**Functional Requirements**

**Complete**

At this stage, functionality has been implemented wholly for the completion of a standard minesweeper game in an isolated environment. Previous listings of functional requirements detailed the following key aspects of the game:

* Implementation of Title Page Complete with Navigation
* Implementation of Gameplay Grid
* Implementation of Gameplay Cells with Mechanics
* Implementation of Gameplay Flags
* Implementation of End Game

Each gameplay elements have been completed in regards to the standard minesweeper game. Aside from this, no-functional requirements have been implemented including:

* GUI
* Performance – No lag
* No install

**Ongoing**

Previously, functional requirements pertained to a single instance of the minesweeper game – being the standard minesweeper. An overhaul of functional requirements will be necessary as previously, they only described a single game mode. New functional requirements will need to be created to accommodate for the additional game modes being: hex and colour. In terms of previous functional requirements, the only one which is still ongoing is the Implementation of a Title Page.

In terms of prioritization, the following list describes the important tasks to complete and the order in which they will be prioritized.

* **Title Page Complete with Navigation to all Screens** – The reason this is the ease of implementation vs the structure it will provide to the subsequent requirements. It will be important to place the already completed Minecraft game within the context of the entire project. Furthermore, it will be useful to develop the subsequent game types within the context of the entire game as it will give a good feel for the flow of the UI as a whole while developing.
* **Prototyping of Hex Minesweeper** **–** Hex Minesweeper will be the next challenge and the same functional requirements used within the standard minesweeper game can be translated as a subset for this version. Very few changes should be necessary, however, the GUI will need to be rethemed.
* **Prototyping of Colour Minesweeper –** Once Hex has been implemented, a new game type will need to be developed. At present, The concept for this game is confusing so the details of the precise functional requirements involved will be necessary. This will be prioritised, mapping the functional requirements for the colour game and developing a prototype.
* **Cross-Platform Capabilities –** This final requirement will be important to address once the initial Minesweeper project has been completed. At present, the goal is to provide functionality across web browser and for Microsoft, Windows and Linux devises running web browsers.

**Use Cases**

**Software Architecture**

The current IDE being utilised to implement minesweeper is brackets. This is free software and relatively simple to use. The language being used is JavaScript. Though there are no class models implementable in JavaScript, there are ways in which one may mimic them. Finally, the p5.js library is being used to display the front end GUI for the project

For a small project such as minesweeper, this software architecture is surprisingly efficient. There is unlimited room for GUI customization with the p5 library and JavaScript has more than enough power to deliver the required functionality. “Classes” may be written into modules and methods may be applied via prototyping, these modules may be called from the main script to display components to the user with inbuilt logic.

In terms of feasibility, this architecture is easy to use, costs no money, is flexible and is able to get the job done. All things considered, this architecture may be basic but it is more than feasible for this particular project. This architecture will remain throughout project completion.

There will be modules to govern the separate functionalities of each type of cell based on the game type. For instance, a hex tile acts slightly different to a colour tile. These three modules will be built upon a collection of modules which are common to the functionality of all types of cell. Ie, the ability for a cell to be revealed.

**Persistent Data Management, Security, UI**

Persistent Data Management is not utilized in this project. Data which might be useful to storage would potentially be high scores or something of the like. In this case, a JSON file could be included along with the implementation. This would guarantee persistent storage across sessions; however, the data would not be necessarily secure. The level of sophistication in this case is exceptionally minimal but it is a thought.

Access control and security features have not been included. The code for this implementation will in fact, be free for anyone to browse, alter and study if they so choose. Implementing security and access control for such a basic application is excessive as the application itself lacks sophistication.

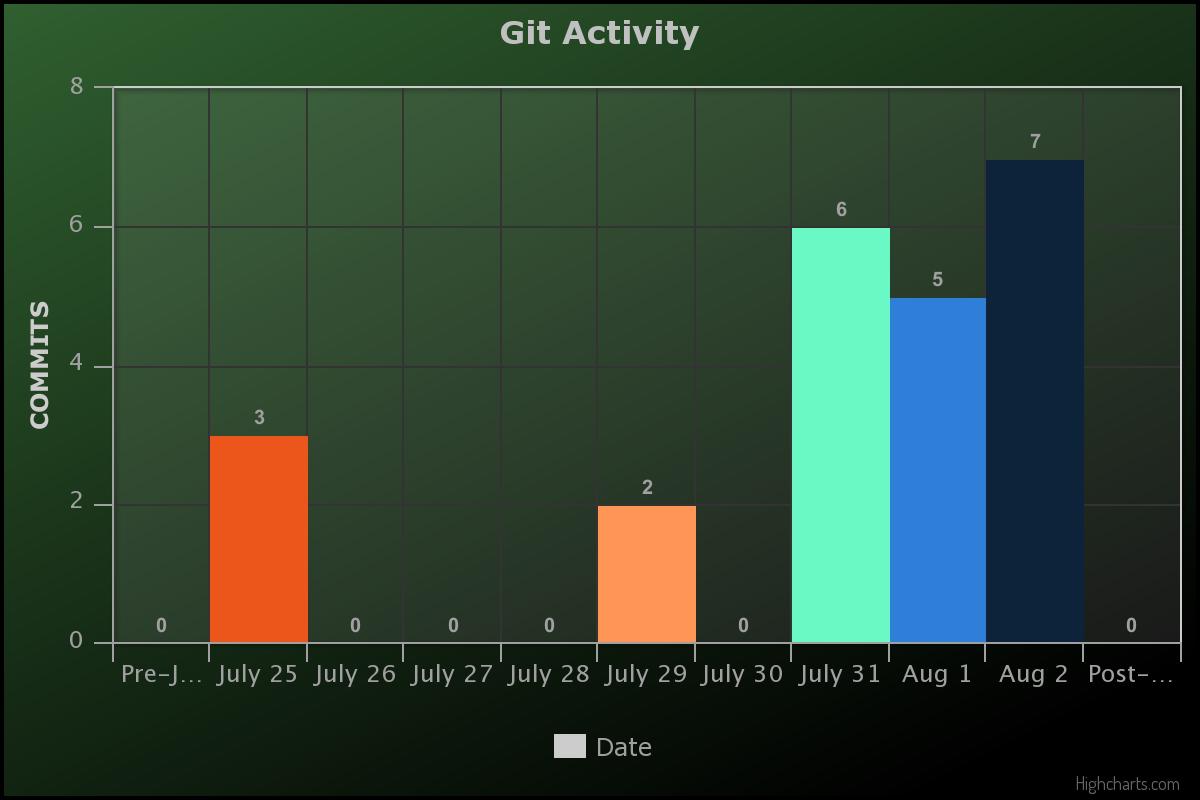
As for User interface, the sophistication of the intended UI design for this project in comparison to the original minesweeper on Windows computers is objectively on par. It doesn’t look amazing but neither does the original. The intention is to give the design an interesting theme and charm and this is possible with p5. Presently, the standard minesweeper game has an ocean them and the hex game will likely be a bee hive. Colour sweeper will need some more thought on design once the gameplay mechanics become clearer.

**Testing**

Up until this point, the type of testing utilised has not been formalised. It is essentially trial and error. The entire standard minesweeper game has been tested for errors. Each function has been tested throughout and upon completion and when a completed function or element is introduced to the overall game, the game is tested to ensure successful integration. If something does not work or a problem is caused, the cause of that problem is determined and a solution is developed. This solution is then tested against the entire functionality of the program and once everything is working as intended, a new function will be created and introduced. In this way, new solutions will not affect the efficiency of previous ones.

This is how testing will take place for the future of the project. Perhaps a more formal testing method should be implemented in order to test every possible case within the game project and perhaps this would make development more efficient. At the rate however, it is reasonable to believe that a durable application will be produced utilising current methods. If it comes to light that current testing methods are unable to fulfil the requirements in terms of playability for the minesweeper game, other testing methods may be implemented to account for this.

**Version Control**

****

There appears to be a significant spike in effort according to the history of commits documented. Development of the standard Minesweeper game took place over the space of time between July 25 and August second. No progress toward further development has taken place at this point. That is not to say that significant effort has not gone into software engineering as a whole, simply, other aspects of Engineering have been prioritised over the minesweeper implementation up to this point. This effort has gone towards understanding various Engineering concepts such as modelling, various architectures, many separate development environments and tasks.

With the production deadline approaching, a fresh spike of effort dedicated to the project is likely to occur. There is still a plethora of effort which will be expended on other concepts surrounding Software Engineering during this development period, however, that will take a back seat to the development of this particular application.