1. Numerical analysis Numerical analysis is the "area of mathematics and computer science that creates, analyzes, and implements algorithms for solving numerically the problems of continuous mathematics." [2]. In essence, solving mathematical problems that cannot be solved analytically or symbolically, often due to their complexity or the lack of closed-form solutions. It is no doubt that the world is growing steadily more complex, resulting in "numerical analysis of increasing sophistication being needed to solve these more accurate and complex mathematical models of the world" [3].

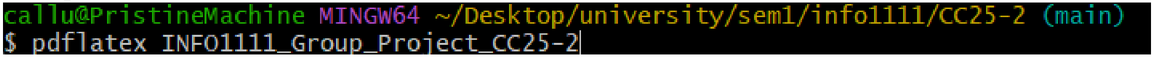
2. Computational modelling As stated in [4], proficiency in computational modeling is crucial for creating accurate representations of complex systems using mathematics, physics, and computer science. Skilled individuals can adjust variables within models to simulate various scenarios and analyze outcomes, enabling virtual experiments and hypothesis exploration. System thinking and the Turing pattern [5] emphasize the necessity of computational modelling, wherein minor adjustments to a basic formula can spawn an incomprehensibly extensive array of patterns. It is no doubt computational modeling skills accelerate the discovery and innovation process across diverse disciplines such as engineering, biology, and all other sciences.

3. High performance computing (HPC) HPC is ”technology that uses clusters of powerful processors that work in parallel to process massive multi-dimensional data sets” [6]. As [7] walks through, a well-known HPC solution is the supercomputer. Which displays the ability to solve more abstract problems than ever before, allowing for groundbreaking discoveries to be made. It also majorly assists with larger data sets, which when quickly processed will allow for innovations in technologies like the Internet of Things, artificial intelligence, and 3-D imaging. Although it is extremely useful in progressing future technologies, it is not as fundamental in doing so as the previous two key techniques.

PROOF OF GIT SKILLS

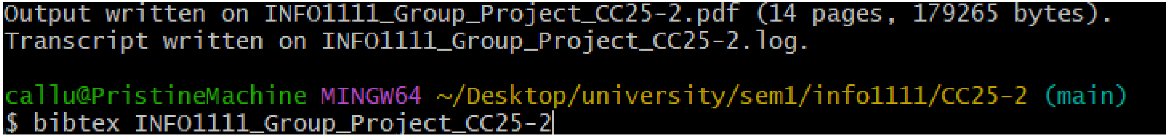
Provide evidence that you can compile from the command line.

Compile command 1:



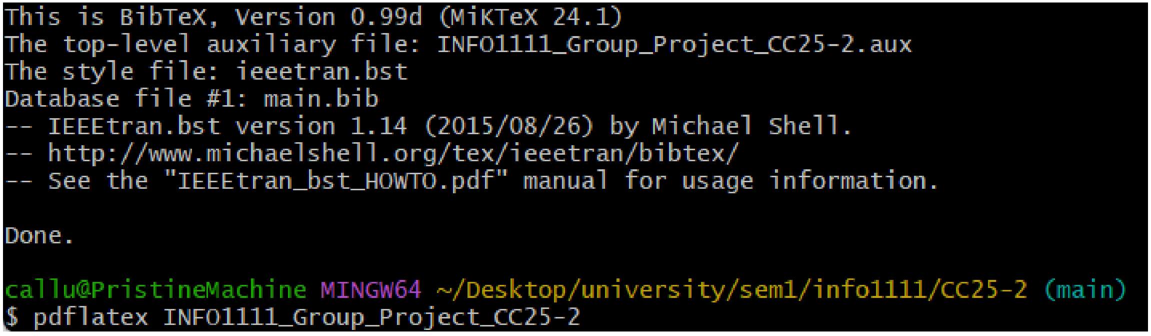
```
callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/CC25-2 (main)
$ pdflatex INFO1111_Group_Project_CC25-2
```

Output from 1 + Compile command 2:



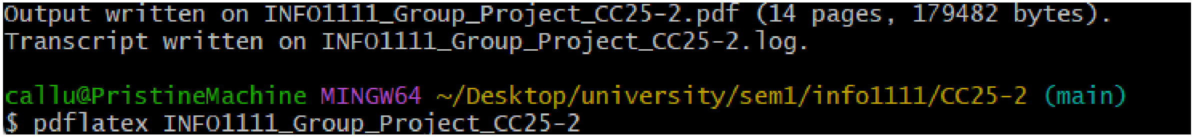
```
Output written on INFO1111_Group_Project_CC25-2.pdf (14 pages, 179265 bytes).
Transcript written on INFO1111_Group_Project_CC25-2.log.

callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/CC25-2 (main)
$ bibtex INFO1111_Group_Project_CC25-2
```

Output from 2 + Compile command 3:



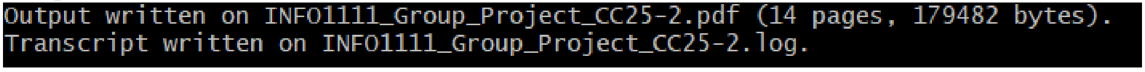
```
This is BibTeX, Version 0.99d (MiKTeX 24.1)
The top-level auxiliary file: INFO1111_Group_Project_CC25-2.aux
The style file: ieeetran.bst
Database file #1: main.bib
-- IEEEtran.bst version 1.14 (2015/08/26) by Michael Shell.
-- http://www.michaelshell.org/tex/ieeetran/bibtex/
-- See the "IEEEtran_bst_HOWTO.pdf" manual for usage information.

Done.

callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/CC25-2 (main)
$ pdflatex INFO1111_Group_Project_CC25-2
```

Output from 3 + Compile command 3:



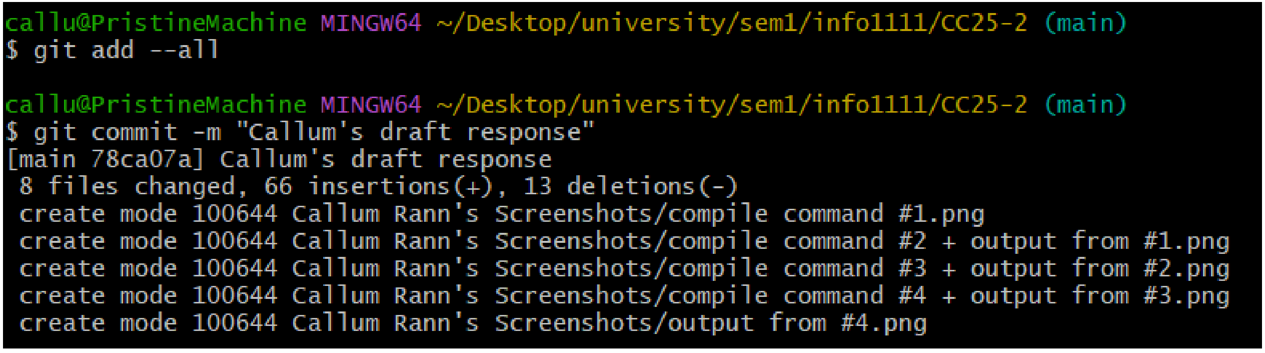
```
Output written on INFO1111_Group_Project_CC25-2.pdf (14 pages, 179482 bytes).
Transcript written on INFO1111_Group_Project_CC25-2.log.

callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/CC25-2 (main)
$ pdflatex INFO1111_Group_Project_CC25-2
```

Output from 4:



```
Output written on INFO1111_Group_Project_CC25-2.pdf (14 pages, 179482 bytes).
Transcript written on INFO1111_Group_Project_CC25-2.log.
```

Provide evidence that you can commit to your local repo.



```
callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/CC25-2 (main)
$ git add --all

callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/CC25-2 (main)
$ git commit -m "Callum's draft response"
[main 78ca07a] Callum's draft response
 8 files changed, 66 insertions(+), 13 deletions(-)
 create mode 100644 Callum Rann's Screenshots/compile command #1.png
 create mode 100644 Callum Rann's Screenshots/compile command #2 + output from #1.png
 create mode 100644 Callum Rann's Screenshots/compile command #3 + output from #2.png
 create mode 100644 Callum Rann's Screenshots/compile command #4 + output from #3.png
 create mode 100644 Callum Rann's Screenshots/output from #4.png
```

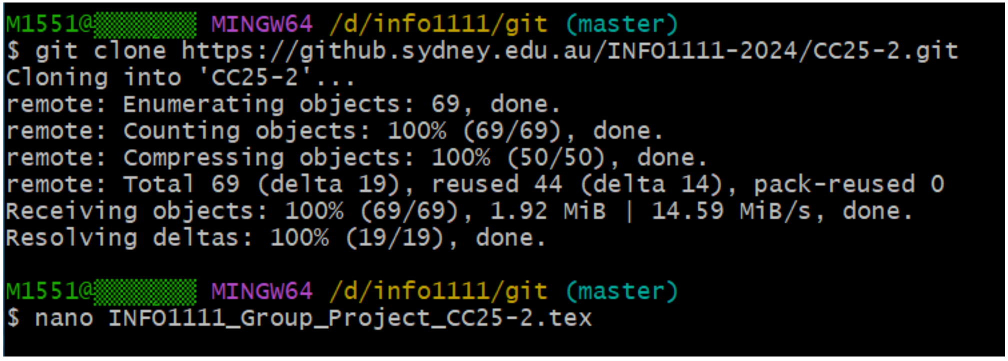
1.2. Skills for Data Science: MAI, ZIXI

In my opinion,the most important skills are DATS, DENG and DTAN.
1.Data Science: Data science serves as the fundamental knowledge base for this field[8], primarily focusing on the analysis of large volumes of data characterized by high velocity and variety, encompassing numerical, symbolic, textual, auditory, and visual information. It plays a pivotal role in the entire data lifecycle, from identifying and acquiring diverse data sources to refining requirements, validating models, and evolving them over time. By uncovering intricate patterns, trends, and correlations within datasets, data science facilitates informed decision-making and propels business growth through the discovery of valuable insights. It is indispensable for making predictions, generating recommendations, and extracting deeper insights from complex data landscapes.
2.Data Engineering: Data engineering includes designing, building, operationalising, securing and monitoring data pipelines and data stores, which provides the essential infrastructure and data resources that enable data science activities.[9] It prepares and processes data, making it accessible and usable for data scientists. Meanwhile, data science relies on data engineering support to access, analyze, and understand large-scale and heterogeneous data, enabling the discovery of valuable insights and the solution of complex problems. Thus, their relationship is symbiotic, working together to advance data-driven decision-making and business development. So DENG is a crucial part for this major.
3.Data Modelling and Design: Data modeling and design involves developing models and diagrams to represent and communicate data requirements and data assets. It is essential for structuring and organizing data in a way that aligns with business requirements and analytical needs[10], which provides the infrastructure and data structures for data science, enabling data scientists to access, understand, and analyze data effectively. Also, data science depends on the outcomes of data modelling and design to extract meaningful insights and conduct accurate analysis from structured and well-designed data. So data modelling and design is vital to this major.

PROOF OF GIT SKILLS

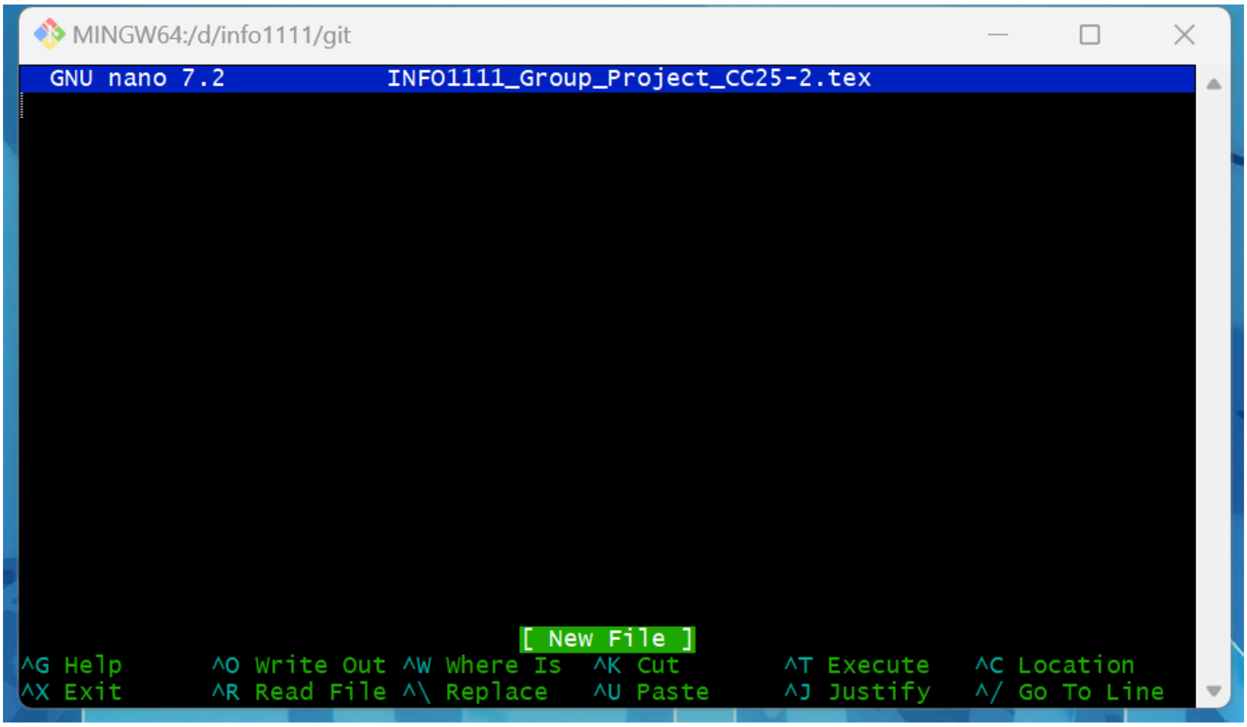
Provide evidence that you can compile from the command line.

Compile command 1 + output from 1 :



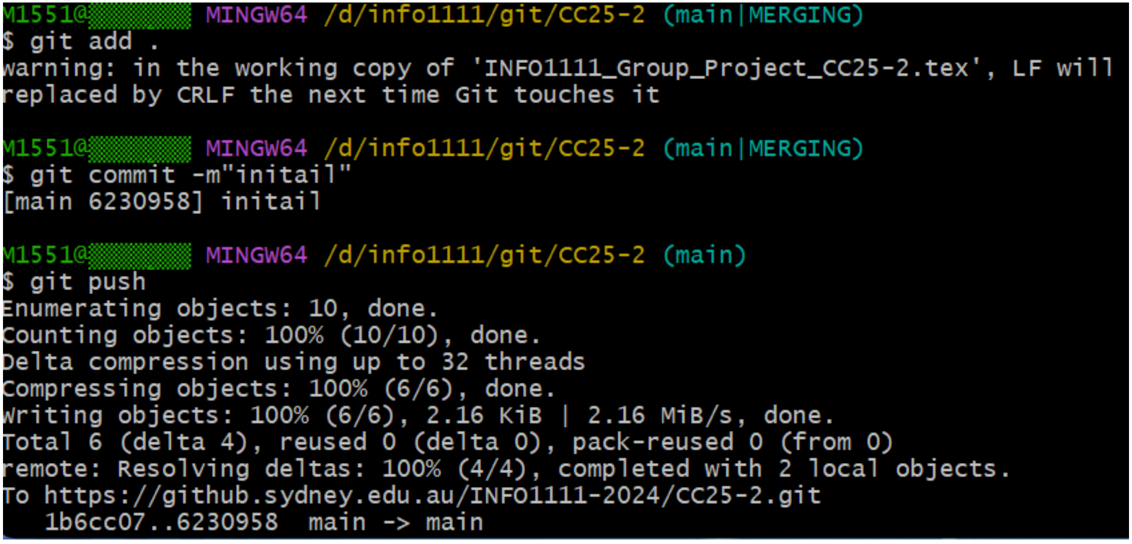
```
M1551@[redacted] MINGW64 /d/info1111/git (master)
$ git clone https://github.sydney.edu.au/INFO1111-2024/CC25-2.git
Cloning into 'CC25-2'...
remote: Enumerating objects: 69, done.
remote: Counting objects: 100% (69/69), done.
remote: Compressing objects: 100% (50/50), done.
remote: Total 69 (delta 19), reused 44 (delta 14), pack-reused 0
Receiving objects: 100% (69/69), 1.92 MiB | 14.59 MiB/s, done.
Resolving deltas: 100% (19/19), done.

M1551@[redacted] MINGW64 /d/info1111/git (master)
$ nano INFO1111_Group_Project_CC25-2.tex
```

Compile command 2:



```
  GNU nano 7.2              INFO1111_Group_Project_CC25-2.tex




                                  [ New File ]
^G Help         ^O Write Out  ^W Where Is  ^K Cut         ^T Execute    ^C Location
^X Exit         ^R Read File  ^\ Replace   ^U Paste       ^J Justify    ^/ Go To Line
```

Compile command 3 4 5 + output from 3 4 5:



```
M1551@[redacted] MINGW64 /d/info1111/git/CC25-2 (main|MERGING)
$ git add .
warning: in the working copy of 'INFO1111_Group_Project_CC25-2.tex', LF will
replaced by CRLF the next time Git touches it

M1551@[redacted] MINGW64 /d/info1111/git/CC25-2 (main|MERGING)
$ git commit -m"initail"
[main 6230958] initail

M1551@[redacted] MINGW64 /d/info1111/git/CC25-2 (main)
$ git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 32 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 2.16 KiB | 2.16 MiB/s, done.
Total 6 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
To https://github.sydney.edu.au/INFO1111-2024/CC25-2.git
   1b6cc07..6230958  main -> main
```

1.3. Skills for SW Development: ZHONG, QILONG

I think in descending order of importance, the three most important skills for software engineering students are Programming, Software Design and System Design.

Programming(PROG) is crucial for students in software engineering, as it is the core of building and modifying software to solve practical problems [11]. By mastering the ability of programming, not only can we use different programming languages such as Python and Java to complete our logical requirements, but we can also understand algorithms, data structures, software development cycles, and processes. Another crucial reason is that this skill can directly affect whether software design, testing, and post maintenance can be effectively implemented. Therefore, for those of us who want to make progress in the field of software development [12], PROG is our most important skill.

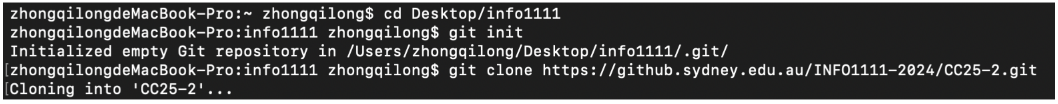
Software Design (SWDN) is critical for software engineering students because it bridges the gap between coding skills and the creation of practical [13], efficient solutions that meet user needs. This skill directly prepares students for future roles in software engineering, making it a critical component for those seeking to stand out in a competitive field [14]. SWDN is, therefore, an indispensable skill for students aiming for strong competitiveness in the software industry.

System Design (DESN) is crucial for software engineering students as it enables them to design comprehensive systems that effectively integrate various software components. This skill requires students to fully understand and design the structure of the entire system, ensuring that all parts of the system work together in an orderly manner to meet the requirements [15]. Mastering system design means that students can foresee how individual components will adapt to larger ecosystems, which is a key ability to develop efficient, scalable, and sustainable software solutions [16]. This global perspective is crucial for those who wish to make significant contributions in complex software project development.

PROOF OF GIT SKILLS

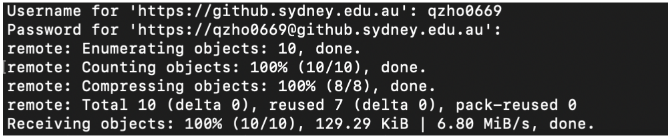
Provide evidence that you can compile from the command line.

Compile command 1 + output from 1 + Compile command 2:



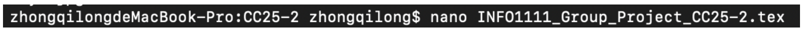
```
zhongqilongdeMacBook-Pro:~ zhongqilong$ cd Desktop/info1111
zhongqilongdeMacBook-Pro:info1111 zhongqilong$ git init
Initialized empty Git repository in /Users/zhongqilong/Desktop/info1111/.git/
zhongqilongdeMacBook-Pro:info1111 zhongqilong$ git clone https://github.sydney.edu.au/INFO1111-2024/CC25-2.git
Cloning into 'CC25-2'...
```

Output 2:



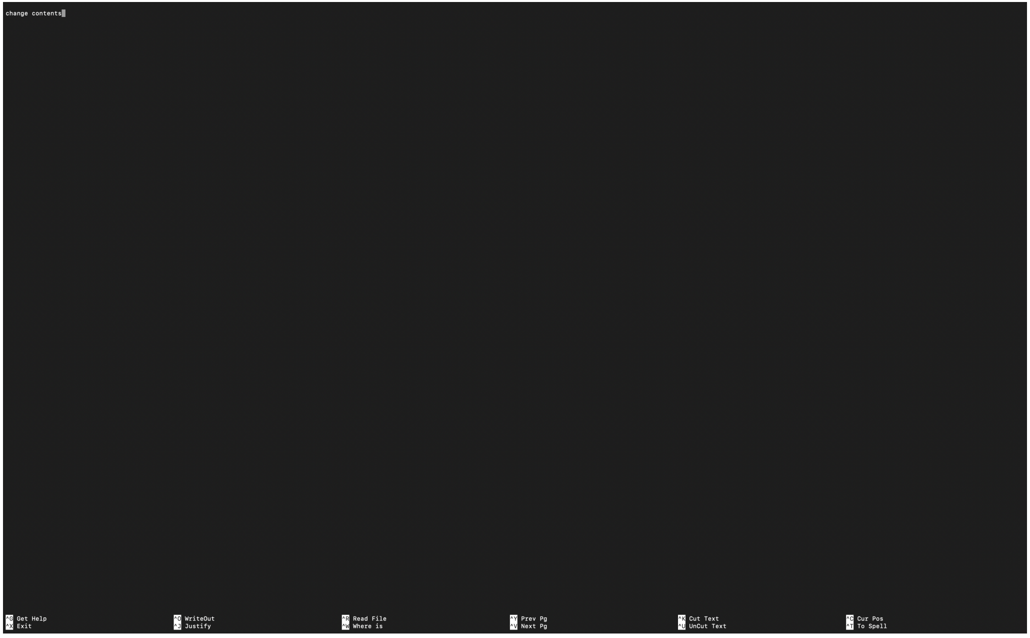
```
Username for 'https://github.sydney.edu.au': qzho0669
Password for 'https://qzho0669@github.sydney.edu.au':
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 10 (delta 0), reused 7 (delta 0), pack-reused 0
Receiving objects: 100% (10/10), 129.29 KiB | 6.80 MiB/s, done.
```

Compile command 3:



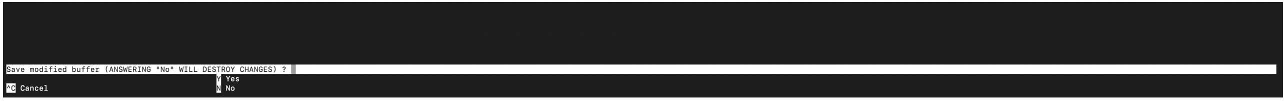
```
zhongqilongdeMacBook-Pro:CC25-2 zhongqilong$ nano INFO1111_Group_Project_CC25-2.tex
```

Output 3:



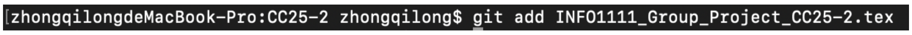
```
change contents
```

Compile command 4:



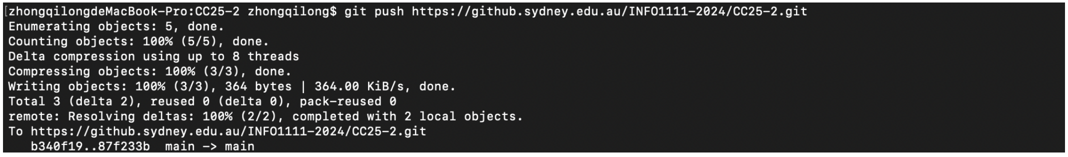
```
Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
```

Compile command 5:



```
[zhongqilongdeMacBook-Pro:CC25-2 zhongqilong$ git add INFO1111_Group_Project_CC25-2.tex
```

Compile command 6 + Output 5



```
[zhongqilongdeMacBook-Pro:CC25-2 zhongqilong$ git push https://github.sydney.edu.au/INFO1111-2024/CC25-2.git
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 364 bytes | 364.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.sydney.edu.au/INFO1111-2024/CC25-2.git
   b340f19..87f233b  main -> main
```

1.4. Skills for Cyber Security: Li, Senhao

SCTY(Information Security): It's the most important and required part for the cybersecurity industry. As stated in [17], It acts as a foundational skill to protect digital database from all unauthorized breaches and modifications,and addresses the core objective of cybersecurity: to protect information assets. Since the industry of cybersecurity is significant in the whole computer science, now frrequency and sophistication of cyber attacking is increasing day by day. Besides, the SCTY includes various applications of policies, technologies, and controls to safeguard related sensitive data, intellectual property, and other digital assets. Hence, this skill is really essential for every organization to guard their information privacy and enable the whole enterprise to gain customers' trust, which is bound to enhance profits and improve into a larger scale.

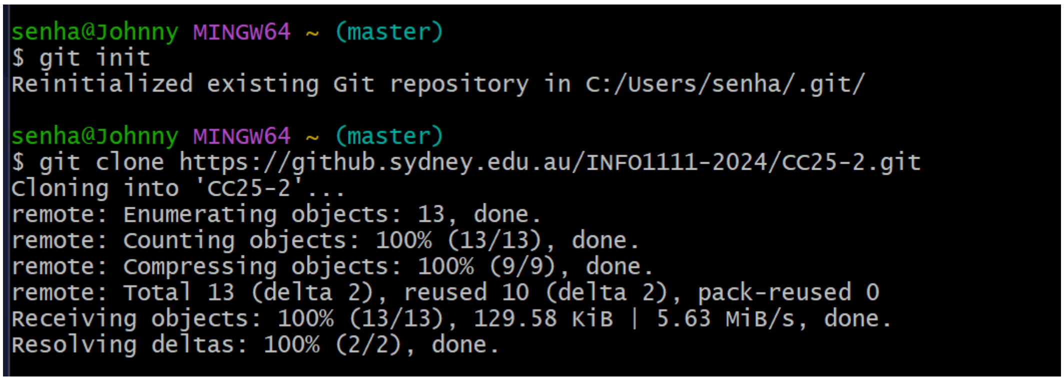
USUP(Incident Management): Managing incidents is also crucial for efficiently and effectively ensuring security in order to avoid cyber threats or breaches. Due to its critical role in responding to and recovering from security incidents, this skill follows the extent of requirement. Through using models to macroscopically control the changes and development trends of events, the skill enables specialist in this field to protect computer effectively and easily. In addition, as stated in [18], the demand for professionals who can navigate the complexities of incident responses, including communication, analysis and recovery, is high, also owing to nowaday's more sophisticated cyber attacks, as demonstrated in [19]. Based on such circumstance, USUP also acts as an significant skill for cybersecurity.

BURM(Risk Management): Though this skill is slightly less hands-on than the other two skills in direct cybersecurity activities, it's also a key to protect the whole environment of cybersecurity. BURM involves identification, classification, assessment, organization and communication of risks followed by the application of resources to minimize, control, and monitor the impact of these risks, just like an useful assistant that helps orderly manage those risks so as not to lead to troubles, bridging the gap between cybersecurity efforts and business objectives. It ensures that security practices not only protect information assets, but also support different types of business goals, which makes it also a key strategic skill in the cybersecurity domain.

PROOF OF GIT SKILLS

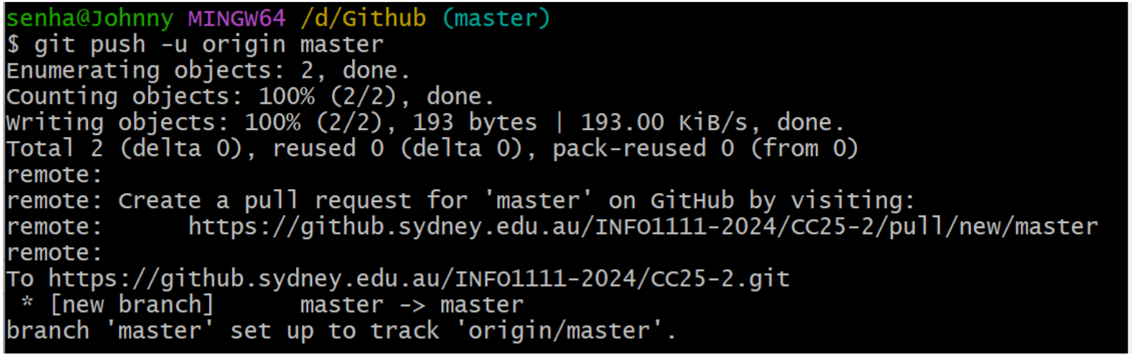
Provide evidence that you can compile from the command line.

Compile command 1 + output from 1 + Compile command 2 + output from 2:



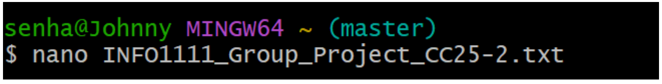
```
senha@Johnny MINGW64 ~ (master)
$ git init
Reinitialized existing Git repository in C:/Users/senha/.git/

senha@Johnny MINGW64 ~ (master)
$ git clone https://github.sydney.edu.au/INFO1111-2024/CC25-2.git
Cloning into 'CC25-2'...
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 13 (delta 2), reused 10 (delta 2), pack-reused 0
Receiving objects: 100% (13/13), 129.58 KiB | 5.63 MiB/s, done.
Resolving deltas: 100% (2/2), done.
```

Compile command 3 + output from 3:



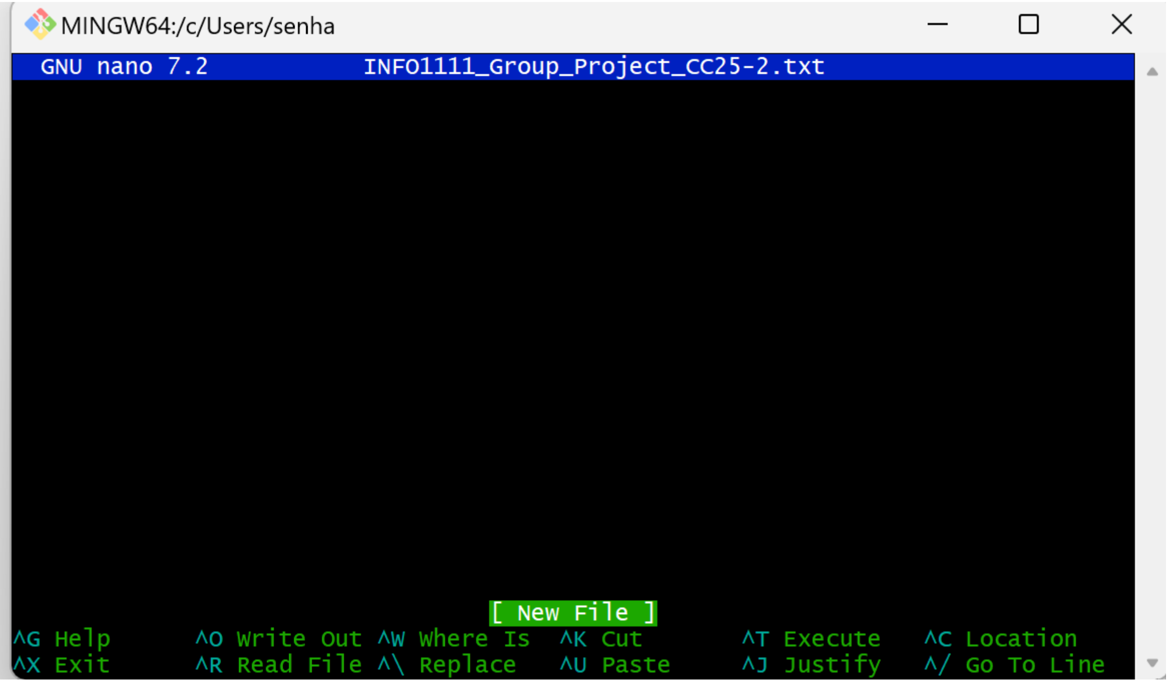
```
senha@Johnny MINGW64 /d/Github (master)
$ git push -u origin master
Enumerating objects: 2, done.
Counting objects: 100% (2/2), done.
Writing objects: 100% (2/2), 193 bytes | 193.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.sydney.edu.au/INFO1111-2024/CC25-2/pull/new/master
remote:
To https://github.sydney.edu.au/INFO1111-2024/CC25-2.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

Compile command 4 + output from 4:



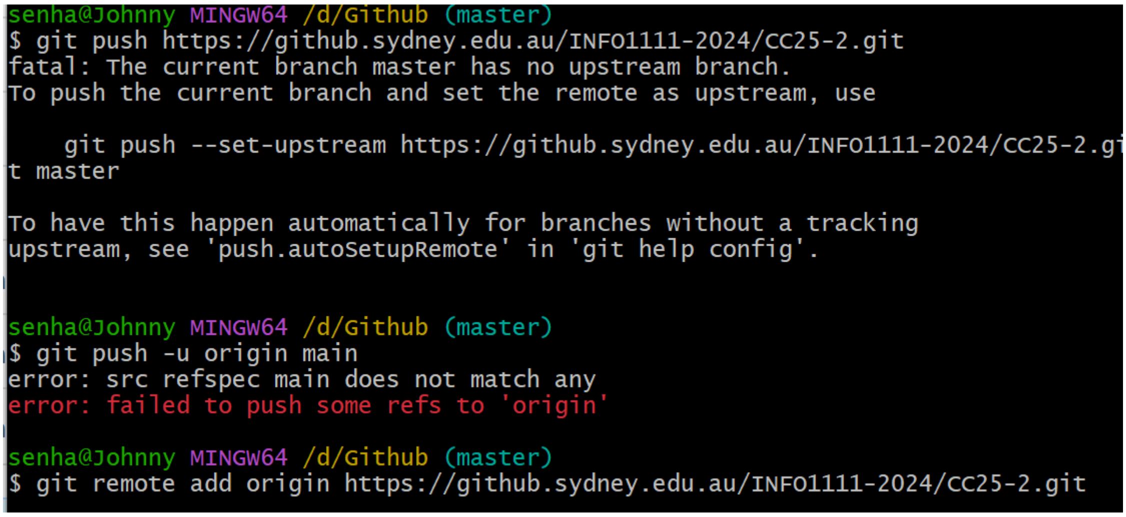
```
senha@Johnny MINGW64 ~ (master)
$ nano INFO1111_Group_Project_CC25-2.txt
```

Compile command 5 + output from 5:



```
MINGW64:/c/Users/senha
  GNU nano 7.2              INFO1111_Group_Project_CC25-2.txt

                                    [ New File ]
^G Help         ^O Write Out  ^W Where Is   ^K Cut          ^T Execute      ^C Location
^X Exit         ^R Read File  ^\ Replace    ^U Paste        ^J Justify      ^/ Go To Line
```

Compile command 6 + output from 6:



```
senha@Johnny MINGW64 /d/Github (master)
$ git push https://github.sydney.edu.au/INFO1111-2024/CC25-2.git
fatal: The current branch master has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream https://github.sydney.edu.au/INFO1111-2024/CC25-2.gi
t master

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.


senha@Johnny MINGW64 /d/Github (master)
$ git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'origin'

senha@Johnny MINGW64 /d/Github (master)
$ git remote add origin https://github.sydney.edu.au/INFO1111-2024/CC25-2.git
```

2. Task 2 (Advanced): Advanced Skills

Task 2 contains two components (both required).

Component 1: Exploration of Tech Tools

The first component focuses on exploration of relevant tech tools used within professional computing employment. All companies make use of a range of technologies and tools (often as part of a tech stack). These tools might be implementation languages; design tools; data analysis tools; collaboration technologies, etc. Each student should identify two tools that are widely used in industry, and which relate to the major you are focusing on for this project. You should then describe:

1. What are the two tools you have identified for your chosen major
2. The main functionality of those tools;
3. The ways in which those tools are used in the industry of your chosen major;
4. Any weaknesses or limitations of those tools.

This task consists of two parts:

1. **Part A**: Generate a set of questions that you can put to ChatGPT in order to obtain answers to each of the above four questions. Using ChatGPT, then generate the answers for each of the two tools. You must include in the report below both the questions that you posed to ChatGPT, and the answers that it provided. (100–250 words each).
2. **Part B**: For each of the four answers from Part A, assess the answer that ChatGPT provided and explain to us why you agree or disagree with the answer (100 words for each question above).

As examples of the tools which might be selected (which you shouldn't now use):

- Computer Science: Eclipse.
- Software Development: GitHub.
- Cyber Security: Wireshark.
- Data Science: Hadoop.

Note also that no two students in the same tutorial should choose the same tools, so your tutor will maintain a list of those that have already been selected. You should therefore check this list with your tutor and then confirm your choice with your tutor prior to researching your proposed tools and spending time writing about them. (Target = ∼200-400 words per tool).

Component 2: Advanced LaTeX and Git Skills

The second component of Task 2 focuses on more advanced technical skills in LaTeX and Git. The following is a list of advanced Git and LaTeX skills/features. Each student in your team that is attempting the Advanced task should select a different pair of items from each list (e.g. you might choose ”Resetting and Tags” from the git list, and ”Cross-referencing and Custom commands” from the LaTeX list). You then need to demonstrate actual use of each item (either through activity in Git, or through including items in this report). (Target = ∼100-200 words per student for each feature).

1. Git
    (a) Rebasing and Ignoring files
    (b) Forking and Special files
    (c) Resetting and Tags
    (d) Reverting and Automated merges
    (e) Hooks and Tags
2. LaTeX
    (a) Cross-referencing and Custom commands
    (b) Footnotes/margin notes and creating new environments
    (c) Floating figures and editing style sheets
    (d) Graphics and advanced mathematical equations
    (e) Macros and hyperlinks

OVERALL REQUIREMENTS:
To achieve an ”OK” rating for this task you must individually accomplish the following:

- **Component 1 - Exploration of Tech Tools**
    - Identified two tools that are widely used in industry, and which relate to the major chosen for this project.
        - The two tools selected are not the same as the tools selected by other students in the tutorial.
        - The two tools selected are relevant to the major chosen.
    - Answer the following questions as instructed in ’Part A’ & ’Part B’:
        - What are the two tools you have identified for your chosen major
        - 3 main functionality of each of the identified tools
        - The ways in which those tools are used in the industry of your chosen major;
        - 2 weaknesses or limitations of each of the tools
    - **Part A**: Generate a set of questions (minimum 5 questions) that can be put to ChatGPT in order to obtain answers to each of the above four questions. Using ChatGPT, then generate the answers for each of the two tools. You must include in the report below both the questions that you posed to ChatGPT, and the answers that it provided. (100 - 250 words for each question)
    - **Part B**: For each of the four answers from Part A, assess the answer that ChatGPT provided and explain to us why they agree or disagree with the answer (100 words for each question above).
- **Component 2 - Advanced LaTex & Git Skills**

  - Each member of the team has selected one pair of items from each list below and demonstrate actual use of each item (i.e. a Git item and a LaTeX item).
  - **Git**
    - Rebasing and Ignoring files
    - Forking and Special files
    - Resetting and Tags
    - Reverting and Automated merges
    - Hooks and Tags
  - **LATEX**
    - Cross-referencing and Custom commands
    - Footnotes/margin notes and creating new environments
    - Floating figures and editing style sheets
    - Graphics and advanced mathematical equations
    - Macros and hyperlinks
  - This means no two members of the team have not chosen the same item from either of the lists above.
  - You have demonstrated the use of your selected items either through activity in Git, or through including items in this report.
  - This means for Git items:
    - You have added your tutor to your git repository and when they view it they are able to see your activity that demonstrates the use of your selected items (e.g. forks, hooks, tags, merges etc.).
    - You have included screenshots and annotations (where necessary) in your report and provided an explanation of ∼100 words of your use of advanced Git features.
  - and for LaTeX items:
    - You have included items you have chosen in your LaTeX report document submission and the tutor is able to clearly see it (e.g. the pdf document written in LaTeX has hyperlinks, macros, cross referencing etc. included in it).
    - You have included screenshots and annotations (where necessary) in your report and provided an explanation of ∼100 words of your use of advanced LaTeX features.

- Referencing
  - You have provided in-text references (IEEE) to support your claims or where they gathered the information from.
  - You have a reference list following the IEEE referencing guidelines.
    - Some common things to look for to see whether your have correctly followed the referencing guide are:
    - Sources are listed in alphabetical order
    - The sources you have listed are only the sources that are present in-text.
    - All sources seen in-text are included in the reference list.

  - ∗ You followed the correct convention for references that don’t have author’s details or multiple sources have the same author and year of publication
  - ∗ You have included the required information for the source type as outlined in the guide.
  - ∗ Sources are not a list (i.e. dotpoints)

To achieve a ”STRONG” rating you must accomplish all of the above in addition to the following:

- The answers provided to the 4 questions (component 1b) use ChatGPT and independent research and analysis is excellent, showing a deep understanding of industry.
- You have used advanced Git features such as branching when demonstrating the items you selected (component 2a).

2.1. Tools and Skills for Computer Science: RANN, Callum

Part A: Exploration of tech tools

Tool Identification

The two tools identified for the Computer Science major was taken from industry guest presentation in week 7. From here I chose tools that offered functions I was not familiar with, in this case Docker’s containerisation, and Poetry’s package management. I was able to confirm these were widely used in industry by searching for the reccommended tools for these functions and these two popping up on multiple occassions [20], [21].

Tool 1 Questions - Docker

Question 1: What is the main functionality of thecomputer science tool Docker, in 250 words or less?

Docker is a powerful tool used in computer science primarily for containerization. It allows developers to package their applications and dependencies into standardized units called containers, which can then be easily deployed across different computing environments, from local machines to cloud servers.

The main functionality of Docker revolves around creating, managing, and deploying these containers. It provides a lightweight runtime environment that encapsulates the application and all its dependencies, ensuring consistency and reliability across various platforms.

With Docker, developers can build their applications in a consistent environment, eliminating the "it works on my machine" problem often encountered in software development. Containers created with Docker are portable and can run seamlessly on any system that supports Docker, regardless of the underlying infrastructure.

Docker also streamlines the development workflow by enabling rapid deployment and scaling of applications. Developers can quickly spin up containers, test their applications, and deploy updates without worrying about compatibility issues or configuration differences between environments.

Furthermore, Docker facilitates microservices architecture, allowing developers to break down complex applications into smaller, manageable components deployed as separate containers. This modular approach enhances scalability, flexibility, and maintainability of software systems.

Overall, Docker revolutionizes the way applications are developed, shipped, and run by providing a standardized and efficient containerization solution that simplifies deployment and management processes while promoting consistency and reproducibility in software development.

Question 2: How is Docker used in the computer science industry, in 250 words or less?

In the computer science industry, Docker is extensively used across various stages of the software development lifecycle. Initially, developers utilize Docker to create isolated development environments, ensuring consistency between different team members' setups and reducing "it works on my machine" issues. They package their applications and dependencies into Docker containers, which can be easily shared and replicated across development, testing, and production environments.

During testing, Docker facilitates continuous integration and continuous deployment (CI/CD) pipelines by automating the process of building, testing, and deploying applications. Integration with tools like Jenkins or GitLab CI allows for seamless integration of Docker containers into the testing workflow, enabling faster and more reliable delivery of software updates.

In production environments, Docker provides a lightweight and efficient platform for deploying and managing applications at scale. It enables organizations to adopt microservices architecture, breaking down monolithic applications into smaller, decoupled services deployed as independent containers. This approach improves scalability, fault isolation, and resource utilization, allowing for easier maintenance and updates.

Moreover, Docker plays a crucial role in cloud computing, enabling cloud-native development and deployment practices. Platforms like Kubernetes leverage Docker containers to orchestrate and manage complex distributed systems, automating tasks such as scaling, load balancing, and service discovery.

Overall, Docker revolutionizes software development and deployment practices in the computer science industry by providing a standardized and efficient containerization solution that improves consistency, scalability, and agility across the software development lifecycle.

Question 3: Does Docker have any weaknesses or limitations, in 250 words or less?

Yes, Docker, like any technology, has its share of weaknesses and limitations. These include:

Performance Overhead: Docker introduces a slight performance overhead due to the additional layer of abstraction it adds. While usually minimal, this overhead can impact resource utilization, especially in high-performance computing environments.

Security Concerns: Docker containers share the same kernel as the host system, which poses security risks. Vulnerabilities in the kernel or misconfigurations could potentially allow containerized applications to access host resources, leading to security breaches.

Persistence and State Management: Docker containers are typically designed to be stateless and ephemeral. Managing persistent data and stateful applications within Docker containers can be challenging and may require additional configurations or external storage solutions.

Learning Curve: Docker introduces new concepts and terminologies, which may have a learning curve for users unfamiliar with containerization. Understanding how to create efficient Dockerfiles, manage images, and orchestrate containers with tools like Docker Compose or Kubernetes requires time and effort.

Compatibility Issues: Docker dependencies on specific versions of Docker Engine and container runtimes can lead to compatibility issues. Ensuring compatibility between different versions of Docker and containerized applications can be a concern, particularly in large-scale deployments with diverse environments.

Despite these limitations, Docker remains a powerful tool for containerization, offering significant benefits in terms of portability, scalability, and efficiency for modern software development and deployment.

Tool 2 Questions - Poetry

Question 1: What is the main functionality of the computer science tool Poetry, in 250 words or less?

Poetry is a tool primarily used for managing Python dependencies and virtual environments. It simplifies the process of dependency management by providing a robust and consistent approach. With Poetry, developers can define project dependencies and their versions in a pyproject.toml file, similar to other modern package managers like npm or pipenv.

One of its main features is dependency resolution, which ensures that all dependencies are compatible with each other and with the Python version specified for the project. Poetry also creates and manages isolated virtual environments for each project, preventing conflicts between dependencies across different projects.

Furthermore, Poetry facilitates package publishing to PyPI, making it easy for developers to share their Python projects with the community. It automates tasks such as building and publishing packages, saving developers time and effort.

Overall, Poetry streamlines the Python development workflow by providing a comprehensive solution for dependency management, virtual environment creation, and package publishing. Its user-friendly interface and powerful features make it a popular choice among Python developers for managing project dependencies and ensuring project stability and compatibility.

Question 2: How is Poetry used in the computer science industry, in 250 words or less?

In the computer science industry, Poetry is widely used to streamline Python development workflows, particularly in projects of varying scales and complexities. Its popularity stems from several key factors.

Firstly, Poetry simplifies dependency management by allowing developers to define project dependencies in a single pyproject.toml file. This eliminates the need for complex requirements.txt files or manual dependency tracking, reducing errors and ensuring consistency across development environments.

Secondly, Poetry automates the creation and management of virtual environments, which are isolated environments where project dependencies can be installed without affecting system-wide Python installations. This isolation prevents conflicts between dependencies and ensures project stability.

Thirdly, Poetry's dependency resolution algorithm ensures that all dependencies are compatible with each other and with the specified Python version. This helps prevent dependency hell and ensures that projects can be easily reproduced on different systems.

Additionally, Poetry provides tools for packaging and publishing Python projects to platforms like PyPI, simplifying the process of sharing code with the wider community or within an organization. This makes it easier for developers to collaborate and contribute to open-source projects.

Overall, Poetry has become an essential tool in the computer science industry for managing Python projects efficiently and reliably. Its user-friendly interface, robust features, and focus on best practices make it a go-to choice for developers looking to streamline their development workflows and ensure project success.

Question 3: Does Poetry have any weaknesses or limitations, in 250 words or less?

While Poetry offers numerous benefits for managing Python projects, it's not without its limitations and potential drawbacks.

One limitation is that Poetry's integration with certain development environments or continuous integration (CI) systems may not be as seamless as with other tools. While it has improved compatibility over time, some setups may require additional configuration or workarounds to fully integrate Poetry into the development workflow.

Another consideration is that Poetry is still relatively young compared to more established tools like pipenv or traditional requirements.txt files. As a result, it may lack some features or integrations that users have come to expect from more mature tools. However, the Poetry community is actively working to address these limitations and improve the tool's functionality.

Furthermore, while Poetry excels at managing Python dependencies, it may not be the best choice for projects with complex multi-language dependencies or specialized requirements outside of Python development.

Additionally, there may be a learning curve for developers who are accustomed to other dependency management tools or workflows. Transitioning to Poetry may require time and effort to become familiar with its unique features and conventions.

Despite these limitations, Poetry continues to evolve and gain traction in the Python community due to its focus on simplicity, reliability, and modern best practices. As with any tool, it's essential to evaluate Poetry's strengths and weaknesses in the context of your specific project requirements and development workflow.

Part B: Analysis

Tool 1 Response Evaluation - Docker

Functionality of Docker:

ChatGPT's response effectively captures Docker's primary functionality, closely echoing the content provided in [22]. Docker is described as a platform for developing, shipping, and running applications, "Docker enables you to separate your applications from your infrastructure so you can deliver software quickly". This aligns with ChatGPT's explanation of Docker's role in containerization, which allows developers to package applications and dependencies into standardized units called containers. Therefore, ChatGPT's response provides an accurate summary of Docker's main functionality.

Usage of Docker in the computer science industry:

ChatGPT's response accurately outlines Docker's usage across various stages of the software development lifecycle, which closely mirrors the content provided in [22]. The source emphasizes Docker's significance in continuous integration and continuous delivery workflows, stating that "Containers are great for continuous integration and continuous delivery (CI/CD) workflows". Similarly, ChatGPT highlights Docker's role in facilitating CI/CD pipelines during testing and deployment phases. Therefore, ChatGPT's response aligns well with the provided source, effectively summarizing Docker's usage in the computer science industry.

Weaknesses or limitations of Docker:

ChatGPT's response identifies several weaknesses and limitations of Docker, which are consistent with the drawbacks outlined in [23]. The source mentions security risks associated with Docker containers, stating that "Docker containers are not completely isolated from the host system". Similarly, ChatGPT mentions security concerns as one of Docker's limitations. Furthermore, the source discusses the complexity of Docker adoption, stating that "Docker can be complex to learn and use", which aligns with ChatGPT's point about Docker's learning curve. Therefore, ChatGPT's response provides a balanced assessment of Docker's strengths and weaknesses, effectively summarizing key points from the provided source.

In conclusion, ChatGPT's responses accurately summarize Docker's functionality, usage, and limitations, aligning closely with the information provided in the sources.

Tool 2 Response Evaluation - Poetry

Functionality of Poetry:
ChatGPT's response accurately portrays Poetry as a tool for managing Python dependencies and virtual environments, simplifying dependency management with a consistent approach. This aligns with the functionality described "Poetry is a tool for dependency management and packaging in Python. It allows you to declare the libraries your project depends on and it will manage (install/update) them for you." [24].

Additionally, Poetry offers features like a lockfile to ensure repeatable installs and project distribution capabilities. ChatGPT's response effectively captures Poetry's primary functionality, as described in the source.

Usage of Poetry in the Computer Science Industry:
ChatGPT describes Poetry's widespread adoption in the computer science industry for streamlining Python development workflows. While not directly quoting [25], which mentions considering Poetry for managing Python applications, the response captures the essence of the source by emphasizing Poetry's role in improving dependency management in Python projects.

Weaknesses or Limitations of Poetry:
ChatGPT acknowledges Poetry's limitations, such as integration challenges and potential drawbacks. These limitations are consistent with the experiences shared in [25], where users mention issues with extra dependencies and build-time concerns when adopting Poetry for dependency management. One user expresses concerns about Poetry's impact on their workflow:

"Easily the biggest pain-point of having moved to, and now wanting to move away from, poetry is that you now have an extra build-time and / or runtime dependency you have to install on the system prior to installing your dependencies."

This quote highlights the practical challenges users may encounter when transitioning to Poetry, supporting ChatGPT's assertion regarding Poetry's limitations.

Overall, ChatGPT's responses effectively integrate information from the provided sources, providing a comprehensive evaluation of Poetry's functionality, usage, and limitations. The inclusion of quotes strengthens the evaluation by directly referencing user experiences and descriptions of Poetry's functionality from the sources. Therefore, ChatGPT's responses can be considered accurate assessments of Poetry based on the given sources.

Technical Skills (LaTeX and Git)

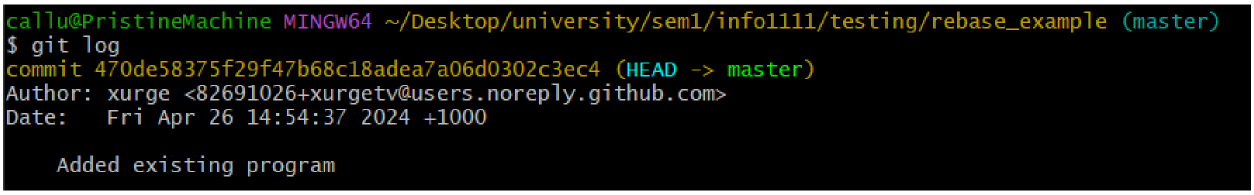
Git Skill: Rebasing and Ignoring Files

Rebasing:

Rebasing provides an alternative to merging, rather than presenting the joining of branches as a single merge commit, it sends the new branches one by one, allowing for a more linear history and easier conflict resolution.

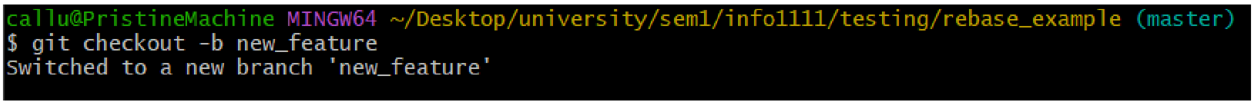
Here is an example:

- Original logs of master branch



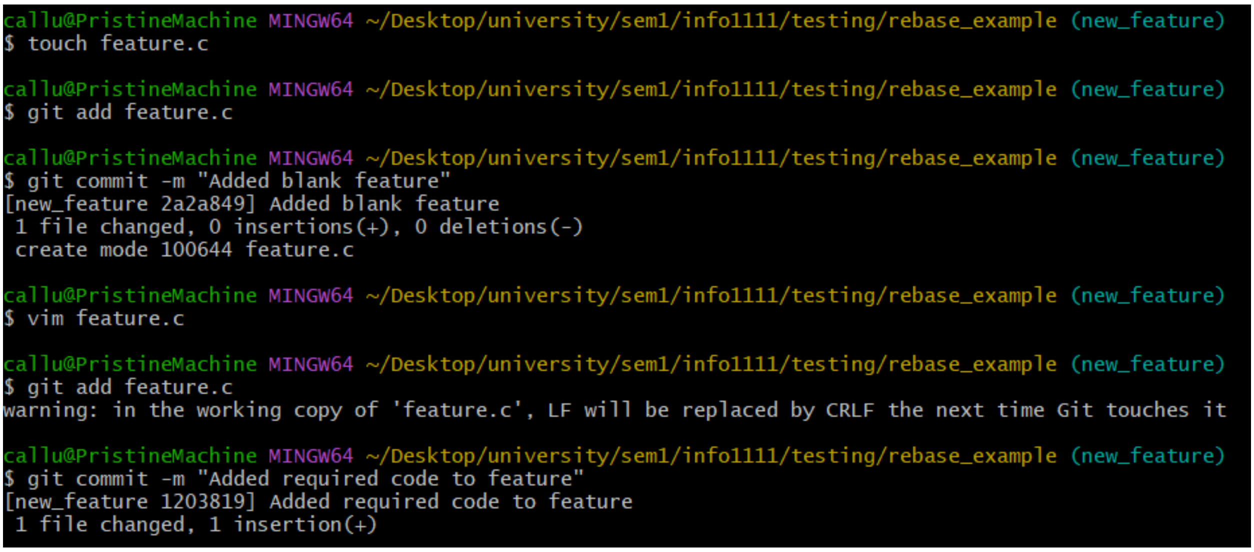
```
callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/rebase_example (master)
$ git log
commit 470de58375f29f47b68c18adea7a06d0302c3ec4 (HEAD -> master)
Author: xurge <82691026+xurgetv@users.noreply.github.com>
Date:   Fri Apr 26 14:54:37 2024 +1000

    Added existing program
```

- Creating the new branch



```
callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/rebase_example (master)
$ git checkout -b new_feature
Switched to a new branch 'new_feature'
```

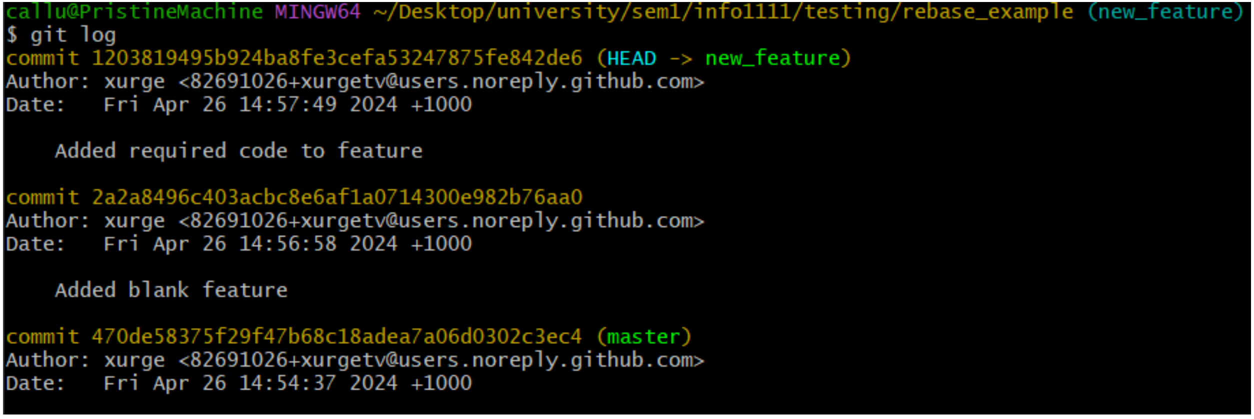
- Adding multiple commits to the new branch



```
callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/rebase_example (new_feature)
$ touch feature.c

callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/rebase_example (new_feature)
$ git add feature.c

callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/rebase_example (new_feature)
$ git commit -m "Added blank feature"
[new_feature 2a2a849] Added blank feature
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature.c

callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/rebase_example (new_feature)
$ vim feature.c

callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/rebase_example (new_feature)
$ git add feature.c
warning: in the working copy of 'feature.c', LF will be replaced by CRLF the next time Git touches it

callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/rebase_example (new_feature)
$ git commit -m "Added required code to feature"
[new_feature 1203819] Added required code to feature
 1 file changed, 1 insertion(+)
```

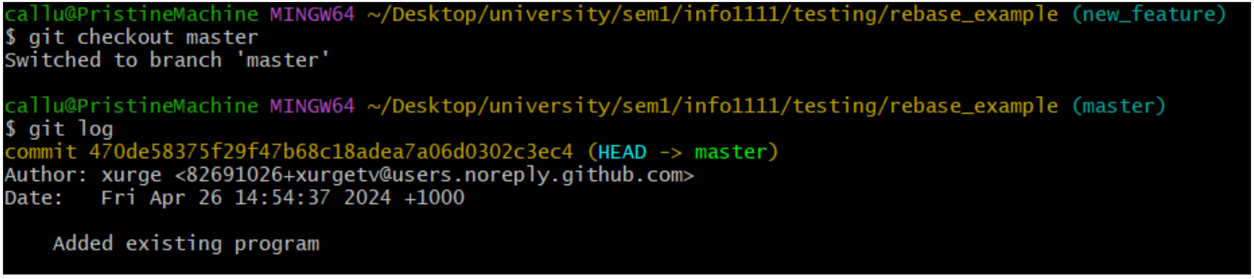
- The new branch logs vs the master logs

- New branch log



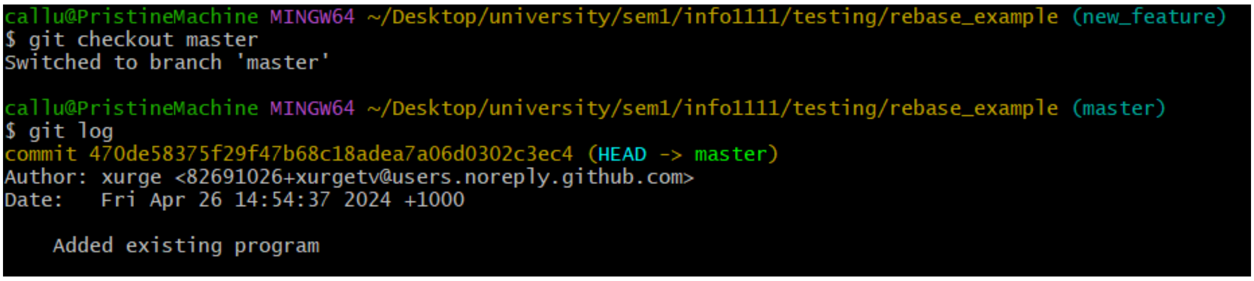
```
callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/rebase_example (new_feature)
$ git log
commit 1203819495b924ba8fe3cefa53247875fe842de6 (HEAD -> new_feature)
Author: xurge <82691026+xurgetv@users.noreply.github.com>
Date:   Fri Apr 26 14:57:49 2024 +1000

    Added required code to feature

commit 2a2a8496c403acbc8e6af1a0714300e982b76aa0
Author: xurge <82691026+xurgetv@users.noreply.github.com>
Date:   Fri Apr 26 14:56:58 2024 +1000

    Added blank feature

commit 470de58375f29f47b68c18adea7a06d0302c3ec4 (master)
Author: xurge <82691026+xurgetv@users.noreply.github.com>
Date:   Fri Apr 26 14:54:37 2024 +1000
```

- Master log



```
callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/rebase_example (new_feature)
$ git checkout master
Switched to branch 'master'

callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/rebase_example (master)
$ git log
commit 470de58375f29f47b68c18adea7a06d0302c3ec4 (HEAD -> master)
Author: xurge <82691026+xurgetv@users.noreply.github.com>
Date:   Fri Apr 26 14:54:37 2024 +1000

    Added existing program
```

- Within the master, rebasing the new branch and seeing the now updated logs with commits from new branch



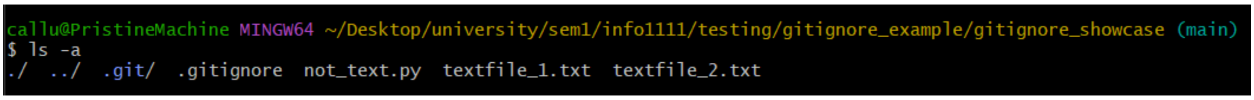
```
callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/rebase_example (new_feature)
$ git checkout master
Switched to branch 'master'

callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/rebase_example (master)
$ git log
commit 470de58375f29f47b68c18adea7a06d0302c3ec4 (HEAD -> master)
Author: xurge <82691026+xurgetv@users.noreply.github.com>
Date:   Fri Apr 26 14:54:37 2024 +1000

    Added existing program
```

Ignoring Files:

A .gitignore file allows for a more efficient use of the git add command, allowing it to ignore specific files or file patterns that appear in the .gitignore file. This is useful for keeping the remote repository clean of temporary files, and reducing the risk of uploading sensitive data.

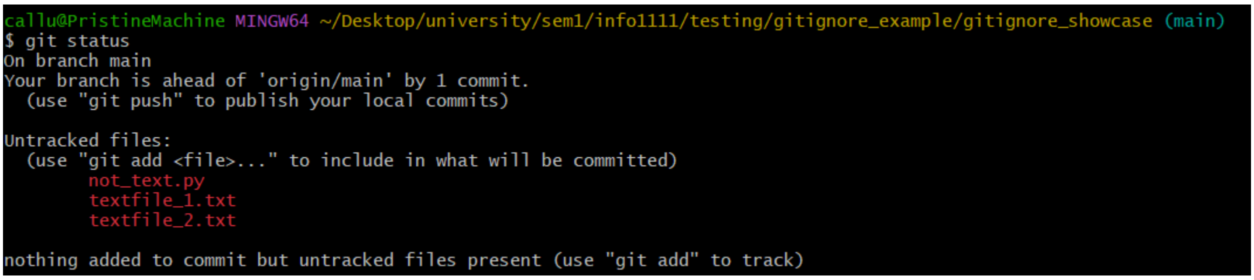
Below is an example:

- View of starting files in repository



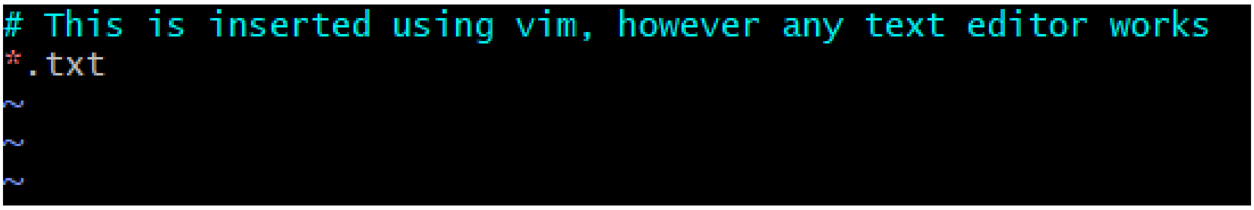
```
callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/gitignore_example/gitignore_showcase (main)
$ ls -a
./  ../  .git/  .gitignore  not_text.py  textfile_1.txt  textfile_2.txt
```

- Using git status to see all unstaged files that have been created or changed



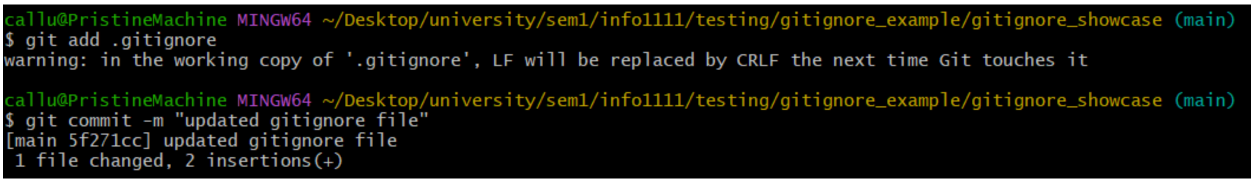
```
callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/gitignore_example/gitignore_showcase (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        not_text.py
        textfile_1.txt
        textfile_2.txt

nothing added to commit but untracked files present (use "git add" to track)
```

- Adding a pattern into the .gitignore file that ignores all files with txt extension



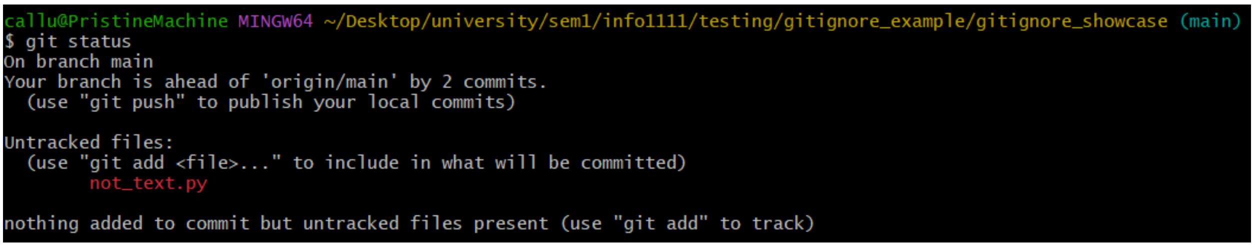
```
# This is inserted using vim, however any text editor works
*.txt
~
~
~
```

- Git adding and committing the .gitignore file



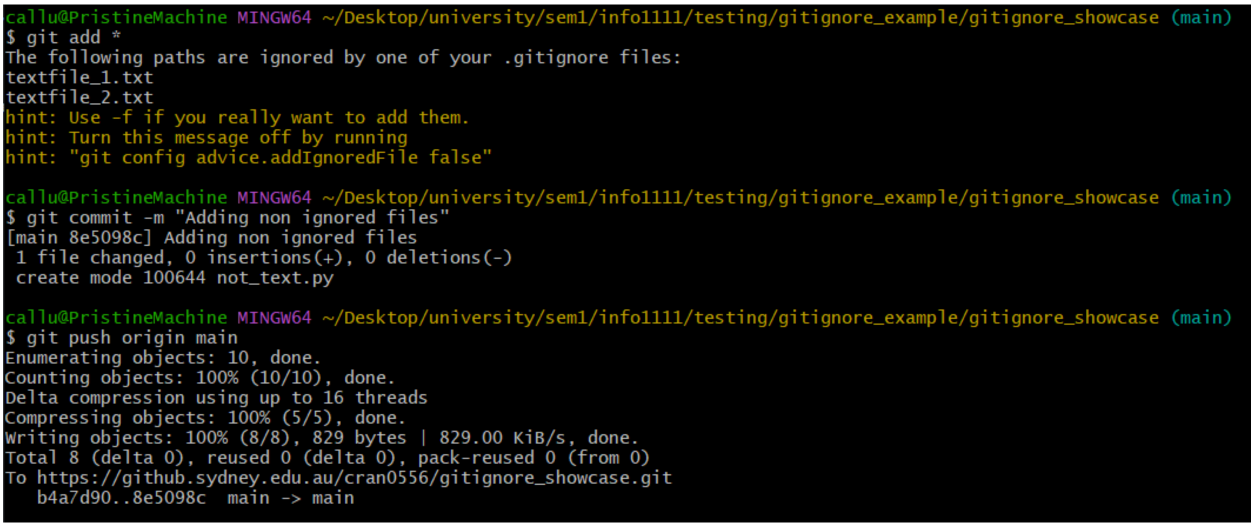
```
callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/gitignore_example/gitignore_showcase (main)
$ git add .gitignore
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it

callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/gitignore_example/gitignore_showcase (main)
$ git commit -m "updated gitignore file"
[main 5f271cc] updated gitignore file
 1 file changed, 2 insertions(+)
```

- Checking the status of the repository again shows all txt files are no longer included in tracking for next commit check



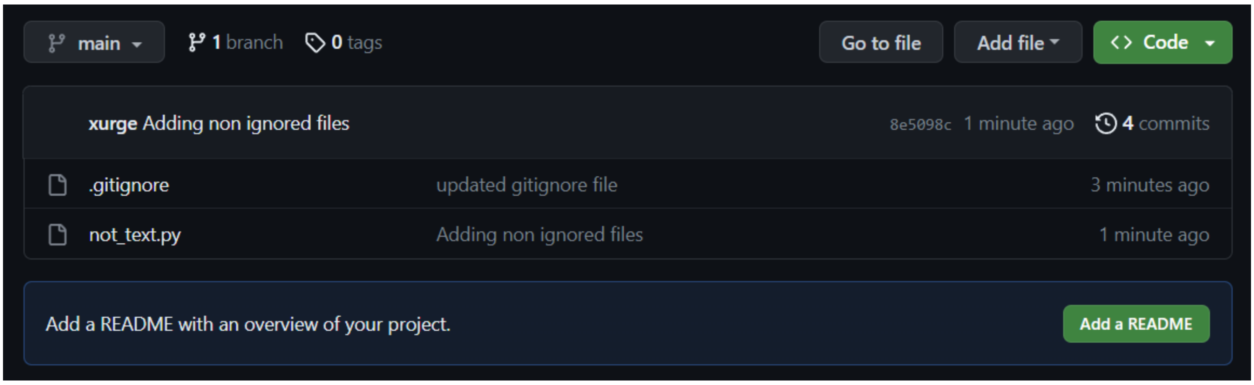
```
callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/gitignore_example/gitignore_showcase (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        not_text.py

nothing added to commit but untracked files present (use "git add" to track)
```

- Now using commands that normally add and commit all changed or created files



```
callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/gitignore_example/gitignore_showcase (main)
$ git add *
The following paths are ignored by one of your .gitignore files:
textfile_1.txt
textfile_2.txt
hint: Use -f if you really want to add them.
hint: Turn this message off by running
hint: "git config advice.addIgnoredFile false"

callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/gitignore_example/gitignore_showcase (main)
$ git commit -m "Adding non ignored files"
[main 8e5098c] Adding non ignored files
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 not_text.py

callu@PristineMachine MINGW64 ~/Desktop/university/sem1/info1111/testing/gitignore_example/gitignore_showcase (main)
$ git push origin main
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (8/8), 829 bytes | 829.00 KiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.sydney.edu.au/cran0556/gitignore_showcase.git
   b4a7d90..8e5098c  main -> main
```

- Can see the remote repository did not receive ignore files in commit



xurge Adding non ignored files 8e5098c 1 minute ago 4 commits

.gitignore	updated gitignore file	3 minutes ago
not_text.py	Adding non ignored files	1 minute ago

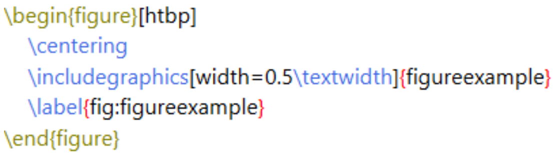
Add a README with an overview of your project.

LaTeX Skill: Cross-referencing and Custom Commands

Cross-referencing:

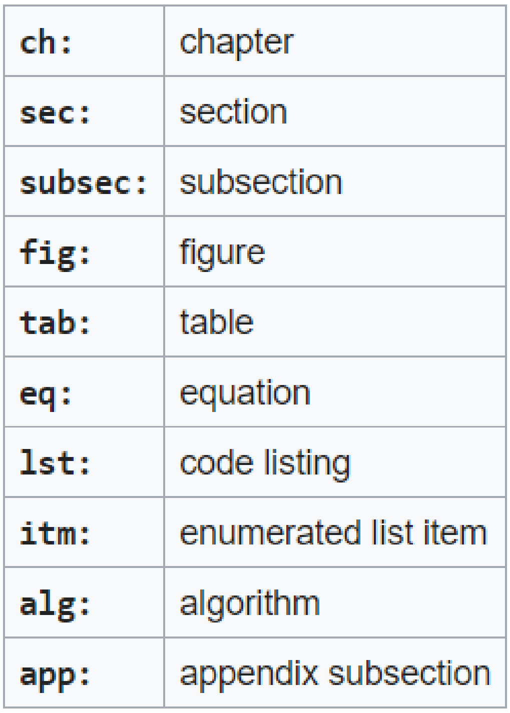
Cross-referencing allows for text to be linked to section, figures, equations, etc., each with their own numbers and bracket formatting, allowing for a clear indication of what information is being referred to in text. This is done through the `\label{type:name}` and `\ref{type:name}` commands, ensuring that the name in both is the same. It is important that the type exists in the correct environment or on the line of code that is is refering to. This pattern can be seen in action in the following:

- I use the fig: type here 2.1. along with an environment to apply appropriate labling and brackets and numbering.



```
\begin{figure}[htbp]
    \centering
    \includegraphics[width=0.5\textwidth]{figureexample}
    \label{fig:figureexample}
\end{figure}
```

- Now here is a cross-reference to a figure again, but we will refer to it as a table (sourced from [26]). The table displays all the different types of lables 2.1. Notice that it also has the lable 2.1. This indicates that the referenced material is in section 2.1 of the latex document.

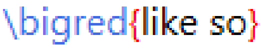


`ch:`	chapter
`sec:`	section
`subsec:`	subsection
`fig:`	figure
`tab:`	table
`eq:`	equation
`lst:`	code listing
`itm:`	enumerated list item
`alg:`	algorithm
`app:`	appendix subsection

Custom Commands:

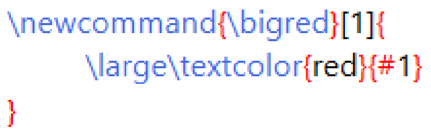
The benefits of using `\newcommand`, is that it allows for the defining of otherwise tedious, or space consuming commands to be used under a new name, such as a command that makes text large and red. This can be done like so . Here is what's happening:

- The command



```
\bigred{like so}
```

- The code



```
\newcommand{\bigred}[1]{
    \large\textcolor{red}{#1}
}
```

Breaking done what is happening here:
This is done by first giving the new command a name in curly brackets following the `\newcommand\{name here\}`. From here another set of curly brackets holds the commands that you want to perform. If you require input into your command, place square brackets between the sets of curly brackets like this:

`\newcommand{name}[number of inputs]{command description}`,

which then holds the number of inputs you require. To then refer to these inputs, you must use the order they appear in after a hastag, following this exampple. See how there are two inputs, one in the first curly brackets, and one in the next. The first text provided in refered to in the first line of new command descriptor and makes the text blue, referring with #1. Whereas the second input is referred to with #2, and is used in the second line to make the text red.

2.2. Tools and Skills for Data Science: MAI, ZIXI

Part A: Exploration of Tech Tools

Tool Identification

In the field of data science, I have chosen R and SQL, two highly practical tools that are widely used by data scientists. These tools are essential due to their powerful functionalities in data analysis, management, and statistical computation, making them indispensable in the industry.

Tool 1 Questions - R

Question 1: What is the main function of R in the field of data science, in 250 words or less?

R is a programming language and software environment specifically designed for statistical computing and graphics. Its primary function in data science is to provide a comprehensive suite of tools that facilitate data manipulation, statistical analysis, and visualization.

R offers a vast array of statistical tests and models, enabling data scientists to perform complex analyses, from basic descriptive statistics to advanced inferential techniques. The language's built-in functions and extensive package ecosystem support a wide range of statistical methodologies, making it an ideal tool for researchers and analysts.

R's powerful data visualization capabilities, particularly through the ggplot2 package, allow for the creation of high-quality, customizable plots that help in exploring and presenting data insights. Additionally, R provides robust data manipulation tools, such as the dplyr and tidyr packages, which streamline the process of cleaning and transforming data for analysis.

These features make R a versatile tool that supports the entire data analysis workflow, from data preparation and exploration to modeling and reporting, ensuring that data scientists can efficiently derive meaningful insights from their data.

Question 2: How is R used in the data science industry, in 250 words or less?

R is extensively used in the data science industry for a variety of applications across different sectors.

In academia and research, R is a preferred tool for conducting statistical analyses and hypothesis testing, thanks to its comprehensive statistical functions and reproducibility. Researchers utilize R to develop new statistical methods and to publish detailed analytical results.

In the pharmaceutical industry, R is critical for analyzing clinical trial data, aiding in drug development and ensuring regulatory compliance. Bioinformatics professionals use R for analyzing genomic data, performing sequence analysis, and developing predictive models for biological phenomena.

The finance sector also heavily relies on R for risk modeling, portfolio optimization, and quantitative analysis, where accurate statistical analysis is crucial. Businesses leverage R to build predictive models that forecast sales, customer behavior, and market trends, facilitating data-driven decision-making.

R's ability to handle complex statistical tasks and its extensive library of packages tailored for various analytical needs make it a vital tool in the data science industry, enabling professionals to tackle diverse data challenges effectively.

Question 3: Does R have any weaknesses or limitations, in 250 words or less?

Yes, R has several weaknesses and limitations.

One major limitation is its memory management. R often requires loading entire datasets into memory, which can be problematic when dealing with very large datasets. This limitation can lead to performance issues, as processing large datasets may require substantial RAM, which might not be available on standard personal computers. Consequently, R's performance can degrade significantly with increasing data size, leading to slower computations and potential crashes.

Another limitation of R is its steep learning curve, especially for users who are not familiar with programming or statistical analysis. The extensive range of functions and packages can be overwhelming for beginners, making it challenging to navigate and utilize R effectively without substantial time and effort invested in learning.

Additionally, while R excels in statistical analysis and visualization, it may not be as efficient for general-purpose programming tasks or large-scale data processing compared to other languages like Python. These weaknesses highlight the need for careful consideration when choosing R for specific data science tasks, ensuring that its strengths align with the project requirements and available resources.

Tool 2 Questions - SQL

Question 1: What is the main function of SQL in the field of data science, in 250 words or less?

SQL, which stands for Structured Query Language, is a specialized programming language designed for managing and manipulating relational databases. Its primary function in data science is to facilitate the efficient querying, updating, and management of structured data stored in relational databases.

SQL allows data scientists to perform complex queries to filter, sort, and aggregate data, enabling them to extract valuable insights from large datasets. The language's straightforward syntax makes it easy to write and understand queries, which helps in retrieving specific data needed for analysis.

SQL supports various operations such as data insertion, updating, and deletion, ensuring data integrity and consistency through the use of constraints and transactions. Additionally, SQL's aggregate functions, such as SUM, AVG, COUNT, MAX, and MIN, allow for effective statistical analysis directly within the database, making it a powerful tool for generating reports and summarizing data.

Overall, SQL's robust capabilities in managing and querying structured data make it an essential tool for data scientists who need to work with large and complex datasets stored in relational databases.

Question 2: How is SQL used in the data science industry, in 250 words or less?

SQL is widely used in the data science industry for managing and analyzing structured data across various sectors.

In the finance industry, SQL is essential for handling large volumes of transactional data, customer records, and financial information. Financial analysts use SQL to query databases for real-time insights, generate financial reports, and perform risk analysis.

In the retail sector, SQL helps manage inventory, track sales, and analyze customer data, providing valuable insights for optimizing stock levels and personalizing marketing strategies. SQL is also crucial in healthcare, where it is used to manage patient records, clinical data, and research information, enabling healthcare professionals to conduct data-driven research and improve patient care.

In technology companies, SQL is used for managing big data generated by applications, websites, and IoT devices. Data engineers and analysts use SQL to process and analyze this data, integrating it with other data processing tools to create comprehensive data workflows.

SQL's ability to handle structured data efficiently and its compatibility with various database systems make it a versatile and indispensable tool in the data science industry, facilitating data management, analysis, and reporting across different domains.

Question 3: Does SQL have any weaknesses or limitations, in 250 words or less?

Yes, SQL has several weaknesses and limitations.

One of the main limitations of SQL is its limited expressive power when dealing with complex unstructured data. SQL is designed primarily for structured data and excels in managing and querying relational databases, but it struggles with unstructured data such as text, images, and videos. Handling such data often requires additional tools or languages, such as NoSQL databases, which are better suited for managing unstructured data.

Another limitation is performance overhead. While SQL queries are efficient for structured data, complex queries on large datasets can lead to performance bottlenecks, requiring optimization of database structures, indexing, and query tuning. Ensuring compatibility between different versions of SQL and database systems can also pose challenges, particularly in large-scale deployments with diverse environments.

Additionally, SQL's reliance on a predefined schema can be restrictive in scenarios where data models need to evolve quickly. Schema changes can be cumbersome and may require significant modifications to existing queries and applications.

Despite these limitations, SQL remains a powerful tool for managing and querying structured data, but its effectiveness can be enhanced when used in conjunction with other technologies that address its weaknesses.

Part B: Analysis

Tool 1 Response Evaluation - R

Functionality of R:

I agree with this answer. It rightly highlights the main functions of R in data science, including statistical analysis, data manipulation, and visualization. The mention of specific packages such as ggplot2 for visualization and dplyr and tidyr for data manipulation emphasizes R's strengths in handling diverse data science tasks. This answer effectively summarizes R's comprehensive capabilities and clearly illustrates why R is the tool of choice in data science.

Usage of R in the Data Science Industry:

I agree with this answer. It accurately describes how R is used in different industries, demonstrating its versatility. Also, R is a popular choice among data scientists and is used by many organizations, including Fortune 500 companies, government agencies, and top research institutions.[27] The examples provided—academia, pharmaceutical industry, bioinformatics, finance, and business—are relevant and highlight the wide range of applications for R. The answer is very well written and organized. Emphasis on R's ecosystem of packages further demonstrates its adaptability and effectiveness in handling specific analytical needs. This answer is comprehensive and in line with practical applications of R in the data science industry.

Weaknesses or Limitations of R:

I agree with this answer. It rightly points out the significant limitations of R, including memory management issues and a steep learning curve. For example, a large and varied set of built-in functions is not enough to do all the cool things you can do with R, from plotting compelling data visualizations to training powerful machine learning models.[28] The difference between R's strengths in statistical tasks and its inefficiencies in general-purpose programming or large-scale data processing has been well pointed out. These limitations are critical considerations for anyone choosing to use R for a data science project, ensuring that users understand the challenges they may face.

Tool 2 Response Evaluation - SQL

Functionality of SQL:

I agree with this answer. It succinctly explains the main functions of SQL in data science, focusing on its ability to manage and query structured data in relational databases. SQL server functions are pre-built actions that perform calculations, manipulate data, and return results. At the most fundamental level, these functions simplify complex queries and automate repetitive tasks.[29] References to SQL's simple syntax and ability to perform statistical analyses directly in a database emphasized its usefulness and efficiency. The answer effectively communicates the primary role of SQL and its importance in working with structured data.

Usage of SQL in the Data Science Industry:

I agree with this answer. It provides a clear and comprehensive overview of the use of SQL in different industries, emphasizing its versatility and efficiency in managing structured data. SQL is extremely important for data science. It allows you to work with structured data stored in databases. As a data scientist, you need SQL to extract, manipulate, and analyze data from these databases. Big data tools like Hadoop and Spark also use SQL for processing structured data. Learning SQL skills like querying, joining tables, filtering, and aggregating data is crucial. Combining SQL with Python makes data management and analysis convenient without switching between languages. Whether dealing with relational databases or working with big data, mastering SQL is essential for any aspiring data scientist. It empowers you to wrangle, prepare, and gain insights from data effectively.[30]

Weaknesses or Limitations of SQL:

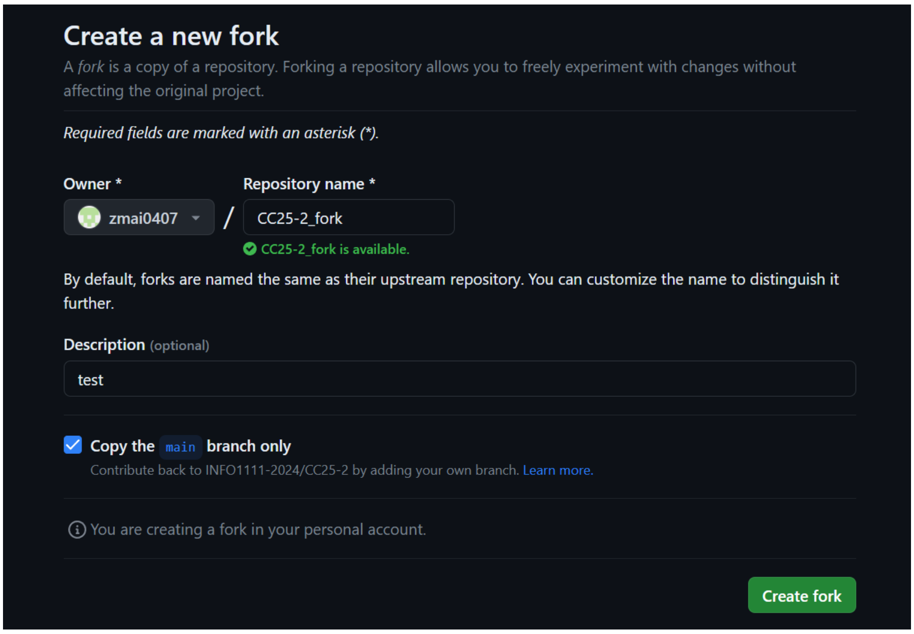
I agree with this answer. There are several weaknesses, including poor interface, cost inefficiency, partial control, and security concerns. Poor Interface: SQL has a poor interface, which makes things look very complex, even if they are not. Users find it difficult to deal with databases due to the difficulty of using the SQL interface. Cost Inefficiency: SQL Server Standard costs around $1,418/year. The high cost makes it difficult for some programmers to use it. Partial Control: SQL doesn't grant its users complete control over databases due to some hidden business rules. This can be a huge drawback for many. Security: Regardless of the SQL version, databases in SQL are constantly under threat as they hold huge amounts of sensitive data.[31]

Technical Skills (LaTeX and Git)

Git Skills: Forking and Special Files

Forking: Forking is a common form of collaborative development that allows developers to develop independently without affecting the original project. Forking gives you the freedom to experiment and improve your project, and when you're done, you can submit your changes to the original project via pull request for code merging.

There are two ways to create a fork. The one way is clicking the 'fork' button in Github and creating it.



Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Required fields are marked with an asterisk (*).

Owner *: zmai0407

Repository name *: CC25-2_fork

CC25-2_fork is available.

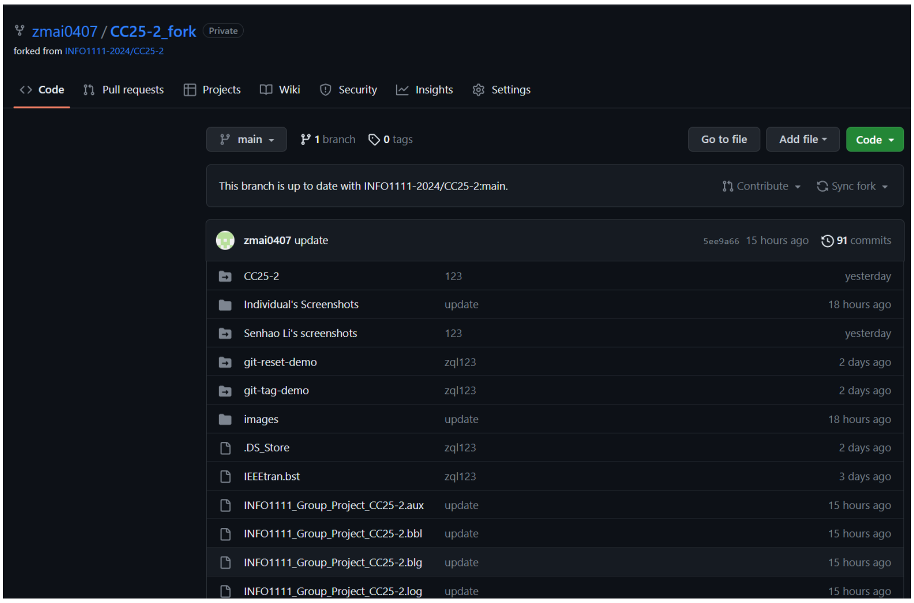
By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional): test

☑ Copy the `main` branch only

Contribute back to INFO1111-2024/CC25-2 by adding your own branch. Learn more.

You are creating a fork in your personal account.



zmai0407 / CC25-2_fork Private
forked from INFO1111-2024/CC25-2

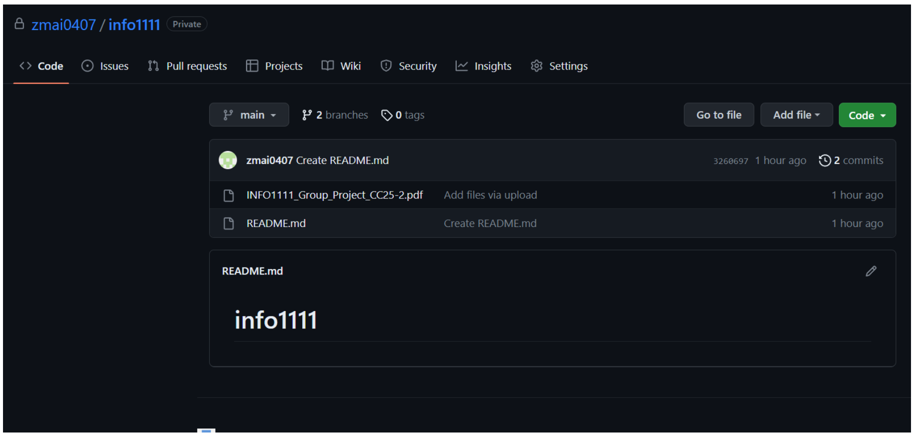
1 branch 0 tags

This branch is up to date with INFO1111-2024/CC25-2:main.

zmai0407 update 5ee9a66 15 hours ago 91 commits

CC25-2	123	yesterday
Individual's Screenshots	update	18 hours ago
Senhao Li's screenshots	123	yesterday
git-reset-demo	zql123	2 days ago
git-tag-demo	zql123	2 days ago
images	update	18 hours ago
.DS_Store	zql123	2 days ago
IEEEtran.bst	zql123	3 days ago
INFO1111_Group_Project_CC25-2.aux	update	15 hours ago
INFO1111_Group_Project_CC25-2.bbl	update	15 hours ago
INFO1111_Group_Project_CC25-2.blg	update	15 hours ago
INFO1111_Group_Project_CC25-2.log	update	15 hours ago

And another way is creating a fork in git.
Firstly, I create a new repository in github.



zmai0407 / info1111 Private

2 branches 0 tags

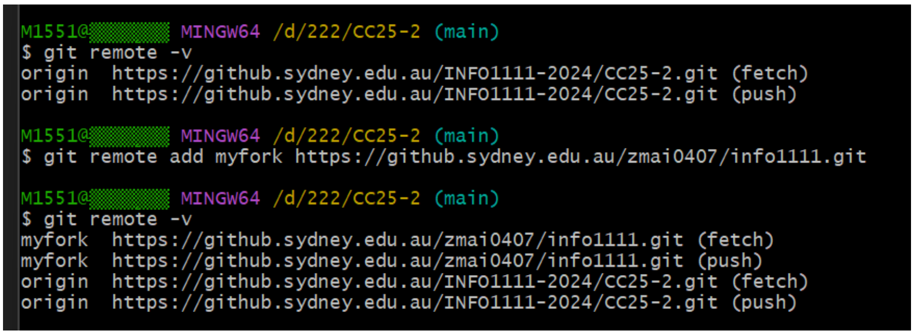
zmai0407 Create README.md 3260697 1 hour ago 2 commits

INFO1111_Group_Project_CC25-2.pdf	Add files via upload	1 hour ago
README.md	Create README.md	1 hour ago

README.md

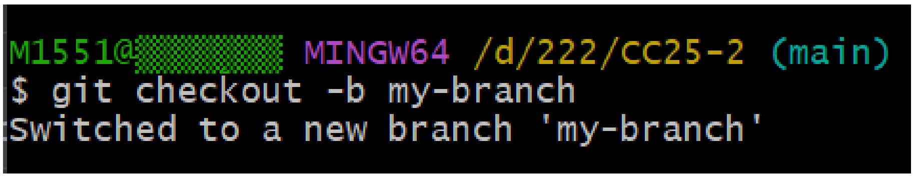
info1111

Secondly, I cloned the original repository locally and added a new remote repository.



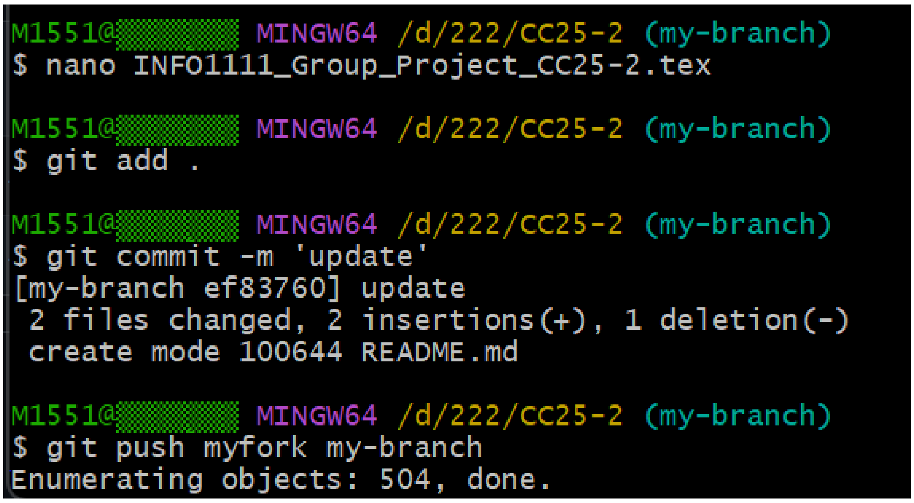
```
M1551@[redacted] MINGW64 /d/222/CC25-2 (main)
$ git remote -v
origin  https://github.sydney.edu.au/INFO1111-2024/CC25-2.git (fetch)
origin  https://github.sydney.edu.au/INFO1111-2024/CC25-2.git (push)

M1551@[redacted] MINGW64 /d/222/CC25-2 (main)
$ git remote add myfork https://github.sydney.edu.au/zmai0407/info1111.git

M1551@[redacted] MINGW64 /d/222/CC25-2 (main)
$ git remote -v
myfork  https://github.sydney.edu.au/zmai0407/info1111.git (fetch)
myfork  https://github.sydney.edu.au/zmai0407/info1111.git (push)
origin  https://github.sydney.edu.au/INFO1111-2024/CC25-2.git (fetch)
origin  https://github.sydney.edu.au/INFO1111-2024/CC25-2.git (push)
```

Then I created a new branch for independent development.



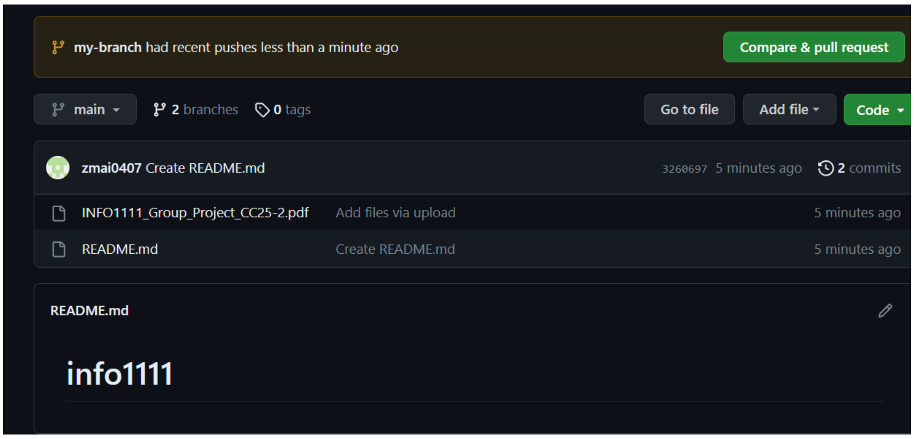
```
M1551@[redacted] MINGW64 /d/222/CC25-2 (main)
$ git checkout -b my-branch
Switched to a new branch 'my-branch'
```

After making changes and commits on the new branch,I pushed the new branch to the remote repository and committed the changes to the original project via Pull Request for code review and merge.



```
M1551@[redacted] MINGW64 /d/222/CC25-2 (my-branch)
$ nano INFO1111_Group_Project_CC25-2.tex

M1551@[redacted] MINGW64 /d/222/CC25-2 (my-branch)
$ git add .

M1551@[redacted] MINGW64 /d/222/CC25-2 (my-branch)
$ git commit -m 'update'
[my-branch ef83760] update
 2 files changed, 2 insertions(+), 1 deletion(-)
 create mode 100644 README.md

M1551@[redacted] MINGW64 /d/222/CC25-2 (my-branch)
$ git push myfork my-branch
Enumerating objects: 504, done.
```

Finally, I can find the compare & pull request in Github.



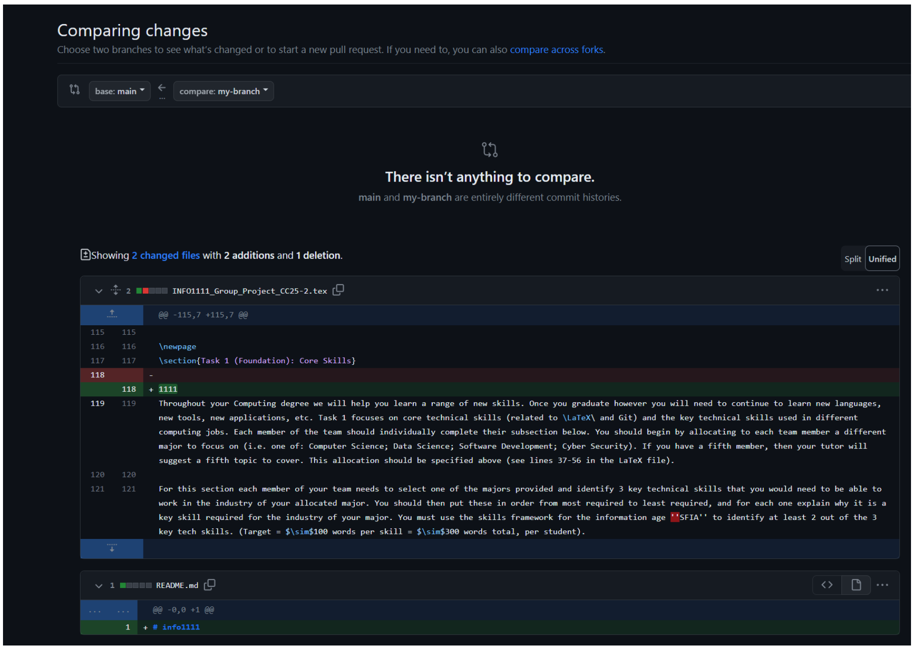
my-branch had recent pushes less than a minute ago

zmai0407 Create README.md 3260697 5 minutes ago 2 commits

INFO1111_Group_Project_CC25-2.pdf	Add files via upload	5 minutes ago
README.md	Create README.md	5 minutes ago

README.md

info1111



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

There isn't anything to compare.

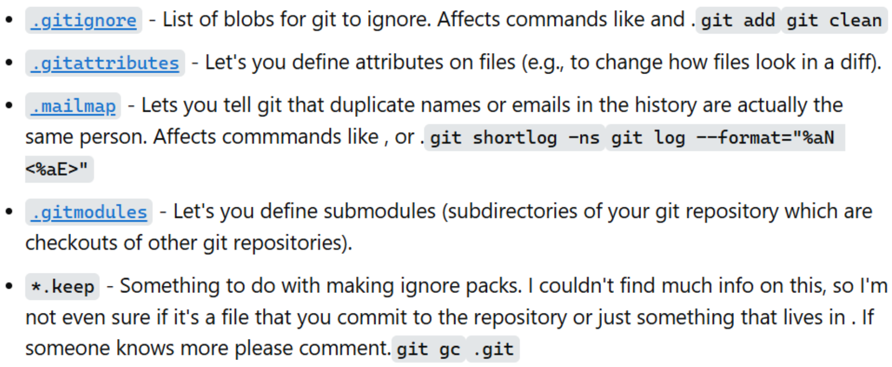
main and **my-branch** are entirely different commit histories.

Showing 2 changed files with **2 additions** and **1 deletion**.

```
2 INFO1111_Group_Project_CC25-2.tex
@@ -115,7 +115,7 @@

\newpage
\section{Task 1 (Foundation): Core Skills}
-
+ 1111
Throughout your Computing degree we will help you learn a range of new skills. Once you graduate however you will need to continue to learn new languages, new tools, new applications, etc. Task 1 focuses on core technical skills (related to \LaTeX\ and Git) and the key technical skills used in different computing jobs. Each member of the team should individually complete their subsection below. You should begin by allocating to each team member a different major to focus on (i.e. one of: Computer Science; Data Science; Software Development; Cyber Security). If you have a fifth member, then your tutor will suggest a fifth topic to cover. This allocation should be specified above (see lines 37-56 in the LaTeX file).

For this section each member of your team needs to select one of the majors provided and identify 3 key technical skills that you would need to be able to work in the industry of your allocated major. You should then put these in order from most required to least required, and for each one explain why it is a key skill required for the industry of your major. You must use the skills framework for the information age ''SFIA'' to identify at least 2 out of the 3 key tech skills. (Target = $\sim$100 words per skill = $\sim$300 words total, per student).
```

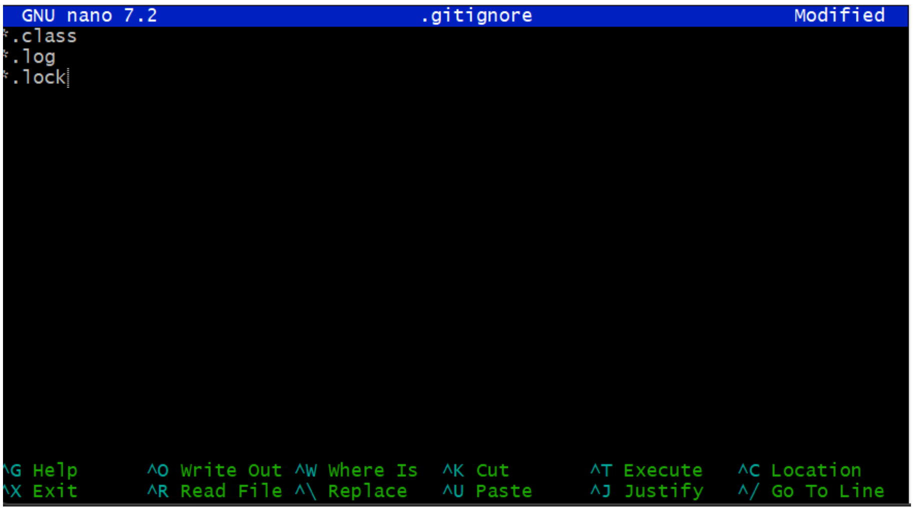
```
1 README.md
@@ -0,0 +1 @@
+ # info1111
```

Special Files: In Git, Special Files refer to configuration files with specific purposes that help you manage and control the behavior of your Git repository. Common special files include .gitignore and .gitattributes, but there are other special files:



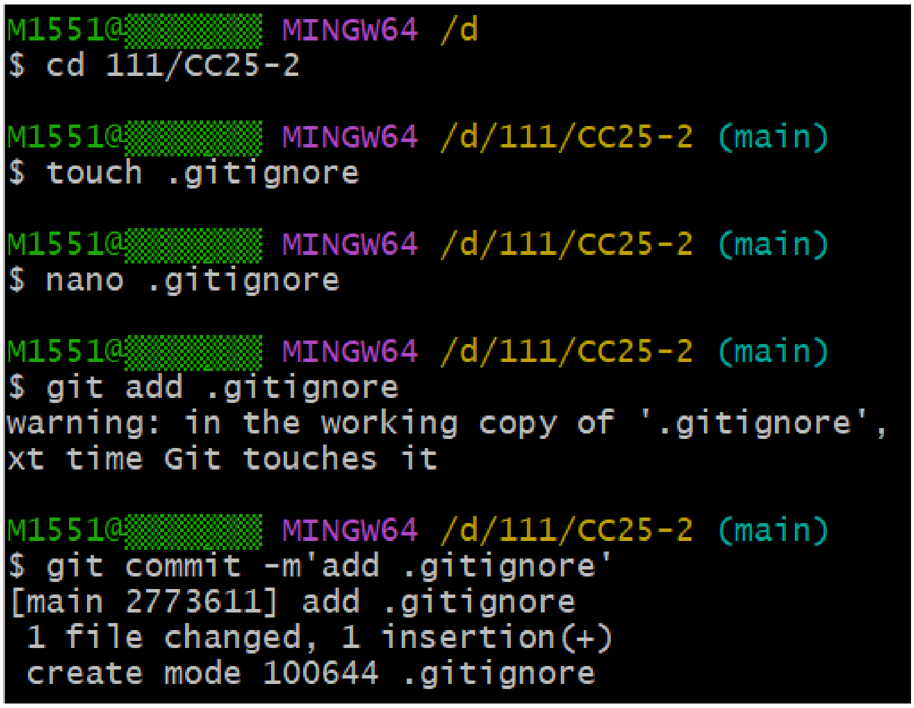
- `.gitignore` - List of blobs for git to ignore. Affects commands like and . `git add` `git clean`
- `.gitattributes` - Let's you define attributes on files (e.g., to change how files look in a diff).
- `.mailmap` - Lets you tell git that duplicate names or emails in the history are actually the same person. Affects commmands like , or . `git shortlog -ns` `git log --format="%aN <%aE>"`
- `.gitmodules` - Let's you define submodules (subdirectories of your git repository which are checkouts of other git repositories).
- `*.keep` - Something to do with making ignore packs. I couldn't find much info on this, so I'm not even sure if it's a file that you commit to the repository or just something that lives in . If someone knows more please comment. `git gc` `.git`

.gitignore: Specifies which files or directories Git should ignore and not include in version control. This is useful for excluding temporary files, compilation output, and sensitive information. -Ignore all .log, .class and .lock files



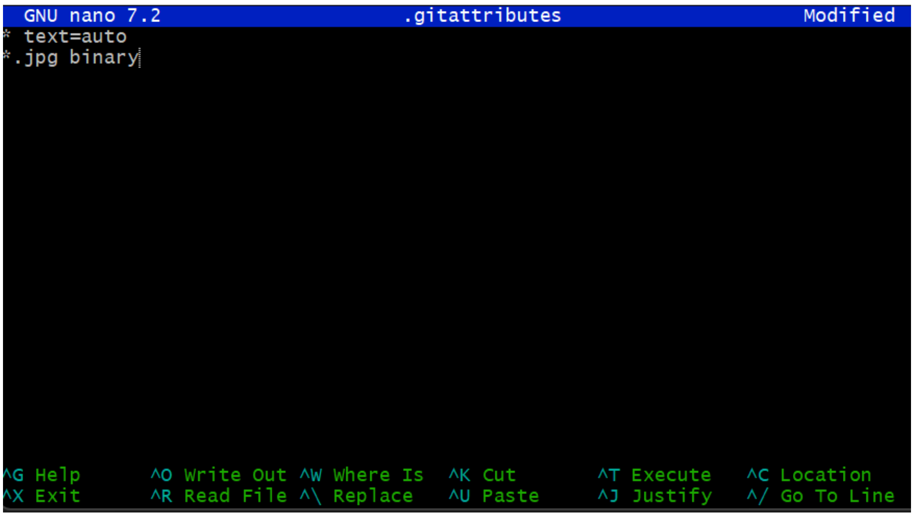
```
  GNU nano 7.2                    .gitignore                      Modified
*.class
*.log
*.lock
```

-add .gitignore to the staging area



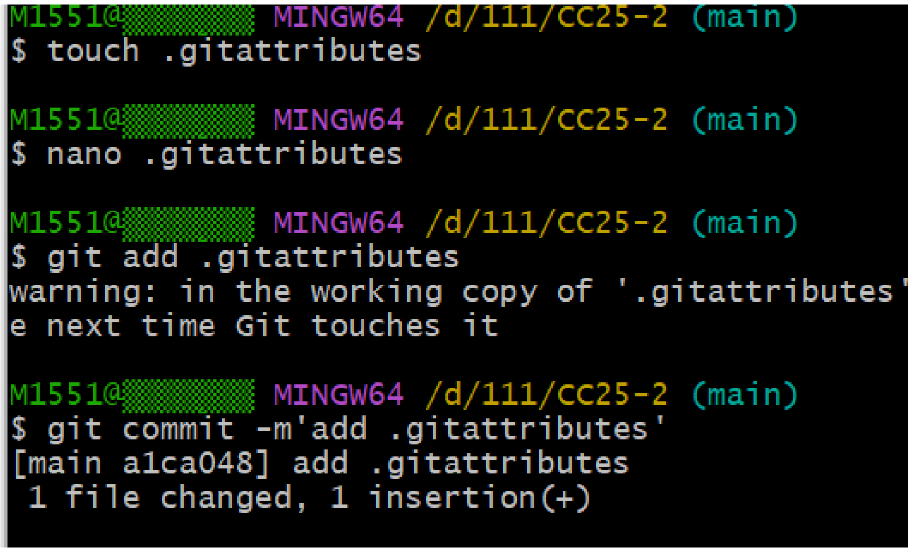
```
M1551@[redacted] MINGW64 /d
$ cd 111/CC25-2

M1551@[redacted] MINGW64 /d/111/CC25-2 (main)
$ touch .gitignore

M1551@[redacted] MINGW64 /d/111/CC25-2 (main)
$ nano .gitignore

M1551@[redacted] MINGW64 /d/111/CC25-2 (main)
$ git add .gitignore
warning: in the working copy of '.gitignore',
xt time Git touches it

M1551@[redacted] MINGW64 /d/111/CC25-2 (main)
$ git commit -m'add .gitignore'
[main 2773611] add .gitignore
 1 file changed, 1 insertion(+)
 create mode 100644 .gitignore
```

.gitattributes: Defines file attributes that control how Git handles these files. This can be used to specify line terminators for text files, merge policies, etc. -Set all text files to use LF line endings and Mark .jpg files as binary



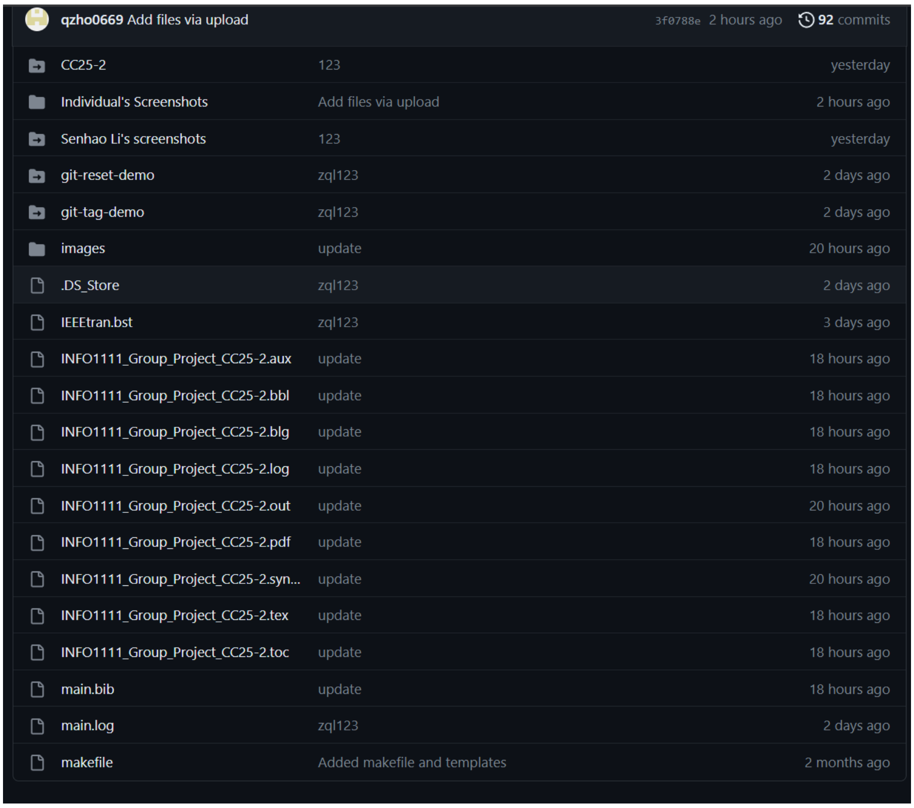
```
* text=auto
*.jpg binary
```

-add .gitattributes to the staging area



```
M1551@[redacted] MINGW64 /d/111/CC25-2 (main)
$ touch .gitattributes

M1551@[redacted] MINGW64 /d/111/CC25-2 (main)
$ nano .gitattributes

M1551@[redacted] MINGW64 /d/111/CC25-2 (main)
$ git add .gitattributes
warning: in the working copy of '.gitattributes'
e next time Git touches it

M1551@[redacted] MINGW64 /d/111/CC25-2 (main)
$ git commit -m'add .gitattributes'
[main a1ca048] add .gitattributes
 1 file changed, 1 insertion(+)
```

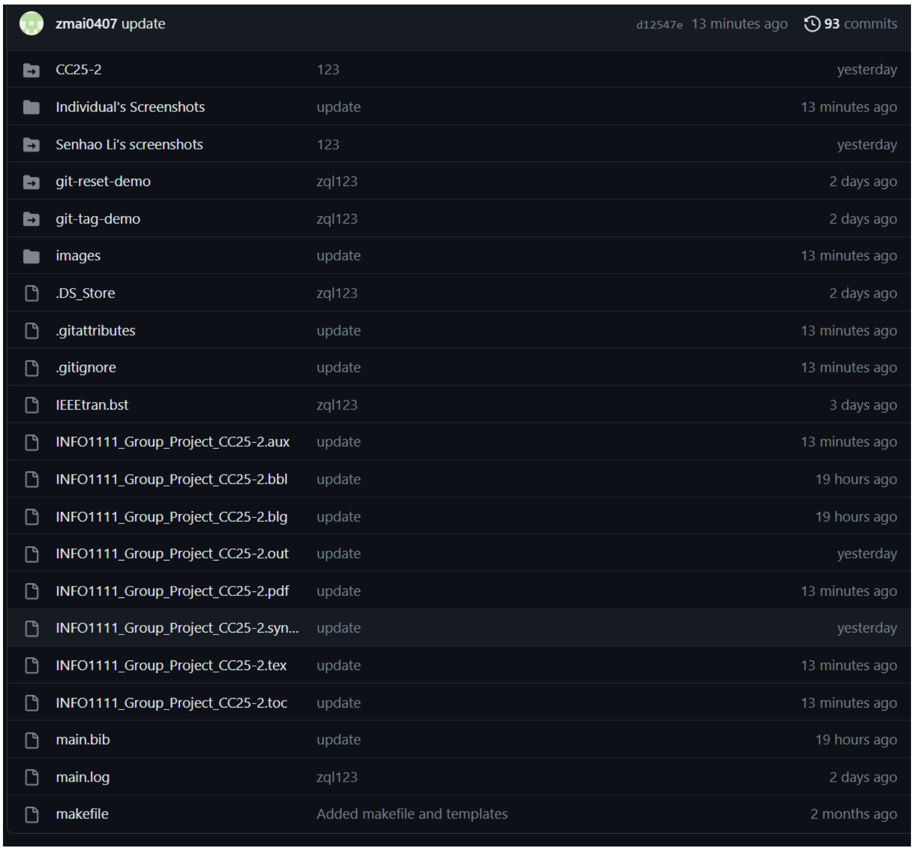
-the result of uploading with lots files



qzho0669 Add files via upload 3f0788e 2 hours ago **92** commits

CC25-2	123	yesterday
Individual's Screenshots	Add files via upload	2 hours ago
Senhao Li's screenshots	123	yesterday
git-reset-demo	zql123	2 days ago
git-tag-demo	zql123	2 days ago
images	update	20 hours ago
.DS_Store	zql123	2 days ago
IEEEtran.bst	zql123	3 days ago
INFO1111_Group_Project_CC25-2.aux	update	18 hours ago
INFO1111_Group_Project_CC25-2.bbl	update	18 hours ago
INFO1111_Group_Project_CC25-2.blg	update	18 hours ago
INFO1111_Group_Project_CC25-2.log	update	18 hours ago
INFO1111_Group_Project_CC25-2.out	update	20 hours ago
INFO1111_Group_Project_CC25-2.pdf	update	18 hours ago
INFO1111_Group_Project_CC25-2.syn...	update	20 hours ago
INFO1111_Group_Project_CC25-2.tex	update	18 hours ago
INFO1111_Group_Project_CC25-2.toc	update	18 hours ago
main.bib	update	18 hours ago
main.log	zql123	2 days ago
makefile	Added makefile and templates	2 months ago

-the .log file is not uploaded(be ignored).

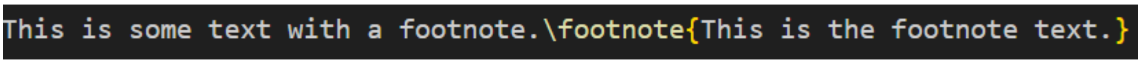


zmai0407 update d12547e 13 minutes ago 93 commits

CC25-2	123	yesterday
Individual's Screenshots	update	13 minutes ago
Senhao Li's screenshots	123	yesterday
git-reset-demo	zql123	2 days ago
git-tag-demo	zql123	2 days ago
images	update	13 minutes ago
.DS_Store	zql123	2 days ago
.gitattributes	update	13 minutes ago
.gitignore	update	13 minutes ago
IEEEtran.bst	zql123	3 days ago
INFO1111_Group_Project_CC25-2.aux	update	13 minutes ago
INFO1111_Group_Project_CC25-2.bbl	update	19 hours ago
INFO1111_Group_Project_CC25-2.blg	update	19 hours ago
INFO1111_Group_Project_CC25-2.out	update	yesterday
INFO1111_Group_Project_CC25-2.pdf	update	13 minutes ago
INFO1111_Group_Project_CC25-2.syn...	update	yesterday
INFO1111_Group_Project_CC25-2.tex	update	13 minutes ago
INFO1111_Group_Project_CC25-2.toc	update	13 minutes ago
main.bib	update	19 hours ago
main.log	zql123	2 days ago
makefile	Added makefile and templates	2 months ago

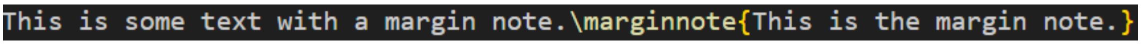
LaTeX Skill: Footnotes/margin notes and creating new environments

Footnotes: Footnotes allow for the addition of supplementary information or references at the bottom of the page, providing readers with additional context without interrupting the main text flow. This is accomplished using the `\footnote{}` command, where the text inside the braces appears as a numbered footnote at the bottom of the page. Just like that:



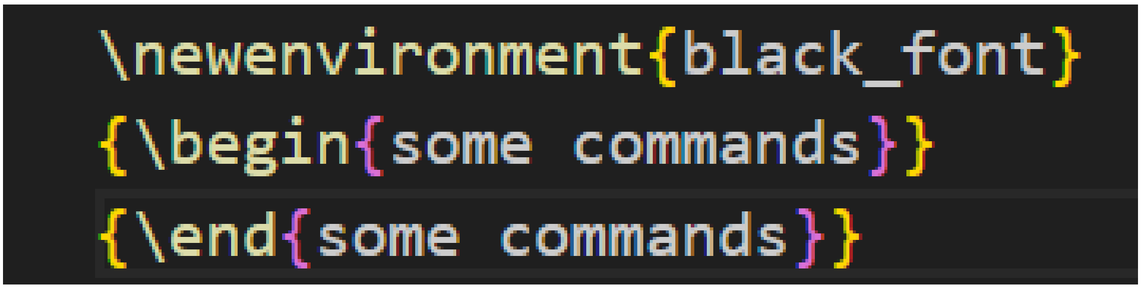
```
This is some text with a footnote.\footnote{This is the footnote text.}
```

Margin notes: Margin notes, on the other hand, offer a way to add brief comments or explanations in the page margins, making them visible alongside the main text. This can be achieved with the `\marginnote{}` command, positioning the note in the right margin by default.



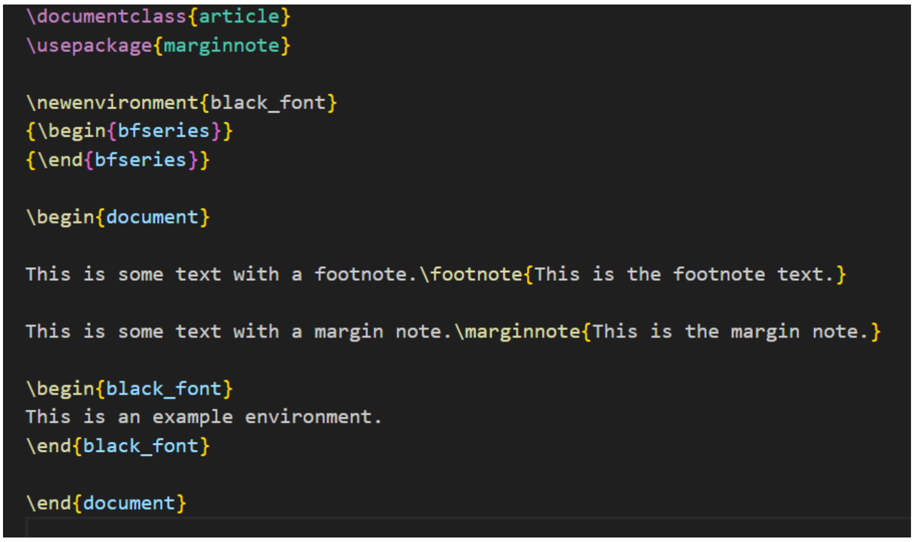
```
This is some text with a margin note.\marginnote{This is the margin note.}
```

Creating a new environment: Additionally, creating new environments in LaTeX allows for the definition of custom formatting for specific sections of text. This is done using the `\newenvironment{name}{begin-code}{end-code}` command, where `name` is the name of the new environment, and `begin-code` and `end-code` specify the formatting to apply at the start and end of the environment, respectively. These tools enhance document readability and organization, as illustrated below:

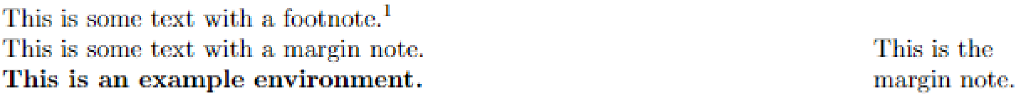


```
\newenvironment{black_font}
{\begin{some commands}}
{\end{some commands}}
```

And I put all the commands together (here is the example of bfseries):



```
\documentclass{article}
\usepackage{marginnote}

\newenvironment{black_font}
{\begin{bfseries}}
{\end{bfseries}}

\begin{document}

This is some text with a footnote.\footnote{This is the footnote text.}

This is some text with a margin note.\marginnote{This is the margin note.}

\begin{black_font}
This is an example environment.
\end{black_font}

\end{document}
```



This is some text with a footnote.[1]
This is some text with a margin note.
This is an example environment.

This is the
margin note.

2.3. Tools and Skills for SW Development: ZHONG, QILONG

Part A: Exploration of tech tools

Tool Identification

In the field of software development, I have chosen Xcode and pycharm, two very practical tools that are widely used by software developers. I am certain that this is a software that software development professionals often need, and due to its convenience and powerful functionality, it has been widely used in practice.

Tool 1 Questions - Xcode

Question 1: What is the main function of Xcode in the field of software development, using 250 words or less?

Xcode is an Integrated Development Environment (IDE) developed by Apple Inc. specifically for creating applications for Apple's platforms, including macOS, iOS, watchOS, and tvOS. Its primary function in software development is to provide a comprehensive suite of tools that facilitate the entire development process from coding to deployment.

Xcode's sophisticated code editor supports languages such as Swift and Objective-C, offering features like syntax highlighting, code completion, and real-time error checking, which significantly enhance coding efficiency and accuracy. The built-in debugger allows developers to inspect code execution in real-time, set breakpoints, and monitor variable states, enabling effective troubleshooting and optimization.

A standout feature of Xcode is its Interface Builder, a visual design tool that allows developers to create and test user interfaces using a drag-and-drop interface. This tool streamlines the design process and ensures a cohesive user experience across different devices.

Additionally, Xcode includes performance analysis tools like Instruments, which help developers profile and optimize their applications, ensuring they run smoothly and efficiently. The IDE also integrates extensive documentation and resources, providing developers with the necessary information and support throughout the development lifecycle.

Overall, Xcode's main function is to offer an all-encompassing environment that supports the development, debugging, and deployment of applications within the Apple ecosystem, making it an indispensable tool for developers targeting Apple platforms.

Question 2: How is Xcode used in the software development industry, in 250 words or less?

Xcode is extensively used in the software development industry for creating applications across Apple's ecosystem, including macOS, iOS, watchOS, and tvOS. It serves as the primary development environment for developers targeting these platforms due to its comprehensive toolset and seamless integration with Apple's software and hardware.

Developers use Xcode to write code in Swift and Objective-C, leveraging its advanced code editor for efficient and error-free programming. The IDE's real-time error detection and code completion features significantly speed up the coding process, allowing developers to quickly identify and fix issues.

Xcode's Interface Builder is a key tool for designing user interfaces. It enables developers to create visually appealing and functional UIs through a drag-and-drop interface, ensuring a consistent user experience across different Apple devices. This visual approach to UI design helps streamline the development process and reduces the time required to create intuitive interfaces.

The built-in debugger in Xcode is crucial for testing and troubleshooting applications. Developers can set breakpoints, monitor variable states, and step through code to identify and resolve bugs effectively. Additionally, Xcode's Instruments tool provides performance analysis, helping developers optimize their applications for better performance and resource management.

In the industry, Xcode is essential for deploying applications to the Apple App Store. It provides all the necessary tools for compiling, testing, and submitting apps, ensuring they meet Apple's stringent quality standards.

Overall, Xcode's integration of coding, debugging, UI design, and performance optimization tools makes it an indispensable environment for professional developers focused on Apple platforms.

Question 3: Does Xcode have any weaknesses or limitations, in 250 words or less?

Yes, Xcode has several weaknesses and limitations. Xcode is exclusive to macOS, meaning it can only be used on Apple computers. This limits accessibility for developers using other operating systems, such as Windows or Linux, requiring them to invest in

Apple hardware if they want to develop applications for Apple's platforms.

Xcode is resource-intensive, demanding significant memory and processing power to run smoothly. On older or less powerful machines, Xcode can be slow and unresponsive, hindering the development process and reducing productivity.

For beginners, Xcode has a steep learning curve. Its extensive range of features and tools can be overwhelming, making it difficult for new developers to navigate and utilize effectively. While comprehensive documentation is available, the complexity of the IDE remains a challenge for those unfamiliar with Apple's development environment.

Performance and stability issues are common complaints. The IDE can sometimes crash or exhibit bugs, especially with new releases or updates. These issues disrupt the development workflow and require time to troubleshoot and resolve.

Xcode's integration with non-Apple technologies and platforms is limited. This can be a disadvantage for developers working with cross-platform tools or frameworks, as they may encounter compatibility issues or lack of support within the Xcode environment.

Overall, while Xcode is a powerful and essential tool for developing applications within the Apple ecosystem, its platform exclusivity, resource demands, steep learning curve, and occasional stability issues are notable limitations.

Tool 2 Questions - Pycharm

Question 1: What is the main function of Pycharm in the field of software development, using 250 words or less?

PyCharm is an Integrated Development Environment (IDE) specifically designed for Python programming, developed by JetBrains. Its primary function in software development is to provide a comprehensive suite of tools that facilitate efficient coding, debugging, and project management for Python developers.

PyCharm offers an intelligent code editor that supports advanced features such as syntax highlighting, code completion, and real-time error detection, which significantly enhance coding accuracy and efficiency. The IDE also includes a powerful debugger that allows developers to inspect and control code execution, set breakpoints, and evaluate expressions, enabling effective troubleshooting and optimization.

One of PyCharm's key functionalities is its integration with popular web frameworks such as Django and Flask. This integration simplifies web development by providing dedicated tools, templates, and support for these frameworks, making it easier to develop, test, and deploy web applications.

PyCharm also includes built-in tools for database management and querying, which streamline interactions with databases directly from the IDE. This feature is particularly useful for data-intensive applications and projects that require seamless database integration.

Additionally, PyCharm supports version control systems like Git, allowing developers to manage their code repositories, track changes, and collaborate with team members efficiently. The IDE's comprehensive support for testing frameworks helps ensure code quality by facilitating the writing, running, and debugging of unit tests.

Overall, PyCharm's main function is to provide a robust, all-encompassing development environment that enhances productivity, code quality, and project management for Python developers.

Question 2: How is Pycharm used in the software development industry, in 250 words or less?

PyCharm is widely used in the software development industry for various types of

Python projects, including web development, data science, machine learning, automation, and scripting. Its robust set of features and tools makes it a popular choice among developers and organizations aiming to improve coding efficiency, maintain high code quality, and streamline project workflows.

In web development, PyCharm's integration with popular frameworks such as Django, Flask, and Pyramid simplifies the development process. It provides built-in templates, project scaffolding, and debugging tools that help developers quickly build, test, and deploy web applications. PyCharm's support for front-end technologies like HTML, CSS, and JavaScript further enhances its utility for full-stack development.

For data science and machine learning, PyCharm offers excellent support for libraries such as NumPy, pandas, Matplotlib, and TensorFlow. Its scientific mode provides an interactive environment similar to Jupyter Notebooks, allowing data scientists to analyze data, visualize results, and experiment with machine learning models efficiently.

PyCharm is also valuable for automation and scripting tasks. Its intelligent code editor, combined with powerful debugging and testing tools, enables developers to write and maintain scripts that automate repetitive tasks, improving productivity and reducing errors.

Additionally, PyCharm's built-in version control system (VCS) integration, particularly with Git, facilitates efficient collaboration among development teams. It allows developers to manage code repositories, track changes, and resolve conflicts seamlessly within the IDE.

Overall, PyCharm is a versatile tool that supports a wide range of Python development activities, making it indispensable for developers and organizations seeking to optimize their software development processes.

Question 3: Does Pycharm have any weaknesses or limitations, in 250 words or less?

Yes, PyCharm has several weaknesses and limitations. One of the primary issues is its resource intensity. PyCharm can be demanding on system resources, particularly when working with large projects or multiple plugins. This can lead to slower performance on less powerful machines, which may hinder productivity.

Another limitation is the cost associated with the professional edition. While PyCharm offers a free community edition, it lacks many advanced features available in the paid version. These features, such as full support for web development frameworks and advanced database tools, are crucial for many professional developers. The cost of the professional edition might be a barrier for individual developers or small teams with limited budgets.

For beginners, PyCharm's extensive feature set can be overwhelming. The IDE's complexity may pose a steep learning curve for new developers, making it challenging to navigate and utilize its full potential effectively. Despite comprehensive documentation and tutorials, the initial setup and configuration can be daunting.

Additionally, while PyCharm excels in Python development, it is less suitable for projects involving multiple languages or non-Python technologies. Developers working on polyglot projects might find PyCharm's support for other languages limited compared to more general-purpose IDEs.

Lastly, occasional stability issues and bugs can disrupt the development workflow. Although JetBrains regularly updates PyCharm to address these issues, they can still affect user experience and productivity.

Overall, while PyCharm is a powerful and feature-rich IDE for Python development, its resource demands, cost, complexity for beginners, limited multi-language support, and occasional stability issues are notable limitations.

Part B: Analysis

Tool 1 Response Evaluation - Xcode

Functionality of Xcode:

I agree with ChatGPT's description of Xcode as an Integrated Development Environment (IDE) designed for Apple platforms like macOS, iOS, watchOS, and tvOS. The explanation of its primary function, including coding, debugging, and deployment, is thorough and highlights key features like the code editor, debugger, Interface Builder, and performance tools.[32]. However, it could mention the potential limitations for developers who are used to more flexible or less tightly integrated environments. While Xcode's all-encompassing nature is a strength, it might also mean less flexibility compared to modular development tools.

Usage of Xcode in the Software Development:

The provided answer effectively captures how Xcode is used across the Apple ecosystem, from coding in Swift and Objective-C to designing interfaces with Interface Builder and deploying apps to the App Store. This explanation aligns well with industry practices. However, it could have addressed how the close integration with Apple's hardware and software, while advantageous, can also lead to challenges when new updates introduce bugs or require significant changes.[33] Additionally, the response could have considered how Xcode's strong focus on Apple's ecosystem might limit its adoption in more diverse or cross-platform development teams.

Weaknesses or limitations of Xcode:

I agree with the detailed identification of Xcode's weaknesses, such as its macOS exclusivity, high resource demands, steep learning curve, and performance issues.[34] These points are all valid and reflect common concerns among developers. However, the answer could have been more nuanced by discussing how Apple's continuous updates and improvements to Xcode address some of these issues over time. It might also have been beneficial to consider how these limitations impact different types of developers differently, with beginners potentially struggling more than experienced developers who can better navigate the complexities of the tool.

Overall, while the answers are comprehensive and accurate, incorporating a bit more critical analysis and considering different perspectives would make the assessment even more insightful.

Tool 2 Response Evaluation - Pycharm

Functionality of Pycharm:

I agree with the answer provided by ChatGPT. It accurately describes PyCharm as an Integrated Development Environment (IDE) designed specifically for Python programming. The explanation highlights its comprehensive suite of tools for efficient coding, debugging, and project management, which are key aspects of any robust IDE.[35] The mention of advanced features like syntax highlighting, code completion, real-time error detection, and powerful debugging tools is appropriate. Additionally, the integration with popular web frameworks and support for database management, version control systems, and testing frameworks underline PyCharm's versatility and effectiveness in enhancing productivity and code quality. However, the answer could further emphasize how PyCharm's focus on Python can limit its use in multi-language projects, reflecting a potential trade-off between specialization and versatility.

Usage of Pycharm in the Software Development:
The answer provides a comprehensive overview of PyCharm's use across various Python projects, including web development, data science, machine learning, automation, and scripting. It rightly points out PyCharm's integration with web frameworks like Django and Flask, support for data science libraries, and utilities for automation tasks.[36] The mention of built-in version control system (VCS) integration and its benefits for team collaboration is also accurate. However, it could have added that while PyCharm excels in Python-centric environments, its utility might be limited in more heterogeneous tech stacks. This perspective highlights the tool's strengths and the specific contexts where it shines, offering a balanced view.

Weaknesses or limitations of Pycharm:
I agree with the identified weaknesses and limitations of PyCharm. The response correctly highlights the resource-intensive nature of PyCharm, which can affect performance on less powerful machines. The cost barrier associated with the professional edition and the steep learning curve for beginners due to its extensive feature set are valid points. The answer also accurately notes PyCharm's limited suitability for multi-language projects compared to general-purpose IDEs and mentions occasional stability issues.[37] However, it could be more nuanced by acknowledging that regular updates from JetBrains often address these stability concerns, though not always perfectly. This assessment provides a thorough and critical view of PyCharm's limitations while considering ongoing improvements.

amsmath graphicx

Technical Skills (LaTeX and Git)

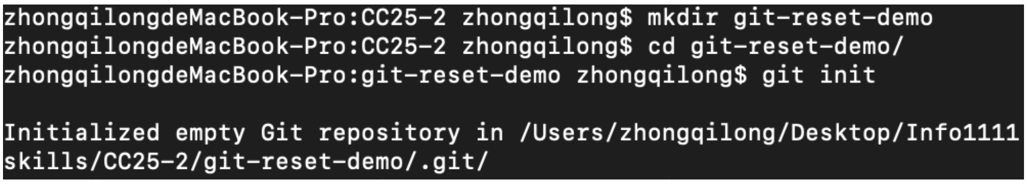
Git Skill: Resetting and Tags

Resetting:

'git reset' is a powerful command in Git used to reset the head of the current branch to the specified commit. This command can selectively change the index and working directory, helping developers to flexibly manipulate code history and working status in different development scenarios.

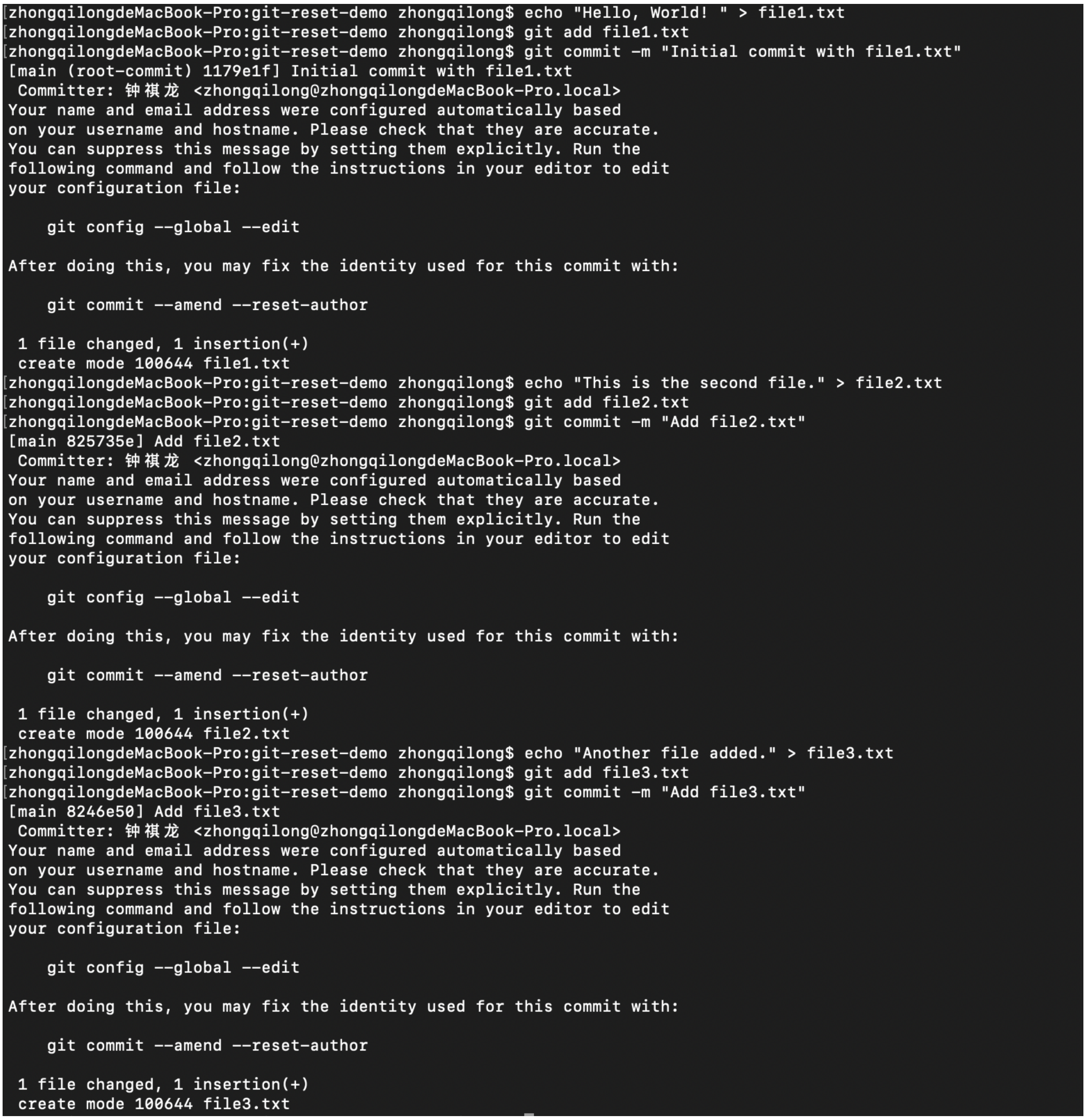
The following are the three reset modes and corresponding examples:

- First, create a new repository at the corresponding location



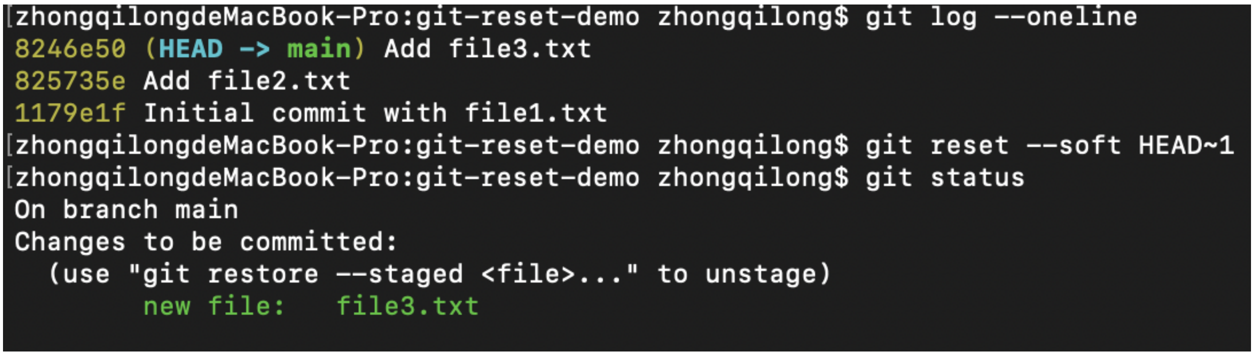
```
zhongqilongdeMacBook-Pro:CC25-2 zhongqilong$ mkdir git-reset-demo
zhongqilongdeMacBook-Pro:CC25-2 zhongqilong$ cd git-reset-demo/
zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ git init

Initialized empty Git repository in /Users/zhongqilong/Desktop/Info1111
skills/CC25-2/git-reset-demo/.git/
```

- Create and submit some files



```
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ echo "Hello, World! " > file1.txt
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ git add file1.txt
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ git commit -m "Initial commit with file1.txt"
[main (root-commit) 1179e1f] Initial commit with file1.txt
 Committer: 钟祺龙 <zhongqilong@zhongqilongdeMacBook-Pro.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ echo "This is the second file." > file2.txt
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ git add file2.txt
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ git commit -m "Add file2.txt"
[main 825735e] Add file2.txt
 Committer: 钟祺龙 <zhongqilong@zhongqilongdeMacBook-Pro.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+)
 create mode 100644 file2.txt
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ echo "Another file added." > file3.txt
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ git add file3.txt
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ git commit -m "Add file3.txt"
[main 8246e50] Add file3.txt
 Committer: 钟祺龙 <zhongqilong@zhongqilongdeMacBook-Pro.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+)
 create mode 100644 file3.txt
```

- Soft mode

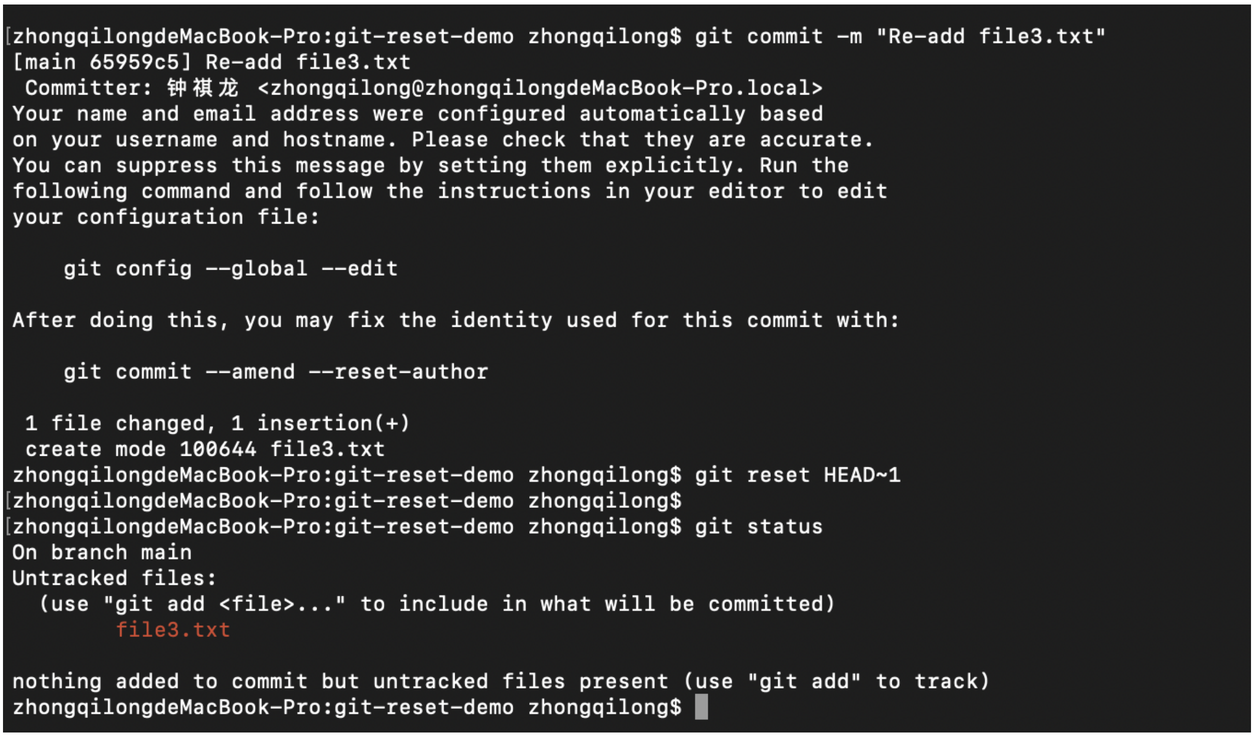


```
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ git log --oneline
8246e50 (HEAD -> main) Add file3.txt
825735e Add file2.txt
1179e1f Initial commit with file1.txt
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ git reset --soft HEAD~1
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file3.txt
```

’git log –oneline’ : View submission history
’git reset –soft HEAD 1’ : Use – soft mode to rollback to the previous submission

’git status’ : Check the Git status, file3.txt should be in the temporary storage area
Details: In this step, file3.txt remains in the temporary storage area and can be resubmitted or modified.

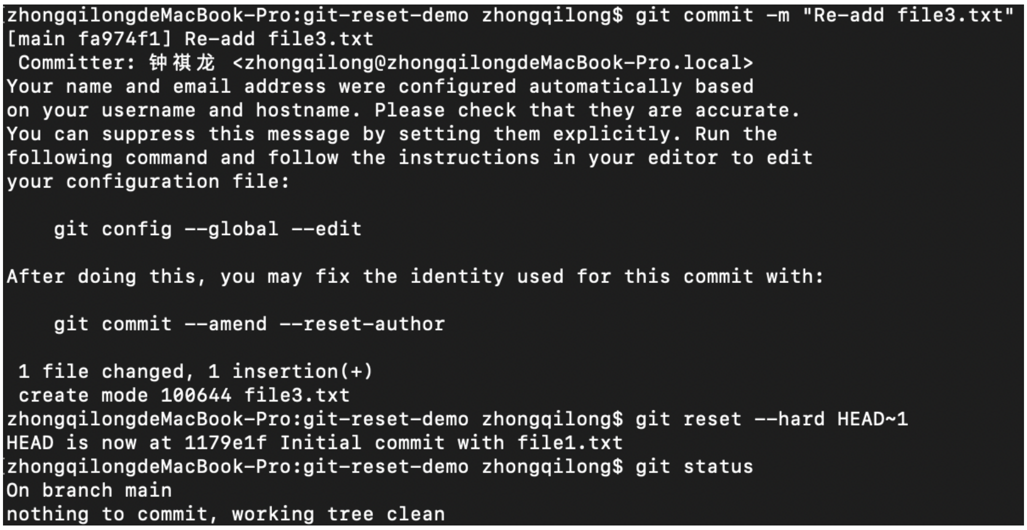
- Mixed mode



```
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ git commit -m "Re-add file3.txt"
[main 65959c5] Re-add file3.txt
 Committer: 钟祺龙 <zhongqilong@zhongqilongdeMacBook-Pro.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+)
 create mode 100644 file3.txt
zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ git reset HEAD~1
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file3.txt

nothing added to commit but untracked files present (use "git add" to track)
zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ 
```

’git commit -m "Re-add file3.txt" ’ : Resubmit file3. txt
’git reset HEAD 1’ : Use – mixed mode to rollback to the previous submission
’git status’ : Check the Git status, file3.txt should be in the working directory but not temporarily stored
In this step, file3.txt is still in the working directory, but needs to be added back to the staging area before it can be submitted.[38] And this is the default mode.

- Hard mode



```
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ git commit -m "Re-add file3.txt"
[main fa974f1] Re-add file3.txt
 Committer: 钟祺龙 <zhongqilong@zhongqilongdeMacBook-Pro.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+)
 create mode 100644 file3.txt
zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ git reset --hard HEAD~1
HEAD is now at 1179e1f Initial commit with file1.txt
[zhongqilongdeMacBook-Pro:git-reset-demo zhongqilong$ git status
On branch main
nothing to commit, working tree clean
```

’git reset –hard HEAD 1’ : Use – hard mode to rollback to the previous submission
’git status’ : Check the Git status, file3.txt should disappear from the working

directory

In this step, file3.txt will disappear from the working directory because the – hard mode discards all uncommitted changes.

- Summary

  1–Soft: Only reset the HEAD pointer, keeping all file changes in the staging area.
  2–Mixed: Reset HEAD and indexes, keep file changes in the working directory.
  3–Hard: Reset HEAD, indexes, and working directory, discard all uncommitted changes.

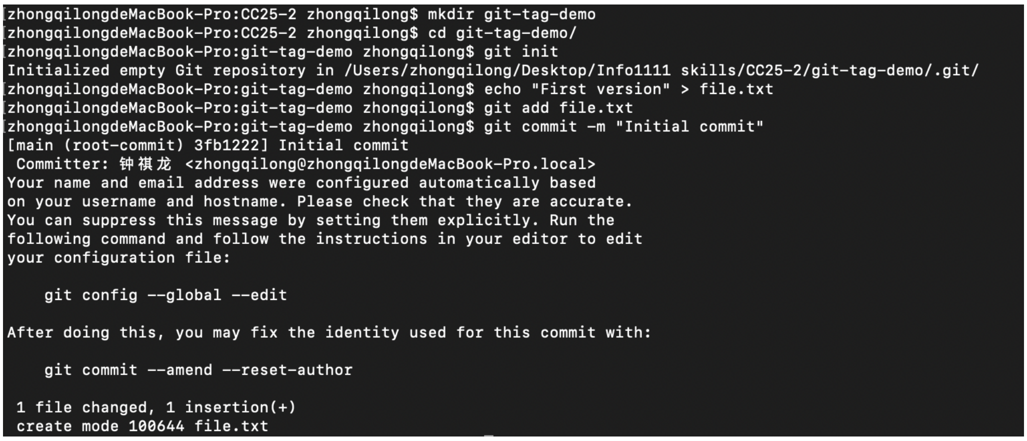
article amsmath graphicx

Tags:

Tags in Git are a feature used to mark specific submissions, typically used to mark version releases. Tags provide us with a way to create snapshots of specific commit points, making it easy for us to return to that specific commit.

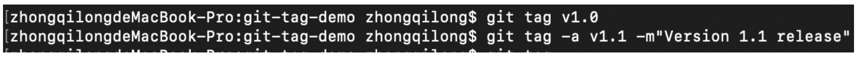
The following is a demonstration of tags and related examples:

- Preparation: Create a new Git repository and initialize it, create files and submit them



```
[zhongqilongdeMacBook-Pro:CC25-2 zhongqilong$ mkdir git-tag-demo
[zhongqilongdeMacBook-Pro:CC25-2 zhongqilong$ cd git-tag-demo/
[zhongqilongdeMacBook-Pro:git-tag-demo zhongqilong$ git init
Initialized empty Git repository in /Users/zhongqilong/Desktop/Info1111 skills/CC25-2/git-tag-demo/.git/
[zhongqilongdeMacBook-Pro:git-tag-demo zhongqilong$ echo "First version" > file.txt
[zhongqilongdeMacBook-Pro:git-tag-demo zhongqilong$ git add file.txt
[zhongqilongdeMacBook-Pro:git-tag-demo zhongqilong$ git commit -m "Initial commit"
[main (root-commit) 3fb1222] Initial commit
 Committer: 钟祺龙 <zhongqilong@zhongqilongdeMacBook-Pro.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+)
 create mode 100644 file.txt
```

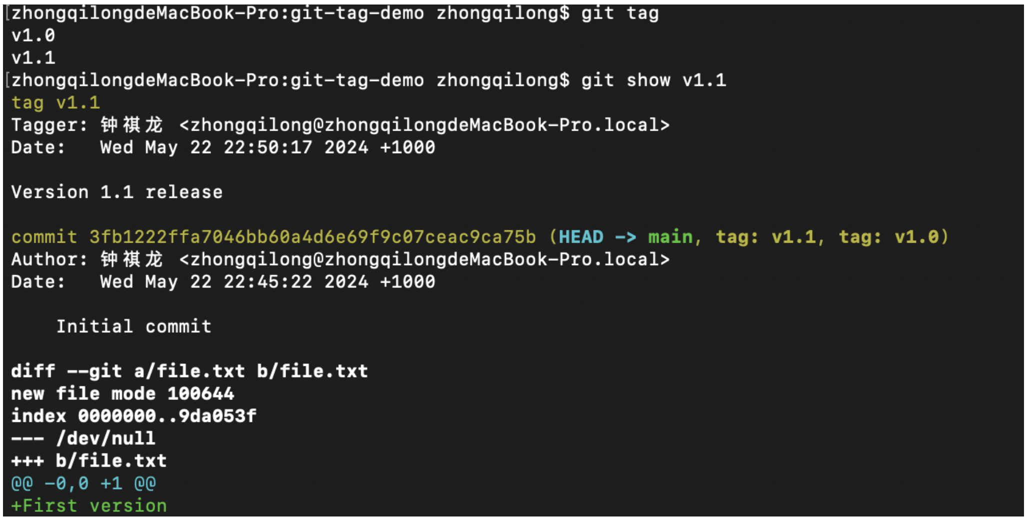
- Create lightweight and annotation tags separately:



```
[zhongqilongdeMacBook-Pro:git-tag-demo zhongqilong$ git tag v1.0
[zhongqilongdeMacBook-Pro:git-tag-demo zhongqilong$ git tag -a v1.1 -m"Version 1.1 release"
```

  'git tag v1.0' : Create lightweight tag
  'git tag -a v1.1 -m "Version 1.1 release" ' : Create annotation tag

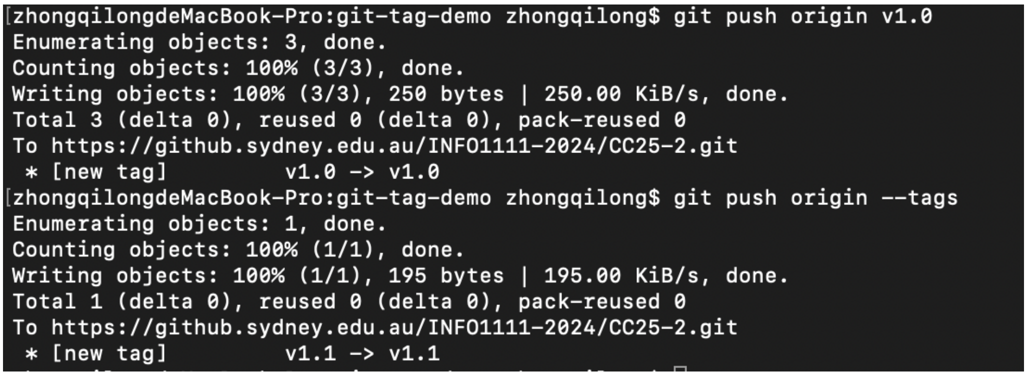
- View tags:



```
[zhongqilongdeMacBook-Pro:git-tag-demo zhongqilong$ git tag
v1.0
v1.1
[zhongqilongdeMacBook-Pro:git-tag-demo zhongqilong$ git show v1.1
tag v1.1
Tagger: 钟祺龙 <zhongqilong@zhongqilongdeMacBook-Pro.local>
Date:   Wed May 22 22:50:17 2024 +1000

Version 1.1 release

commit 3fb1222ffa7046bb60a4d6e69f9c07ceac9ca75b (HEAD -> main, tag: v1.1, tag: v1.0)
Author: 钟祺龙 <zhongqilong@zhongqilongdeMacBook-Pro.local>
Date:   Wed May 22 22:45:22 2024 +1000

    Initial commit

diff --git a/file.txt b/file.txt
new file mode 100644
index 0000000..9da053f
--- /dev/null
+++ b/file.txt
@@ -0,0 +1 @@
+First version
```

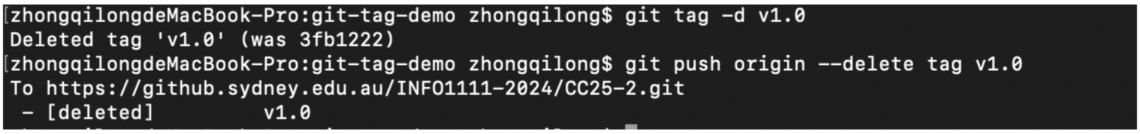
'git tag' and 'git show v1.1' : View tags

- Push tags to Repo:



```
[zhongqilongdeMacBook-Pro:git-tag-demo zhongqilong$ git push origin v1.0
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 250 bytes | 250.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.sydney.edu.au/INFO1111-2024/CC25-2.git
 * [new tag]         v1.0 -> v1.0
[zhongqilongdeMacBook-Pro:git-tag-demo zhongqilong$ git push origin --tags
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 195 bytes | 195.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.sydney.edu.au/INFO1111-2024/CC25-2.git
 * [new tag]         v1.1 -> v1.1
```

- Delete tags:



```
[zhongqilongdeMacBook-Pro:git-tag-demo zhongqilong$ git tag -d v1.0
Deleted tag 'v1.0' (was 3fb1222)
[zhongqilongdeMacBook-Pro:git-tag-demo zhongqilong$ git push origin --delete tag v1.0
To https://github.sydney.edu.au/INFO1111-2024/CC25-2.git
 - [deleted]         v1.0
```

'git tag -d v1.0' and 'git push origin –delete tag v1.0' : Delete tags

- Summary

  Tags provide an easy way to mark specific commit points in Git, which is particularly useful for version releases. Using tags, you can easily manage and revert to different versions of a project, ensuring each version has a clear reference point.[39] With lightweight tags and annotated tags, you can choose the appropriate method to create and manage your tags based on your needs.

LaTeX Skill: Floating figures and editing style sheets

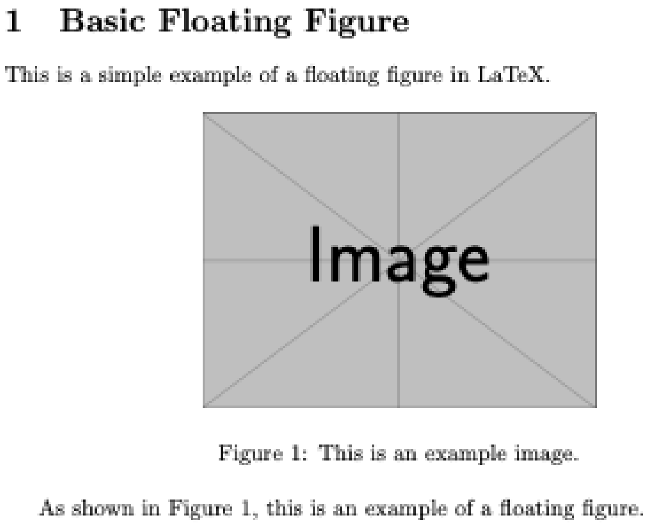
Floating figures:

- Introduction

  In LaTeX, floating bodies (such as figures and tables) allow you to place content in the best position to avoid layout issues. The commonly used floating body commands include the `figure` environment.

- Basic Example

  Here is a simple example of a floating figure in LaTeX:



1 Basic Floating Figure

This is a simple example of a floating figure in LaTeX.

Figure 1: This is an example image.

As shown in Figure 1, this is an example of a floating figure.

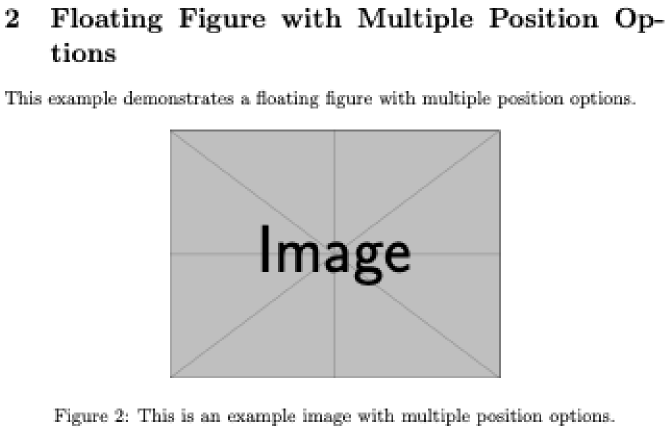
Code:

```
\documentclass{article}
\usepackage{graphicx}

\begin{document}

\item{Introduction}
This is a simple example of a floating figure in LaTeX.

\begin{figure}[h]
    \centering
    \includegraphics[width=0.5\textwidth]{example-image}
    \caption{This is an example image.}
    \label{fig:example}
\end{figure}

As shown in Figure \ref{fig:example}, this is an example of a floating figure.
```

```
\end{document}
```

- Detailed Explanation
  - `figure` environment: Used to create floating figures.
  - `[h]` option: Specifies the position of the image. `h` stands for "here," meaning as close as possible to the current position. Other options include:
    - `t`: Top
    - `b`: Bottom
    - `p`: Separate page for floats
    - `!`: Override certain restrictions (allows more flexibility in float placement)
  - : Centers the image.
  - `example-image}`: Inserts an image and sets its width to half the text width. `example-image` is the image filename.
  - `{This is an example image.}`: Adds a caption to the image.
  - `fig:example}`: Creates a label for the image to reference it in the document.
- Position Options for Floating Bodies

  Combine position options to better control the placement of the image. For example:



2 Floating Figure with Multiple Position Options

This example demonstrates a floating figure with multiple position options.

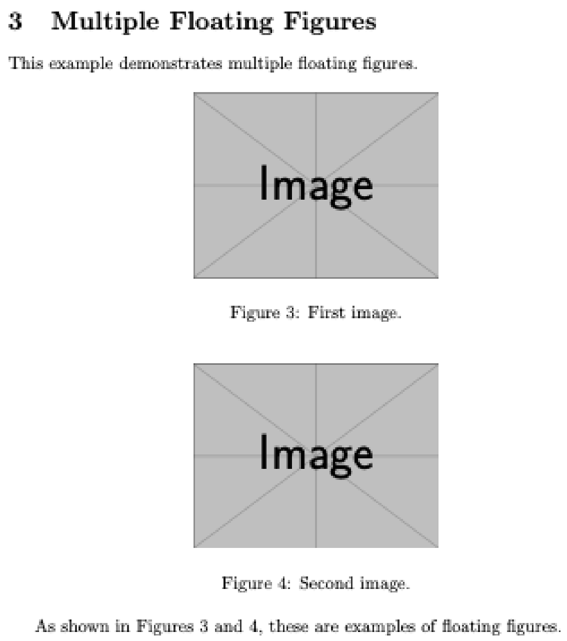
Figure 2: This is an example image with multiple position options.

  Code:

```
\begin{figure}[htbp]
    \centering
    \includegraphics[width=0.5\textwidth]{example-image}
    \caption{This is an example image with multiple position options.}
    \label{fig:example-multi}
\end{figure}
```

- Multiple Floating Figures

If you have multiple floating figures, you can place them together, like:



3 Multiple Floating Figures

This example demonstrates multiple floating figures.

Figure 3: First image.

Figure 4: Second image.

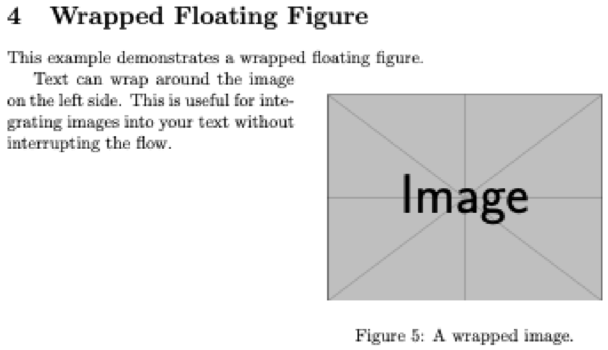
As shown in Figures 3 and 4, these are examples of floating figures.

Code:

```
\documentclass{article}
\usepackage{graphicx}

\begin{document}

\item{Introduction}

This example demonstrates multiple floating figures.

\begin{figure}[h]
    \centering
    \includegraphics[width=0.4\textwidth]{example-image-a}
    \caption{First image.}
    \label{fig:first}
\end{figure}

\begin{figure}[h]
    \centering
    \includegraphics[width=0.4\textwidth]{example-image-b}
    \caption{Second image.}
    \label{fig:second}
\end{figure}

As shown in Figures \ref{fig:first} and \ref{fig:second}, these are examples of float
```

```
\end{document}
```

- Wrapped Floating Figures



4 Wrapped Floating Figure

This example demonstrates a wrapped floating figure.

Text can wrap around the image on the left side. This is useful for integrating images into your text without interrupting the flow.

Figure 5: A wrapped image.

Code:

Use the `wrapfig` package to wrap floating figures around text:

```
\documentclass{article}
\usepackage{graphicx}
\usepackage{wrapfig}

\begin{document}

\item{Introduction}
This example demonstrates a wrapped floating figure.

\begin{wrapfigure}{r}{0.5\textwidth}
    \centering
    \includegraphics[width=0.45\textwidth]{example-image}
    \caption{A wrapped image.}
    \label{fig:wrapped}
\end{wrapfigure}

Text can wrap around the image on the left side. This is useful for integrating image

\end{document}
```

- Conclusion

  Using the above examples, you can implement floating figures in LaTeX and control their placement within your document. Floating figures make the document layout more flexible, avoiding layout issues while maintaining a professional appearance.

Editing Style Sheets:

- Introduction

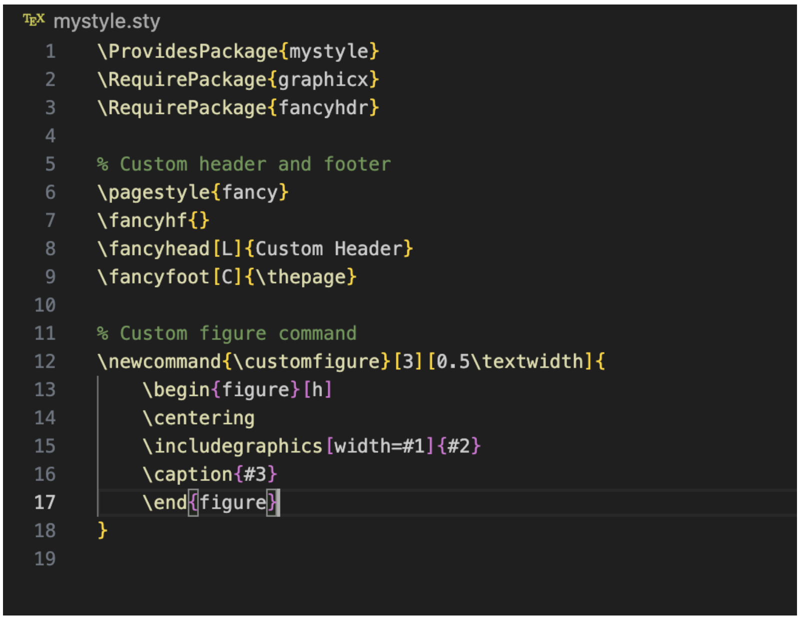
  Editing style sheets refers to the process of creating and using custom style files (typically .sty files) in LaTeX to control the appearance and formatting of a document. By defining style sheets, users can consistently manage the layout, fonts, headers and footers, figure styles, and more across multiple documents, thus enhancing the maintainability and consistency of the documents.

- Instruction

  The main task is to customize the header and footer, as well as custom graphic commands, in files with a. sty suffix. At the same time, it is required to be in the same directory as the. tex file, so that you can proceed with your own target layout.

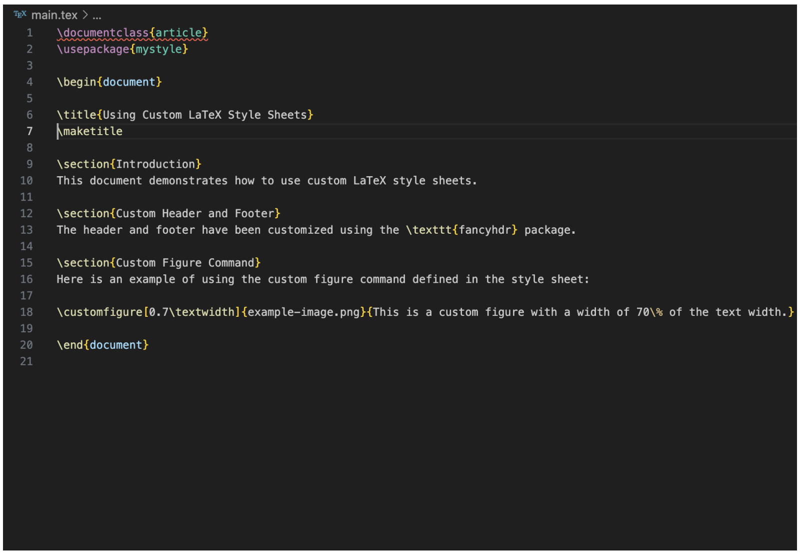
  The following is a code diagram of how to implement it

- .sty



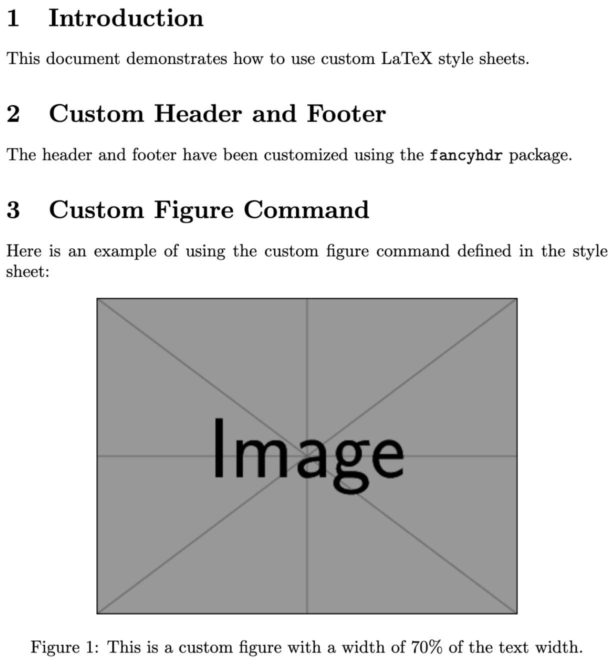
```
\ProvidesPackage{mystyle}
\RequirePackage{graphicx}
\RequirePackage{fancyhdr}

% Custom header and footer
\pagestyle{fancy}
\fancyhf{}
\fancyhead[L]{Custom Header}
\fancyfoot[C]{\thepage}

% Custom figure command
\newcommand{\customfigure}[3][0.5\textwidth]{
    \begin{figure}[h]
    \centering
    \includegraphics[width=#1]{#2}
    \caption{#3}
    \end{figure}
}

```

- .tex



```
\documentclass{article}
\usepackage{mystyle}

\begin{document}

\title{Using Custom LaTeX Style Sheets}
\maketitle

\section{Introduction}
This document demonstrates how to use custom LaTeX style sheets.

\section{Custom Header and Footer}
The header and footer have been customized using the \texttt{fancyhdr} package.

\section{Custom Figure Command}
Here is an example of using the custom figure command defined in the style sheet:

\customfigure[0.7\textwidth]{example-image.png}{This is a custom figure with a width of 70\% of the text width.}

\end{document}

```

- Output



1 Introduction

This document demonstrates how to use custom LaTeX style sheets.

2 Custom Header and Footer

The header and footer have been customized using the `fancyhdr` package.

3 Custom Figure Command

Here is an example of using the custom figure command defined in the style sheet:

Image

Figure 1: This is a custom figure with a width of 70% of the text width.

- Summary

  By editing style sheets, you can easily control the overall appearance and format of LaTeX documents, making their layout more consistent, easy to maintain, and reusable.

2.4. Tools and Skills for Cyber Security: Li, Senhao

Part A: Exploration of tech tools

Your text goes here

Part B: Analysis

Your text goes here

Technical Skills (LaTeX and Git)

Your text goes here

3. Submission contribution overview

For each submission, outline the approach taken to your teamwork, how you combined the various contributions, and whether there were any significant variations in the levels of involvement. (Target = ∼100-300 words).

3.1. Submission 1 contribution overview

In crafting our submission, our team adopted a proactive and collaborative approach from the outset. We began with organizing regular group meetings, which served as a platform for comprehensive planning and clear division of labor. This initial phase set a strong foundation for our teamwork, ensuring that each member understood their responsibilities and the collective goals of the project. Throughout the project, the spirit of mutual support was a cornerstone of our team dynamics. Members readily assisted each other in overcoming technical challenges, particularly in navigating the complexities of LaTeX and Git. This culture of assistance not only facilitated smoother workflow but also contributed to a learning environment where we could share knowledge and solutions. A significant portion of our collaborative effort was dedicated to a thorough exploration of the Skills Framework for the Information Age (SFIA). By delving deep into the framework, we engaged in meaningful discussions to select the most relevant skills that aligned with our project objectives and individual contributions. This rigorous analysis was crucial in ensuring that our report was not only comprehensive but also reflective of a strategic approach to skill selection and application. The combination of our kinds of contributions was seamless, thanks to the consistent levels of involvement from all team members. While there were variations in the specific tasks undertaken by each person, the overall commitment to the project remained uniformly high. This balance of diverse skills and unified focus culminated in the successful completion of our report, symbolizing the collective effort and dedication of our team.

3.2. Submission 2 contribution overview

After drawing on the advice provided by our tutor, we made more detailed modifications to address the previous steps that were not marked and the issues with referencing literature in latex. We are also ready to demonstrate the git operation in front of the tutor.

Callum Rann was on the Advanced section and thus moved forwards by himself.

All of this lead to not much oppurtunity for teamwork, however Callum will be receiving feedback for his section, and together will work to get good marks once all members are up to the Advanced task with the next submission.

3.3. Submission 3 contribution overview

In the third submission, only ZIXIMAI and QILONGZHONG attempted advanced skills, where QILONGZHONG corrected the issue of inappropriate image size in the foundation. The skills selected by ZIXIMAI for git and LaTeX are Forking and Special files, as well as Footnotes/target notes and creating new environments, respectively. The git and LaTeX skills selected by QILONGZHONG are Resetting and Tags and Floating figures and editing style sheets, respectively. Both of them fully followed the requirements of Strong, diligently studied and mastered the relevant skills, and completed the tasks of Skills in full.

Bibliography

[1] SFIA, "Computational science skills in sfia," 2021, see https://sfia-online.org/en/sfia-8/themes-for-sfia-8/computational-science-skills.

[2] Britannica, "Numerical analysis," 2017, see https://www.britannica.com/science/numerical-analysis.

[3] Scholarpedia, "Scholarpedia's numerical analysis," 2007, see http://www.scholarpedia.org/article/Numerical_analysis.

[4] National Institute of Biomedical Imaging and Bioengineering, "Computational modeling," 2020, see https://www.nibib.nih.gov/science-education/science-topics/computational-modeling.

[5] Wikipedia, "Turing pattern," 2023, see https://en.wikipedia.org/wiki/Turing_pattern.

[6] IBM, "What is hpc?" see https://www.ibm.com/topics/hpc.

[7] Netapp, "What is high-performance computing?" 2020, see https://www.netapp.com/data-storage/high-performance-computing/what-is-hpc.

[8] W. Van Der Aalst and W. van der Aalst, *Data science in action*. Springer, 2016. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-662-49851-4_1

[9] J. P. Birnholtz and M. J. Bietz, "Data at work: supporting sharing in science and engineering," in *Proceedings of the 2003 ACM International Conference on Supporting Group Work*, 2003, pp. 339–348. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/958160.958215?casa_token=q4y3N833-dwAAAAA:fDBvScLfhZfU7KIKjLmrRfY04aSIcHE9frY3vVdssnCcbyD9pwQ3lhaabMQObtoASWtjwwuktp8f

[10] M. L. Brodie, "On the development of data models," in *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*. Springer, 1984, pp. 19–47. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4612-5196-5_2

[11] J. A. Durlak, "Why program implementation is important," *Journal of Prevention & Intervention in the community*, vol. 17, no. 2, pp. 5–18, 1998. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1300/J005v17n02_02

[12] J. Bennedsen and M. E. Caspersen, "Revealing the programming process," in *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2005 (St. Louis, Missouri, USA, February 23-27, 2005)*. Association for Computing Machinery, 2005, pp. 186–190. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/1047344.1047413

[13] I. Harel and S. Papert, "Software design as a learning environment," *Interactive learning environments*, vol. 1, no. 1, pp. 1–32, 1990. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/1049482900010102

[14] C. Zhang and D. Budgen, "What do we know about the effectiveness of software design patterns?" *IEEE Transactions on Software Engineering*, vol. 38, no. 5, pp. 1213–1231, 2011. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5975176/

[15] R. J. Boland Jr, "The process and product of system design," *Management science*, vol. 24, no. 9, pp. 887–898, 1978. [Online]. Available: https://pubsonline.informs.org/doi/abs/10.1287/mnsc.24.9.887

[16] G. V. A. Vasantha, R. Roy, A. Lelah, and D. Brissaud, "A review of product–service systems design methodologies," *Journal of Engineering Design*, vol. 23, no. 9, pp. 635–659, 2012. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/09544828.2011.639712

[17] Security Expert Line Community, "Darkreading," 2006, see https://www.darkreading.com/cyber-risk.

[18] Bruce Schneier, "Schneier on security," 2003, see https://www.schneier.com.

[19] IT elites and white hackers, "Infosecurity magazine," 2003, see https://www.infosecurity-magazine.com/cybercriminals-and-convictions.

[20] Mirren McDade, "The top 10 container security tools," 2023, see https://expertinsights.com/insights/the-top-10-container-security-tools/.

[21] Adam Hill, "Python package manager comparison," 2023, see https://dev.to/adamghill/python-package-manager-comparison-1g98.

[22] Docker.docs, "Docker overview," 2024, see https://docs.docker.com/get-started/overview/.

[23] What is Docker? Uses and Limitations, "Jesús cantú," 2023, see https://medium.com/@jesus.cantu217/what-is-docker-uses-limitations-a10eb5ccec24.

[24] Poetry, "Poetry introoduction," see https://python-poetry.org/docs/.

[25] reddit, "Pain points of moving to poetry?" 2022, see https://www.reddit.com/r/Python/comments/y3vzho/pain_points_of_moving_to_poetry/.

[26] Wikibooks, "Latex/labels and cross-referencing," 2023, see https://en.wikibooks.org/wiki/LaTeX/Labels_and_Cross-referencing.

[27] T. Tapadar, "Importance of r in data science," 2023, see https://www.linkedin.com/pulse/importance-r-data-science-tuhin-tapadar#:~:text=R%20is%20a%20programming%20language,any%20data%20scientist%20to%20know.

[28] J. C. Luna, "Using functions in r tutorial," 2023, see https://www.datacamp.com/tutorial/functions-in-r-a-tutorial.

[29] C. Fontanella, "Sql server functions: 29 examples + definitions," 2023, see https://blog.hubspot.com/website/sql-server-functions#:~:text=SQL%20server%20functions%20are%20pre,%2C%20conditional%2C%20and%20aggregate%20functions.

[30] N. Tiwari, "Sql for data science: A beginner guide!" 2024, see https://www.analyticsvidhya.com/blog/2021/06/sql-for-data-science-a-beginners-guide/#:~:text=SQL%20is%20extremely%20important%20for,SQL%20for%20processing%20structured%20data.

[31] S. Thakur, "Advantages and disadvantages of sql simplified (with examples)," 2023, see https://unstop.com/blog/advantages-and-disadvantages-of-sql.

[32] R. Wentk, *Xcode 5 developer reference.* John Wiley & Sons, 2014.

[33] B. Ohland and J. Varma, *Xcode 7 Essentials.* Packt Publishing, 2016.

[34] Z. Lin, G. Li, J. Zhang, Y. Deng, X. Zeng, Y. Zhang, and Y. Wan, "Xcode: Towards cross-language code representation with large-scale pre-training," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 3, pp. 1–44, 2022.

[35] Q. Hu, L. Ma, and J. Zhao, "Deepgraph: A pycharm tool for visualizing and understanding deep learning models," in *2018 25th Asia-Pacific Software Engineering Conference (APSEC).* IEEE, 2018, pp. 628–632.

[36] B. M. Van Horn II and Q. Nguyen, *Hands-On Application Development with PyCharm: Build applications like a pro with the ultimate python development tool.* Packt Publishing Ltd, 2023.

[37] R. C. B. PARCON, J. E. L. SETIAS, M. T. M. HOLIPAS, and E. ED, "Comparison of transaction reduction and dynamic itemset counting implementations of the apriori algorithm in python 3.8."

[38] J. Loeliger and M. McCullough, *Version Control with Git: Powerful tools and techniques for collaborative software development.* " O'Reilly Media, Inc.", 2012.

[39] git official, "Referencing and citation styles: IEEE," 2022, see https://git-scm.com/book/en/v2/Git-Basics-Tagging.