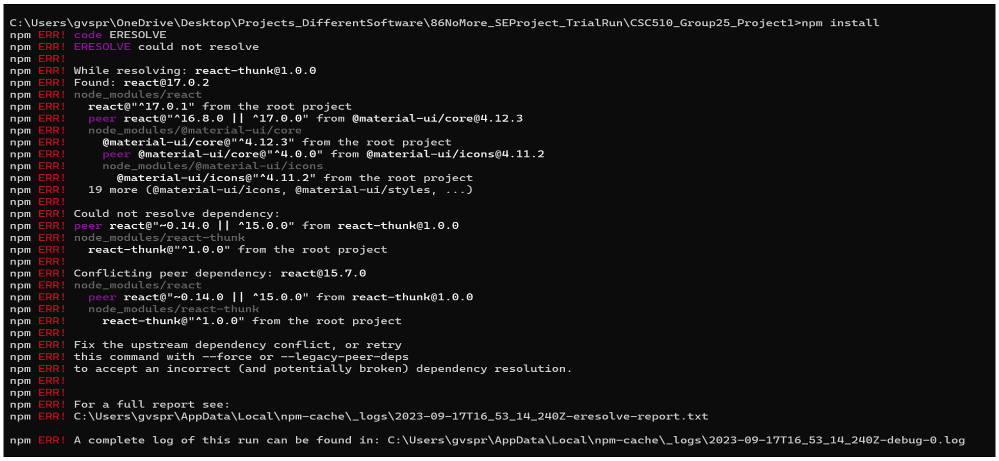
Difficulties faced in the process of selecting and running projects and brainstorming on how that pain could have been avoided.

Selecting and running the prior projects worked on by our seniors, after a significant time gap was in fact challenging and interesting. There were several difficulties faced by our team during the project selection and execution process and we will try to incorporate a few practices in our project2 to avoid the pain.

For most of the projects, even after executing all the steps as documented we still got the issues due to version mismatches, incompatibility, deprecated software functions etc. Some of the major challenges we have faced during the execution of the projects in the order of complexity are:

1) **Dependency Issues:** The software code written by our seniors was developed around a year prior. So in this time gap, software dependencies and environment configuration has changed drastically. There were newer versions of some libraries which were not compatible with older versions of other libraries that we are trying to install. We have faced these issues while executing 3 projects out of the 5 projects assigned to us. These outdated dependencies resulted in errors during the installation process. One such example of this kind of issue is illustrated in Fig1.



```
C:\Users\gvspr\OneDrive\Desktop\Projects_DifferentSoftware\86NoMore_SEProject_TrialRun\CSC510_Group25_Project1>npm install
npm ERR! code ERESOLVE
npm ERR! ERESOLVE could not resolve
npm ERR!
npm ERR! While resolving: react-thunk@1.0.0
npm ERR! Found: react@17.0.2
npm ERR! node_modules/react
npm ERR!   react@"^17.0.1" from the root project
npm ERR!   peer react@"^16.8.0 || ^17.0.0" from @material-ui/core@4.12.3
npm ERR!   node_modules/@material-ui/core
npm ERR!     @material-ui/core@"^4.12.3" from the root project
npm ERR!     peer @material-ui/core@"^4.0.0" from @material-ui/icons@4.11.2
npm ERR!     node_modules/@material-ui/icons
npm ERR!       @material-ui/icons@"^4.11.2" from the root project
npm ERR!   19 more (@material-ui/icons, @material-ui/styles, ...)
npm ERR!
npm ERR! Could not resolve dependency:
npm ERR! peer react@"~0.14.0 || ^15.0.0" from react-thunk@1.0.0
npm ERR! node_modules/react-thunk
npm ERR!   react-thunk@"^1.0.0" from the root project
npm ERR!
npm ERR! Conflicting peer dependency: react@15.7.0
npm ERR! node_modules/react
npm ERR!   peer react@"~0.14.0 || ^15.0.0" from react-thunk@1.0.0
npm ERR!   node_modules/react-thunk
npm ERR!     react-thunk@"^1.0.0" from the root project
npm ERR!
npm ERR! Fix the upstream dependency conflict, or retry
npm ERR! this command with --force or --legacy-peer-deps
npm ERR! to accept an incorrect (and potentially broken) dependency resolution.
npm ERR!
npm ERR!
npm ERR! For a full report see:
npm ERR! C:\Users\gvspr\AppData\Local\npm-cache\_logs\2023-09-17T16_53_14_240Z-eresolve-report.txt

npm ERR! A complete log of this run can be found in: C:\Users\gvspr\AppData\Local\npm-cache\_logs\2023-09-17T16_53_14_240Z-debug-0.log
```

Fig1: Running npm install in the frontend package of 86NoMore throws errors due to package conflicts

2) **Deprecated Technologies:** It was quite challenging for us to find the compatible versions or replacements for the deprecated technologies. Replacing the deprecated libraries required us to understand where and how it was used, learn how to use the newer version of the library and implement them. This was a time consuming and tedious process.

3) **Compatibility with different Operating Systems:** There was one less commonly faced issue which we have encountered running one of our projects. The documentation was

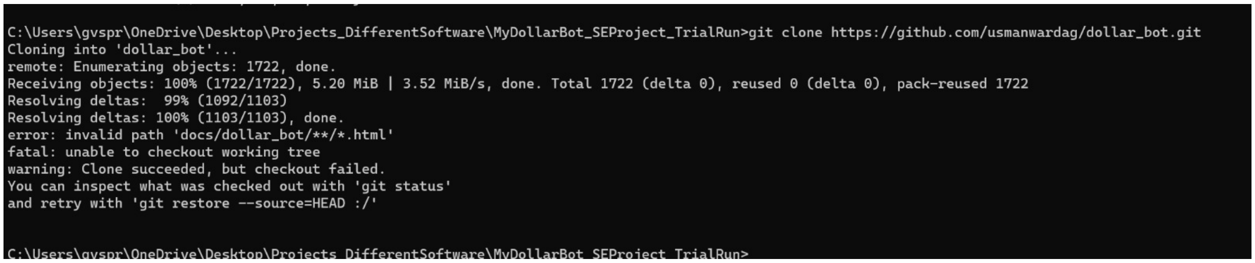
clear and detailed but the steps for executing the project were only relevant to the Linux/Unix based Operating System. The commands such as *chmod* and *shell script files* were specific to the Linux/ Unix OS and no documentation for executing the project in the Windows based environment was available. So it required us to learn the alternative commands for the Windows environment and modify the shell scripts so that it runs on the Windows platform.

4) **Obsolete and Incomplete Documentation:** In some cases we found that the project documentation has become outdated or incomplete
E.g. 1 Obsolete Documentation - incorrect or outdated git clone command - Result: we cloned wrong repository or outdated repository
E.g. 2 Incomplete Documentation - Some of the python projects required C++ build tools support to run but it was not mentioned in the documentation

The steps required to execute the project were not detailed and it just provided a high-level overview of the process in some cases. This made it difficult to understand the setup process and run the project effectively. The lack of up-to-date and complete documentation was one of the significant problems faced by our team.

The availability of a clear, detailed, complete and interactive documentation would have alleviated most of the problems encountered by us while running the project.

And finally some of the errors and issues, which we had no idea of :),
Example Context: We have used the clone path exactly as specified in documentation and got this issue but later managed to resolve them by trying other methods of downloading.



```
C:\Users\gvspr\OneDrive\Desktop\Projects_DifferentSoftware\MyDollarBot_SEProject_TrialRun>git clone https://github.com/usmanwardag/dollar_bot.git
Cloning into 'dollar_bot'...
remote: Enumerating objects: 1722, done.
Receiving objects: 100% (1722/1722), 5.20 MiB | 3.52 MiB/s, done. Total 1722 (delta 0), reused 0 (delta 0), pack-reused 1722
Resolving deltas:  99% (1092/1103)
Resolving deltas: 100% (1103/1103), done.
error: invalid path 'docs/dollar_bot/**/*.html'
fatal: unable to checkout working tree
warning: Clone succeeded, but checkout failed.
You can inspect what was checked out with 'git status'
and retry with 'git restore --source=HEAD :/'


C:\Users\gvspr\OneDrive\Desktop\Projects_DifferentSoftware\MyDollarBot_SEProject_TrialRun>
```

Fig2 : Error while Cloning a repository to the local system using the Git Clone command. The above issue might be the result of obsolete documentation.

Some general practices we are looking to incorporate in Project2:

These are some general practices which we will be incorporating for our upcoming projects in order to improve our code quality as well as to tackle the aforementioned issues.

1) Much detailed and elaborate documentation by including common misconceptions and screenshots of some issues faced by us in order to address the newbie developers.
2) Maintain a clear, detailed and consistent README.md document i.e. Documentation.
3) Including the environment files such as requirements.txt, environment.yml along with the information about important practices to follow during the installation process clearly in the documentation.
4) Including Well Defined and Detailed comments throughout the project's source code to help others understand the purpose of the functions and methods used in the project.
5) One Image can describe a thousand words - So we will try to make the documentation more interactive with the usage of images/ screenshots. Usage of Hyperlinks to refer to external links wherever required.
6) Using libraries which have a strong community support.