Report for assignment 4

Project

We have chosen to work with Textualize/rich.

Rich is a Python library for rich text and beautiful formatting in the terminal. It provides an easy-to-use API to add color and style to the terminal output, and can render tables, progress bars, markdown, syntax highlighted source code, tracebacks, and more.

```
https://github.com/Ceevtj/rich
```

Ecosystem

> *(P+) In the context of Jonas Öberg's lecture last week, where do you put the project that you have chosen in an ecosystem of open-source and closed-source software? Is your project (as it is now) something that has replaced or can replace similar proprietary software? Why (not)?*

Within the ecosystem of open-source and closed-source software, the project is a flexible tool to enhance textual interfaces. As there already exist similar tools and libraries, e.g. https://github.com/gaoliang/tmlpy, the project stands out by its simplicity, flexibility, and comprehensive feature set. Thus, the integration with Python makes it more appealing for certain developers.

Currently, the project has a comprehensive range of features and it seems like a reasonable choice to use Rich over proprietary solutions to perform terminal formatting and styling. Of course, in some cases proprietary tools has to be used because of certain specialized features or requirements.

However, it seems that the project is still growing within the software ecosystem. With the ongoing developments and its current community feedback it could evolve to a leading solution for terminal formatting.

Onboarding experience

We have chosen to work with a new project. As compared to our last project, JavaParser, the onboarding experience was significantly better due to the clear and detailed contributing guidelines provided. Thanks to these comprehensive guidelines, every member of our team was able to get the project up and running without any hitches.

Effort spent

> *For each team member, how much time was spent in*
>
> *1. plenary discussions/meetings;*
> *2. discussions within parts of the group;*
> *3. reading documentation;*
> *4. configuration and setup;*
> *5. analyzing code/output;*
> *6. writing documentation;*
> *7. writing code;*
> *8. running code?*
>
> *For setting up tools and libraries (step 4), enumerate all dependencies you took care of and where you spent your time, if that time exceeds 30 minutes.*

Time spent	**Angelica**	**Tianxing**	**Tobias**	**John**	**Celina**
1	2 h + 3 h	2 h + 3 h	2 h + 3h	2 h + 3 h	2 h + 3 h
2					
3	5 h	2h	2h+1h+1h	3h	6h
4	1 h	1h	1h+1h (bug)	1h	1h
5	2 h	2h	3h+1h	1h	3h
6	6 h	1h	1h		4h
7	1 h	2h+3h+5h	5h+3h	2h+3h+3h+3h	1h
8					
Total	20h	21h	20h	21h	20h

Overview of issue(s) and work done.

> *Summary in one or two sentences*
> *Scope (functionality and code affected).*

Issue #3118 Make classes pretty printable via __rich_repr__

https://github.com/Textualize/rich/issues/3118

The issue is a request to make classes pretty printable via `__rich_class_repr__` . The current architecture already supports `__rich_repr__` for instances but not the class itself, so adding the support will make it function more systematically.

Issue #3208 Global settings/theming config

https://github.com/Textualize/rich/issues/3208

The issue is a request to implement global per-user config that can control settings like theming. Current architecture has a default setting for themes and terminal themes. Importing a global config will enable the user to customize its experience without modifying the code itself.

Requirements for the new feature or requirements affected by functionality being refactored

> *Identify requirements related to the issue(s).*
> *Describe the requirements with an ID, title, and description.*
> *(P+) trace tests to requirements.*

Issue #3118 Make classes pretty printable via __rich_repr__

ID	Title	Description
0	Allow classes to use `__rich_class_repr__`	Classes has method `__rich_class_repr__` should be pretty printed.
1	Pretty print can be derived via `@deriving_rich`	Generate default methods to pretty print an object
2	Pretty print for class can be derived via `@deriving_rich_class`	Generate default methods to pretty print a class

Issue #3208 Add global config/theming

ID	Title	Description
0	Make config loadable	Add a way to load theme and/or style from the default config

1	Add TerminalTheme()	Series of Tuples for terminal colors
2	Add Theme()	Styles for markdown, trees, progressbars, and more

Relevant test cases

Issue #3118 Make classes pretty printable via __rich_repr__

ID	Name	Description	Requirements
0	`test_derive`	Test for deriving rich on classes for objects	1
1	`test_derive_class`	Test for deriving rich for classes itself	0, 2

Issue #3208 Add global config/theming

ID	Name	Description	Requirements
0	`test_global_config`	Tests for global configs	0
1	`test_theme.py`	All tests related to default theme settings and initialization	2
2	`test_terminal_theme.py`	All test related to terminal theme initialization	1

Code changes

Patch

> *(copy your changes or the add git command to show them)*
> *git diff ...*
> *(P+) the patch is clean.*
> *(P+) considered for acceptance (passes all automated checks).*

```
git diff 26152e9 231eaed
```

Test results

> *Overall results with link to a copy or excerpt of the logs (before/after refactoring).*

```
make test
```

testlog-before

testlog-after

```
make typecheck
```

typechecklog-before

typechecklog-after

```
make format-check
```

formatchecklog-before

formatchecklog-after

Testlog after issue resolution

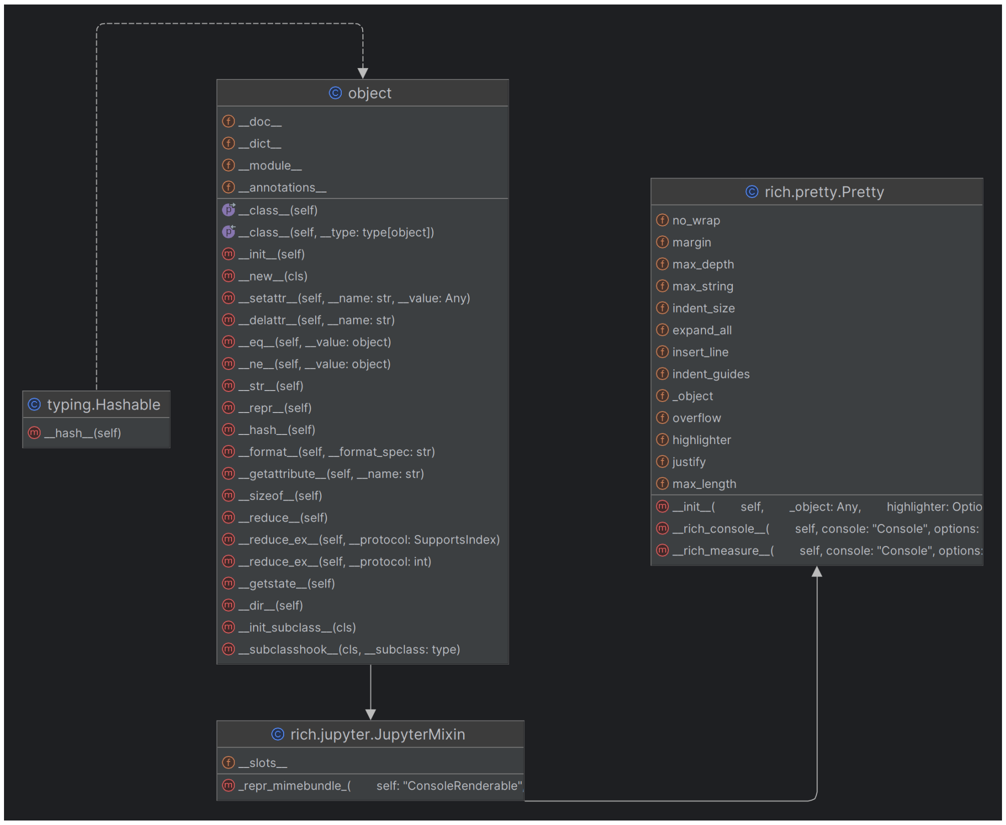
testlog-after.txt

UML class diagram and its description

> *Create a before and after UML diagram that shows the key features of the issue(s)*

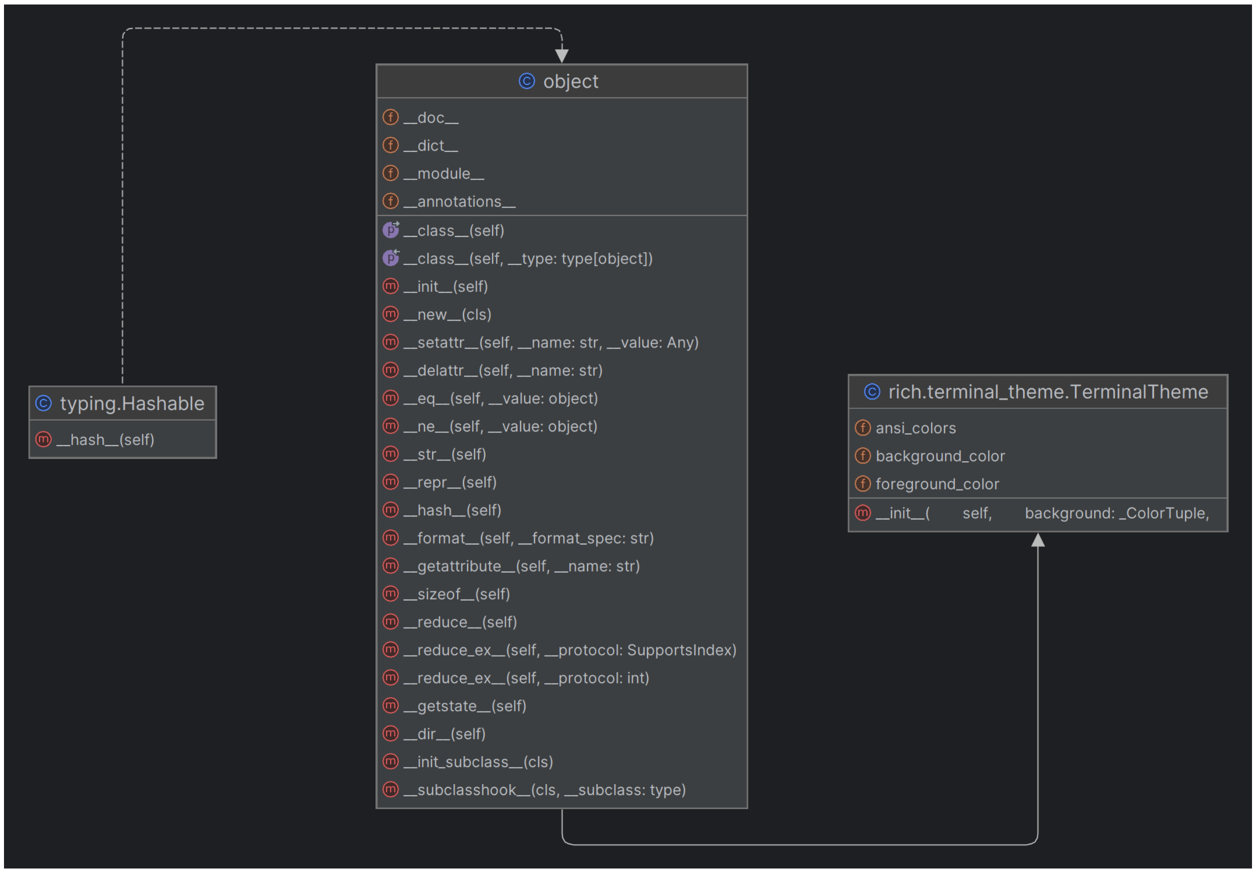
Issue #3118 Make classes pretty printable via __rich_repr__

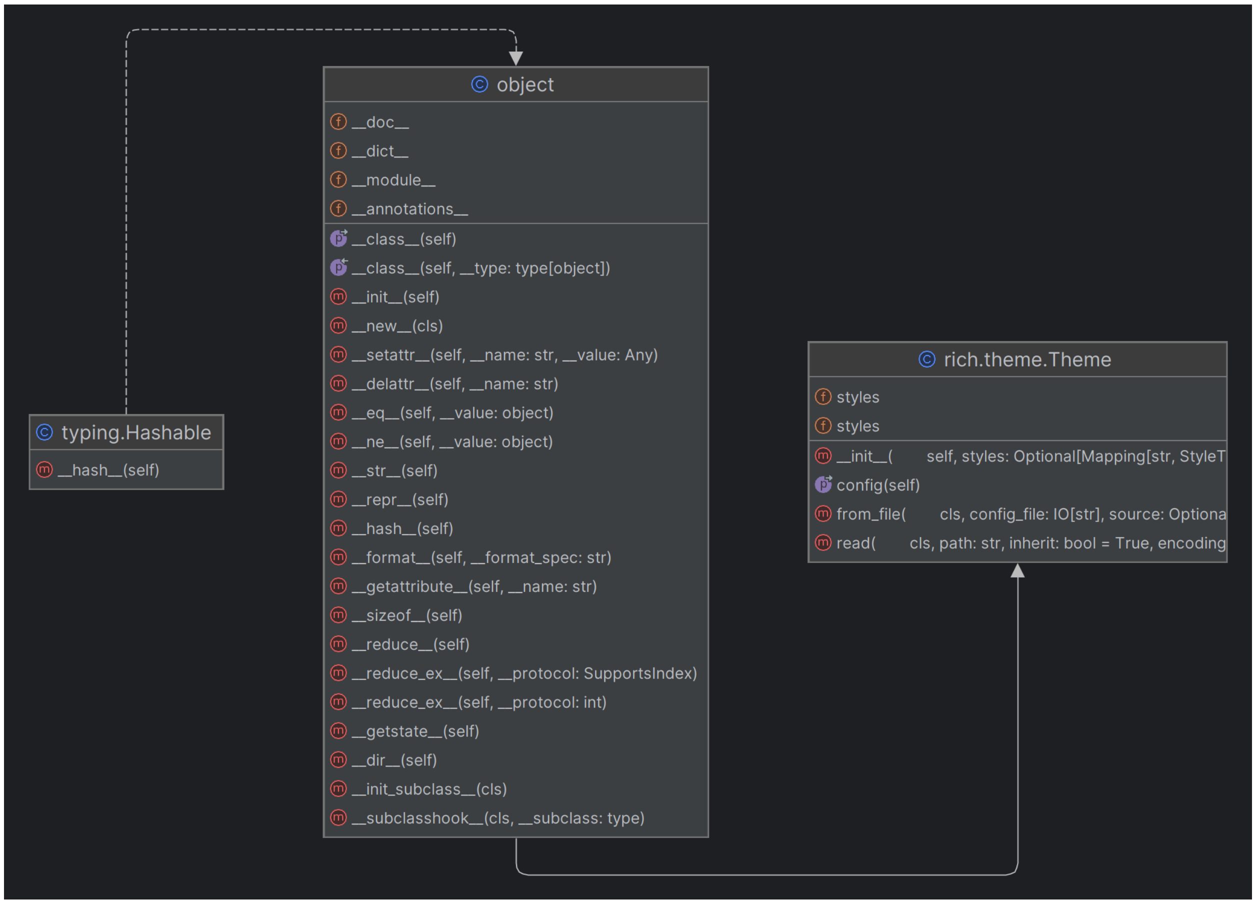
The UML class diagram below shows the key features that concerns issue #3118.


object
__doc__
__dict__
__module__
__annotations__
__class__(self)
__class__(self, __type: type[object])
__init__(self)
__new__(cls)
__setattr__(self, __name: str, __value: Any)
__delattr__(self, __name: str)
__eq__(self, __value: object)
__ne__(self, __value: object)
__str__(self)
__repr__(self)
__hash__(self)
__format__(self, __format_spec: str)
__getattribute__(self, __name: str)
__sizeof__(self)
__reduce__(self)
__reduce_ex__(self, __protocol: SupportsIndex)
__reduce_ex__(self, __protocol: int)
__getstate__(self)
__dir__(self)
__init_subclass__(cls)
__subclasshook__(cls, __subclass: type)
typing.Hashable
__hash__(self)
rich.pretty.Pretty
no_wrap
margin
max_depth
max_string
indent_size
expand_all
insert_line
indent_guides
_object
overflow
highlighter
justify
max_length
__init__(self, _object: Any, highlighter: Optio
__rich_console__(self, console: "Console", options:
__rich_measure__(self, console: "Console", options:
rich.jupyter.JupyterMixin
__slots__
_repr_mimebundle_(self: "ConsoleRenderable"

Issue #3208 Add global config/theming

The UML diagrams below shows the key features that concerns issue #3208.


object
f __doc__
f __dict__
f __module__
f __annotations__
p __class__(self)
p __class__(self, __type: type[object])
m __init__(self)
m __new__(cls)
m __setattr__(self, __name: str, __value: Any)
m __delattr__(self, __name: str)
m __eq__(self, __value: object)
m __ne__(self, __value: object)
m __str__(self)
m __repr__(self)
m __hash__(self)
m __format__(self, __format_spec: str)
m __getattribute__(self, __name: str)
m __sizeof__(self)
m __reduce__(self)
m __reduce_ex__(self, __protocol: SupportsIndex)
m __reduce_ex__(self, __protocol: int)
m __getstate__(self)
m __dir__(self)
m __init_subclass__(cls)
m __subclasshook__(cls, __subclass: type)
typing.Hashable
m __hash__(self)
rich.terminal_theme.TerminalTheme
f ansi_colors
f background_color
f foreground_color
m __init__(self,   background: _ColorTuple,


object
__doc__
__dict__
__module__
__annotations__
__class__(self)
__class__(self, __type: type[object])
__init__(self)
__new__(cls)
__setattr__(self, __name: str, __value: Any)
__delattr__(self, __name: str)
__eq__(self, __value: object)
__ne__(self, __value: object)
__str__(self)
__repr__(self)
__hash__(self)
__format__(self, __format_spec: str)
__getattribute__(self, __name: str)
__sizeof__(self)
__reduce__(self)
__reduce_ex__(self, __protocol: SupportsIndex)
__reduce_ex__(self, __protocol: int)
__getstate__(self)
__dir__(self)
__init_subclass__(cls)
__subclasshook__(cls, __subclass: type)
typing.Hashable
__hash__(self)
rich.theme.Theme
styles
styles
__init__(self, styles: Optional[Mapping[str, StyleT
config(self)
from_file(cls, config_file: IO[str], source: Optiona
read(cls, path: str, inherit: bool = True, encoding

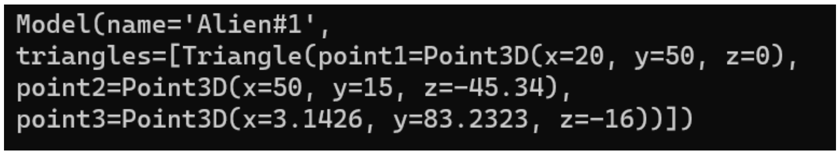
Key changes/classes affected

> *(P+) Architectural overview.*
> *(P+) relation to design pattern(s).*

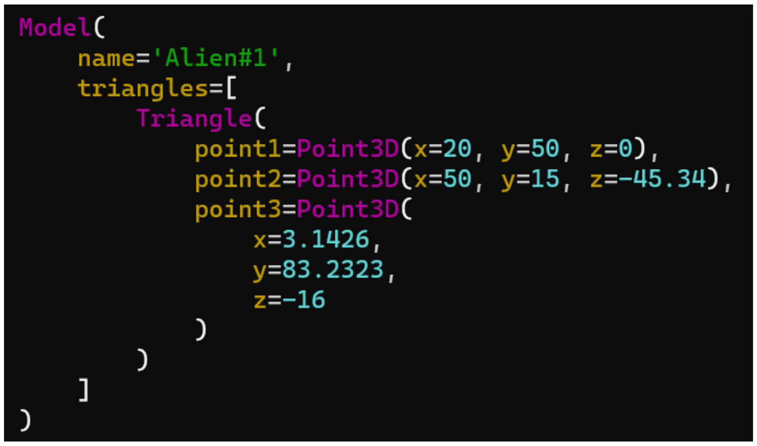
Issue #3118 Make classes pretty printable via __rich_repr__

The UML diagram presented above for issue #3118 consists of 4 different classes. The main class that is affected by the issue is the class `Pretty`.

The `Pretty` class is a renderable that pretty prints an object. The project provided an example of the functionality in `examples/attrs.py` where a initiated class called `Model` would be printed as


```
Model(name='Alien#1',
triangles=[Triangle(point1=Point3D(x=20, y=50, z=0),
point2=Point3D(x=50, y=15, z=-45.34),
point3=Point3D(x=3.1426, y=83.2323, z=-16))])
```

with the already existing python function `repr`. But by using the `Pretty` class you would instead receive a prettier output according to this



```
Model(
    name='Alien#1',
    triangles=[
        Triangle(
            point1=Point3D(x=20, y=50, z=0),
            point2=Point3D(x=50, y=15, z=-45.34),
            point3=Point3D(
                x=3.1426,
                y=83.2323,
                z=-16
            )
        )
    ]
)
```

In order to do so, the `Pretty` class mainly needs one type of argument, such as

- `_object(Any)` An object to pretty print (e.g. the `Model` class)

The rest of the arguments are optional which is used to specify the output with highlighting, margin, max width, and so on.

In order to print a `Pretty` object to the console, one would have to use the `print` method from an `rich.console.Console` object. By taking a object as an argument, the `print` method would call the `is_expandable` function and the objects defined `__rich_console__` method.

The `__rich_console__` method is defined within any string, Segment or object that could be renderable, since the method should provide a renderable result. In relation to design patterns, one could analyze that the structure is similar to the Visitor pattern, since the renderable algorithm is separated from each object that is provided as an argument to the `print` method.

Thus, the `__rich_console__` method is defined within the `Pretty` class. One of the procedures that the method will perform is to prettify the output by expanding new lines to fit the output within a given width, which is done by calling the `pretty_repr` method that returns a possibly multi-line representation of the given `_object` as a string.

In order to do so, the `pretty_repr` method calls the `traverse` method, which traverses the given `_object` and generates a tree. The `traverse` method will similarly use the Visitor design pattern by calling the given objects defined `__rich_repr__` method if the object is not a class, which returns renderable data of the object.

As the issue stated, we would want classes to be able to be printable via `__rich_repr__`. In order to do so, we used the Visitor design pattern to call a new method called `__rich_class_repr__` in `traverse` that classes could define to return renderable data.

Additionally, we added `@deriving_rich_class` and `@deriving_rich` that could be used on a class to generate a default `__rich_class_repr__` method and a default `__rich_repr__` method.

Issue #3208 Add global config/theming

The UML diagrams presented above for issue #3208 consists mainly of two different structures regarding the classes `Theme` and `TerminalTheme` .

The `Theme` class is basically a container for style information that is used by an `rich.console.Console` object. The arguments used for the `Theme` class is

- `styles` a dictionary for mapping of style names on to styles, default is `None`
- `inherit` a bool for inheritance of default styles, default is `true`

The constructor of the class basically assigns the member `self.styles` dictionary to a predefined default style if `inherit` is set. Thereafter, if the argument dictionary `styles` is set, the member `self.styles` is updated with new styles.

In order to change the theme with a global configuration, we added the `load_global_config` function that loads a global config file if there is one. Thus, we modified the constructor of the `Theme` class by using the `load_global_config` function.

If the `load_global_config` function found a configuration file, it would return a dictionary containing the data from the file. Thus, we could thereafter just update the predefined default style with the dictionary we just received.

Continuously, the `TerminalTheme` class is a color theme used when exporting console content. The arguments used for the class is

- `background` a triplet container representing the backgrounds RGB colors
- `foreground` a triplet container representing the foregrounds RGB colors
- `normal` a list of 8 triplet containers representing normal intensity RGB colors
- `bright` a list of 8 triplet containers representing bright RGB colors, default is `None`

The default style was originally assigned as a `TerminalTheme` object with hard coded arguments. To enhance flexibility, the function `load_default_terminal_theme` was added which utilizes the `load_global_config` function that assigns the default style other values if a configuration file was found.

The modifications is known as the refactoring method Extract Method since we extract the hard coded values into separate methods where the data could be modified using a configuration file, which improves modularity and readability.

Overall experience

> *What are your main take-aways from this project? What did you learn? How did you grow as a team, using the Essence standard to evaluate yourself?*
> *(P+) How would you put your work in context with best software*

> *engineering practice?*
> *(P+) Is there something special you want to mention here?*

The work we have done in this project has given us many opportunities to learn and experience how to work as members of a software engineering project, which is experience that will make us more proficient adding to open-source projects in the future. With this assignment we have applied the things we've studied with version control, testing, issue tracking and workflow in a more real world application. Working with a real project with many thousands lines of codes and realizing the complexity of efficiently identifying the relevant pieces of codes and requirements to resolve the issues .The format of the assignment lends itself well to following the SEMAT kernel framework even though our project is of smaller scope its still following the same fundamentals. Since there was a time constraint and with the specific project we chose there wasn't as much back and forth with the stakeholders, in this case the project team, as was outlined by the framework.

We've had to search for an open source project and familiarize ourselves with it, reading about the needs of the "customer" as described in the issues and identify the opportunities for the solutions. The project Rich that we chose had plenty of open issues that we could choose from that stated the need for additional software-based solutions.

We've gotten ample opportunity to experience the social aspect of software engineering where every step had to be discussed and agreed upon which we learnt from. Analyzing the problem at hand and formulating the requirements and what to do to reach them as well as tests needed to ensure the issues would be solved and the system would stay running. It enforced method agility, having to modify and change methods and practices to better adjust for things we've learned and if the needs were to change as well as what the next steps were to be to reach the wanted results.

One of the limiting factors of our work was that the project did not seem to assign issues to devs, while also being very active. Leading to a real possibility of working concurrently on the same issue as other developers, doing work that has already been done, which would be a waste of time and resources.

Since the issues we chose were not all-encompassing we've mostly examined only the relevant part of the code for solving the issue and to not cause any conflicts with existing code. As such we did not have a lot of context for customer and user needs. Mostly relied on the information available in the issues and documentation, which could be a bit lacking at times, and wrote the requirements based on that.

A drawback we encountered were the issues available to us did not require as much coding as most issues revolved around adding features to already well working code and minor bugs. That is to say it wasn't as time consuming as we would have wanted to reach the hours needed for the scope of the assignment. The first issue did not take long so we had to pick an additional one.

Team

Seeded

- [x] ~~The team mission has been defined in terms of the opportunities and outcomes.~~
- [x] ~~Constraints on the team's operation are known.~~
- [ ] Mechanisms to grow the team are in place.
- [x] ~~The composition of the team is defined.~~
- [x] ~~Any constraints on where and how the work is carried out are defined.~~
- [x] ~~The team's responsibilities are outlined.~~
- [x] ~~The level of team commitment is clear.~~
- [x] ~~Required competencies are identified.~~
- [x] ~~The team size is determined.~~
- [x] ~~Governance rules are defined.~~
- [x] ~~Leadership model is determined.~~

Formed

- [x] ~~Individual responsibilities are understood.~~
- [x] ~~Enough team members have been recruited to enable the work to progress.~~
- [x] ~~Every team member understands how the team is organized and what their individual role is.~~
- [x] ~~All team members understand how to perform their work.~~
- [x] ~~The team members have met (perhaps virtually) and are beginning to get to know each other.~~
- [x] ~~The team members understand their responsibilities and how they align with their competencies.~~
- [x] ~~Team members are accepting work.~~
- [x] ~~Any external collaborators (organizations, teams and individuals) are identified.~~
- [x] ~~Team communication mechanisms have been defined.~~
- [x] ~~Each team member commits to working on the team as defined.~~

Collaborating

- [x] ~~The team is working as one cohesive unit.~~

- [x] ~~Communication within the team is open and honest.~~
- [x] ~~The team is focused on achieving the team mission.~~
- [x] ~~The team members know and trust each other.~~

Performing

- [x] ~~The team consistently meets its commitments.~~
- [x] ~~The team continuously adapts to the changing context.~~
- [x] ~~The team identifies and addresses problems without outside help.~~
- [x] ~~Effective progress is being achieved with minimal avoidable backtracking and reworking.~~
- [x] ~~Wasted work and the potential for wasted work are continuously identified and eliminated.~~

Adjourned

- [x] ~~The team responsibilities have been handed over or fulfilled.~~
- [x] ~~The team members are available for assignment to other teams.~~
- [x] ~~No further effort is being put in by the team to complete the mission.~~