Assignment 1: Project Plan

Team: Waterfall

Unit: FIT2101

Members

Jay Requizo
jreq0001@student.monash.edu, 28770641
Matti Haddad
mhad0002@student.monash.edu, 29708966
Phillip Roodt
proo0001@student.monash.edu, 30564425
Sothearith Tith
stit4@student.monash.edu, 27208001

Google Docs:
https://docs.google.com/document/d/1pEkCNIE4VYtrX6gSJb4B0bHdvAK4jAu4J2d9iesQG34/edit?usp=sharing

Contents

Deliverable 1: Project Plan

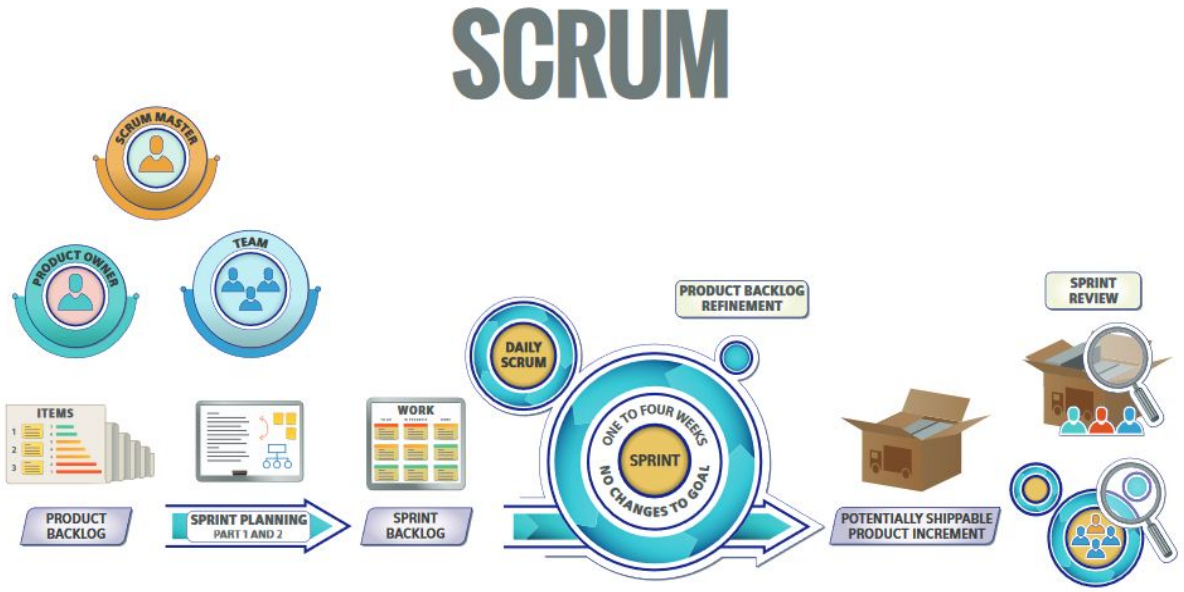
Vision Statement

The purpose of this project is to provide a tool for the user to display Git repository information in an easy to access and understand manner. That said, unlike traditional tools such as built in gitLab tools where all the necessary information and statistics can only be retrieved via in depth knowledge of gitHub, our web app would show it all with minimal effort. Furthermore, GitWorking is our tool that will clear these technical obstacles and save ones precious time with it's intuitive design. That is, by following only two simple steps, the user can have all wanted information and statistics in the form of an interactive graph, which is color coded in a way that allows anyone to easily understand what is going on and what was contributed by whom.

Team Profile

Name	Role	Contact Details
Jay Requizo	Product Owner and Lead Developer: Responsible for overseeing code implementation and review of submitted code; and Responsible for guiding the less experienced members to support their development.	jreq0001@student.monash.edu Discord
Matti Haddad	Scrum Master: Responsible for overseeing and directing team meetings, organisation and reinforces Scrum processes and Agile methodology throughout the project.	mhad0002@student.monash.edu Discord
Sothearith Tith	Developer: Part of the development team.	stit4@student.monash.edu Discord
Philip Roodt	Developer: Part of the development team.	proo0001@student.monash.edu Discord

Process Model: Scrum



Throughout our project, standard Scrum Methodology will be used as our process model. That is, an iterative process where the team works to add incremental value to the end product through each sprint. Moreover, due to funding, project timeline and team size, Scrum was chosen over other Agile methodologies, such as Lean, Prototyping and Waterfall. In addition, said other methodologies require more resources and need less constricting time constraints; we have a strict schedule with established deadlines. Furthermore, Scrum allows us to change the product features dynamically according to evolving customer needs and to frequently deliver incremental updates by collaborating with the customer which will inevitably enable us to deliver a high-quality product.

Alterations

Due to team size, scheduling and time constraints, alterations are being made to our Scrum methodology.

- Daily scrums will be done twice a week face-to-face for 15min (time-boxed) on Tuesdays and Fridays.
- The team will consist of four developers, therefore, the Scrum Master will be responsible for some degree of software development.
- Face-to -face communication with the Product Owner will only be done once a week on Fridays.
- Team velocity cannot yet be reached as more sprints are required to reach a decisive estimate.
- Sprint meetings, reviews and retrospectives will be shorter due to time constraints.

Roles

Standard Scrum roles:

- Product Owner
- Scrum Master
- Team Developers

Sprints

Sprints will last two weeks each starting Friday mornings and ending Thursday evenings. Also, these “sprint meetings” will take place on Fridays for thirty minutes with each involving completing a set of items from the product backlog. “Story point” will be the used unit to estimate the effort of each item. Moreover, a burndown chart will be used to keep track of backlogs during the sprints. At the end of each sprint, there will be a sprint review for one hour (time-boxed) with the Product Owner showcasing the demo if possible. Finally, a sprint retrospective for thirty minutes will be done to enable improvements for the next one.

Definition of Done

The feature/user story should pass:

- Unit testing with presume output and no error
- Code inspection processes
- Black and white box testing with presumed output with no errors
- Integration testing with correct presumed output with no errors
- Regression testing without any errors and presumed output
- Testing against acceptance criteria
- All bugs discovered should be reviewed, fixed and labeled as “closed”
- Sprint backlog should be updated
- Technical documentation should be updated. E.g. tests, comments… etc
- Approval by Product Owner

The final product should have:

- All the code completed and working
- No unintegrated work remaining
- The code pass system testing with output compliance and requirements
- The code pass acceptance testing
- Meet all the acceptance criteria
- All issues raised during development are resolved
- The code should go through Quality assurance
- User documentation(manual) is updated
- The technical documentation(test case, comments) is updated
- Approval from product owner

Task Allocation

Following the Scrum Methodology, cross-functionality and self-management are favoured over a fixed-specialization and hierarchical structure. Therefore, we aim to have a cross-functional and self-organising team. However, the Scrum master will review any decisions to make sure that they do not overlap with other items or impede the work of other team members .

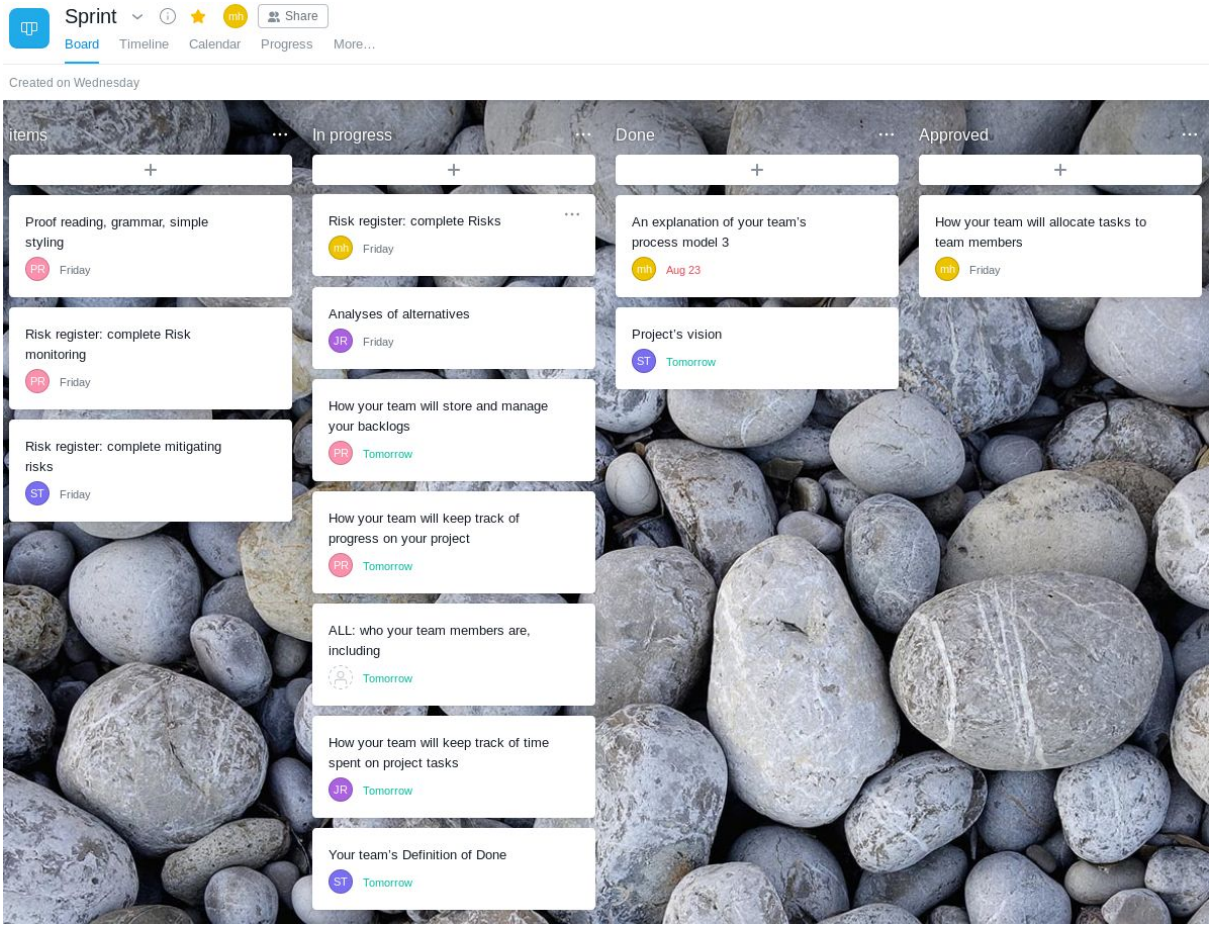
- Self-organisation:
    - Team members will decide on how many items go into the sprint backlog. Furthermore, each team member will self-assign items from the sprint backlog according to the set priority and the team member skills.
- Cross-functionality:
    - Team members will have a high degree of freedom and accountability with which task they want to work on, but at the same time, their skills play a part in their role.

Developer	**Primary skill**	**Secondary Skill**	**Tertiary skill**
Jay Requizo	Full-Stack	Database Admin	Quality Assurance
Matti Haddad	Back-End	Web Design	Testing
Philip Roodt	Quality Assurance	Back-End	Front-End
Sothearith Tith	Web Design	Front-End	Quality Assurance

Following scrum process model, multi-learning will be encouraged to keep the team motivated on the project and account for developer turnover due to lack of learning opportunities. On the other hand multi tasking will be discouraged to avoid the costly drain of divided attention and context-switching.

Progress Reporting Structure

Backlog Management



Asana will be the main backlog manager.

That is, each team member will add tasks that need to be done to the “items” list which will be reviewed by the Scrum Master. In addition, when an item is added and passes reviewing, any team member who is able can assign it to themselves and move it to in progress once they start to work on it.

Moreover, each team member is responsible for checking whether their item passes all the Definition of Done requirements before passing it into done, after which the Lead Developer will review the item and check its validity according to the definition of done. That said, only when the Lead Developer has approved it can it be moved from done into approved.

Each sprint will have a dedicated slot in Asana which will contain all necessary information and tasks for that sprint. After each sprint, these pages will be archived and all approved tasks will be moved into the next sprint to add onto what is being done after.

Moreover, in addition to Asana being used as the backlog manager, Google Sheets will also be an option.

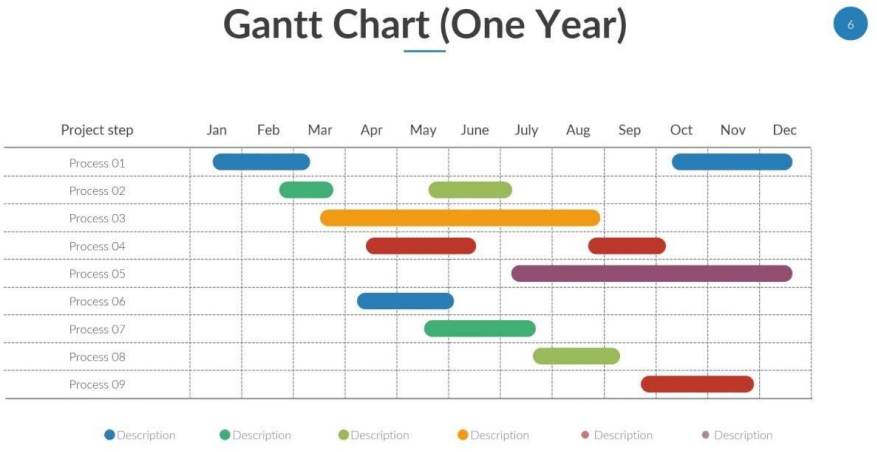
				New Estimates of Effort Remaining at end of Day...					
Product Backlog Item	**Sprint Task**	**Volunteer**	**Initial Estimate of Effort**	**1**	**2**	**3**	**4**	**5**	**6**
As a buyer, I want to place a book in a shopping cart	modify database	Sanjay	5	4	3	0	0	0	
	create webpage (UI)	Jing	3	3	3	2	0	0	
	create webpage (Javascript logic)	Tracy & Sam	2	2	2	2	1	0	
	write automated acceptance tests	Sarah	5	5	5	5	5	0	
	update buyer help webpage	Sanjay & Jing	3	3	3	3	3	0	
	. . .								
Improve transaction processing performance	merge DCP code and complete layer-level tests		5	5	5	5	5	5	
	complete machine order for pRank		3	3	8	8	8	8	
	change DCP and reader to use pRank http API		5	5	5	5	5	5	

Project Task Timesheet Management

In order for the team to keep track efficiently of any and all tasks, a Gantt Chart will be used to document any time spent on any tasks and processes.

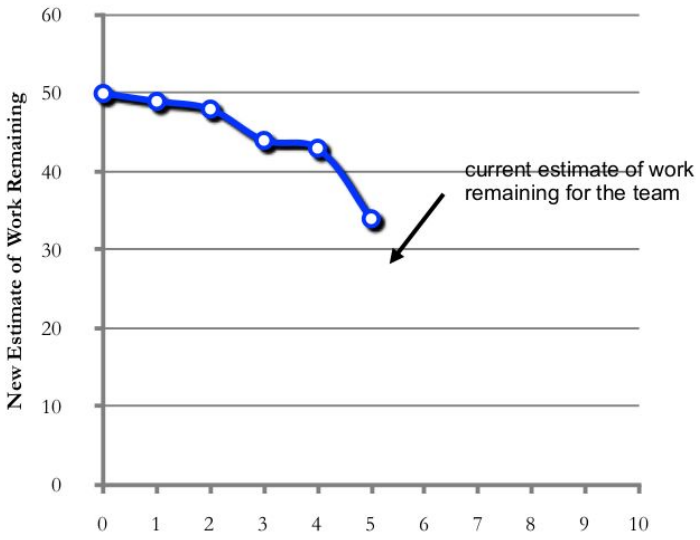
If a team member was to start working on a task, they would first have to document in Google Sheets when they start, what day and then until they finish on what task.

Furthermore, the information entered into the Google Sheets document will be concatenated after every sprint into a full Gantt Chart that will be used as a reference for how long was spent overall on each task and process. This chart will be used to track the efficiency and speed of the work of the team members which will help to plan the following sprints in a better manner than previously done. The Gantt Chart will allow the team to monitor closely deadlines and when it is predicted that we will reach them, and whether or not we are able to finish a sprint in time.



In addition, our team will be using a burndown chart to keep track of the progress of the project which supplements the usage of a Gantt Chart as the burndown chart will be essentially the Gantt Charts merged into one for easier tracking of progress. The burndown chart will effectively tell the Project Lead whether the team is on track, or if we are falling behind or even ahead of time. That said, the use of built in GitHub features will be used to keep track of individual time spent on tasks and processes.

Figure 6. Daily Updates of Work Remaining on the Sprint Backlog



Overall, we will use…

- GitHub time tracking tools.
- Google Sheets document.
- Gantt Charts.
- Burndown Charts.

They will concatenate into each other for an efficient manner of time keeping.

Deliverable 2: Analysis of Alternatives

Introduction

This document will detail the criteria by which we select what tools our team will utilise during our project development. That is, the first section of this document will outline what frameworks and tools are required and the specific capabilities required by our alternatives, while the following sections will compare the different options available for each tool required by our team.

Required Tools and Frameworks

Target Platform

We discussed with our client any requirements and preferences for their target platform. Our client did not specify any preference for a target platform or any requirements but did mention that the interface required some interactivity. Our application must also be able to interface with the GitHub API to enable us to fetch data. Moreover, this necessity of an internet connection better suits the criteria of web applications, which is run in a browser and is more flexible upon which devices it can run, and provides easier development of user interfaces and interactivity with the displayed content.

Criteria:

- Web API interfacing
- UI interaction
- Familiarity amongst team members

Target Frameworks and Languages

Due to the fact that our language requires interactivity and web interfacing, our programming languages are limited to those that provide support for the development of web applications. Furthermore, supplementing our team with familiar languages will enhance our productivity by diverting time from learning and understanding the fundamentals of the language towards understanding the chosen frameworks and to actual feature development. The web UI will have to be handled via HTML and CSS, and if time permits, jQuery to enhance UX. This leaves us with two possible options for our backend programming languages, JavaScript and Python.

Criteria:

- Web API interfacing
- Familiarity with the language (but not necessarily the libraries)

JavaScript provides a range of alternatives for web development, and one popular option is the MEAN stack, used for full-stack web development. Along with this, ChartJS could also be

used for graphing, thus keeping the whole development process within one language. Django is a web application framework library written in Python that focuses on easing the creation of larger complex data-driven websites. Other possible options for Python web application frameworks include Flask, and Pyramid.

A possible option would be to utilize both languages, opting for the more suitable framework where necessary. This, though, may pose unforeseen risks and therefore requires more investigation towards the integration of both languages and frameworks which will be discussed in the analysis section of this document.

Target IDE

The team's preference for its development environment stems from how much experience each member has with the system and its version control system, which is necessary for collaboration and version control. Additionally, the IDE must be tailored towards the target platform and language we are working with. It would be preferable if the IDE was obtainable with a student license or without the requirements of a license to reduce expenditure.

Criteria:

- Provides collaboration support through git
- Cross Platform(Linux, Apple, and Windows)
- Familiarity
- Free or open licensing

Target Project Management Software

Backlog Management

Task allocation and tracking is important during the sprint cycle to ensure the timely completion of deliverables. Furthermore, it provides data points for evaluation during sprint reviews to calibrate sprint pace. Our options for backlog management are Trello and Asana.

Criteria

- Free or student-license accessible
- Collaborative
- Features
- Familiarity

Assessment of Alternatives

Target Platform

Web Application

Web API Interfacing	Very easy in Web Applications.
UI Interaction	Easy to achieve with HTML, CSS, and if further interactions are required, frameworks like jQuery cut down on development time greatly.
Familiarity amongst team members.	Familiar amongst 3 out of 4 team members.

Desktop Application

Web API Interfacing	Requires permissions from OS.
UI Interaction	Needs to be built from the ground-up.
Familiarity amongst team members.	Not as familiar to team members.

Comparative Analysis

Web applications are tailored towards a more general use therefore can be accessible from a larger variety of devices. In contrast to this, desktop applications are developed to run on a specific environment. This greatly cuts down on the amount of testing required to ensure features are stable and functional, but UI development may end up requiring more time to complete and managing network calls may be more restrictive, posing a bigger risk.

Furthermore, the web development process is better understood by our team compared to full-stack desktop application development. The languages involved with web development are also more familiar amongst our team members.

Some risks to be aware of with developing a web application are deployment and upkeep. Desktop applications require very little upkeep after deployment, while web applications require an extra process during deployment and further upkeep in the form of resource cost for hosting solutions. These risks posed by developing for a web application can be mitigated to an acceptable degree, as compared to the risks presented by developing for a web application which require mainly time to reduce or mitigate. Therefore, our target platform preference is a Web Application.

Target Frameworks and Languages

This section assumes that our target platform would be a web application.

Javascript

Web API Interfacing	Full-stack development achievable with JavaScript alone, along with third-party library support for extended features.
Familiarity with the language.	Familiar for 3 out of 4 team members.

Python

Web API Interfacing	Full-stack development achievable with Python alone, third-party library support is less extensive as JavaScript for Web Applications.
Familiarity with the language.	Familiar to 4 out of 4 team members.

Mixed-Use

Web API interfacing	Complete coverage
Familiarity with the language	Negotiable

Comparative Analysis

Javascript is a tried-and-tested web scripting language and therefore posts the least risk due to its extended support. Furthermore, it also has an extensive variety of libraries which can potentially cut down production time. Python is familiar for all our team members, and provides nearly the same level of completeness in terms of Web Application capabilities. A mixed-use case could cover for unexpected programming feature requirements, but also poses greater risk towards development synchronisation and compatibility issues. Both languages require setup for its execution environment.

Because Python is more familiar for our team members, and a level of unfamiliarity with associated web framework libraries exist for both languages, Python comes out as the preferred alternative.

Target IDE

Pycharm

Git Repository Support	Extensive GUI for version control interfacing.
Cross Platform	Able to run on all platforms.
Familiarity	Familiar all team members.
Free or open licensing.	Community edition (free), also available through student license.

Visual Studio Code - VS Code

Git Repository Support	Very basic UI, not as good flow for control process.
Cross Platform	Able to run on all platforms.
Familiarity	Not as familiar for all team members, git interface requires learning to use.
Free or open licensing.	Free.

Comparative Analysis

PyCharm is an IDE specifically designed for Python development, therefore is language-agnostic. It also provides complete features for version control management, which is also familiar for all four team members. A community edition and a student version of the software is available for installation.

Visual Studio Code is a free IDE designed for the creation of web applications. While it is targeted for web development, minimal setup is required for it to support development in other languages. The interface is designed to be minimalistic and git processes are normally done through the built-in terminal. It has a basic interface for committing but conflict resolution is normally done manually. This IDE is less familiar amongst our team, therefore our preferred option for our IDE is PyCharm.

Target Project Management Software

Backlog and Timesheet Management

Asana

Free or student-license accessible	Free for both desktop and mobile
Collaborative	Full collaborative support
Features	Task allocation only accepts a single user Supports deadline assignment
Familiarity	Familiar for 1 out of 4 team members

Trello

Free or student-license accessible	Free for both desktop and mobile
Collaborative	Full collaborative support
Features	Tasks can be collaborated on, but no concrete allocation Supports deadline assignment
Familiarity	Familiar for 2 out of 4 team members

Comparative Analysis

Both applications are nearly objectively equal in the view of our criteria. Task allocation is an important feature and is easier to use in Asana compared to Trello. One notable missing feature amongst both applications are task weighting, which will have to be done manually. Therefore Asana comes out slightly on top due to it being more familiar for our team and its features are slightly more suited for our needs.

Deliverable 3: Risk Register

Introduction

The following risks apply to the Scrum process model.

Risks

Level of severity: very low, low, medium, high, very high

Risk Factor	**Likelihood**	**Impact**
Product owner pulls out or changes.	Very Low	High
High turnover on the project team.	Low	Very high
Gitlab or google drive blackout.	Very low	Very high
Physical tool failures such as laptops, chargers, monitors... etc.	Low	Low
Software failures such as chrome, IDE... etc.	Low	Low
Significant or additional changes of requirements that make sprints go to waste. I.e. a change of scope.	Medium	High
Absence of essential Scrum members.	Low	Low
Mental or physical health risks of project timeline.	Medium	High
Misunderstood or poorly defined requirements.	Medium	High
Optimistic schedule and deadlines.	Medium	High
A project breaking bug in Python or it’s libraries that stops development.	Very low	High
A project breaking bug in the web platform that stops development.	Very low	Very High

Risk monitoring

Risk factor	**Monitoring Strategy**
Product owner pulls out or changes.	Ask the product owner to inform the scrum master whether he/she is going to pull out or change any time soon.
High turnover on the project team	Ask the employee to inform the team leader as early as possible if they are leaving the team. And keep a record of what this person activities
Gitlab or google drive blackout.	Constantly keep a lookout for any notification from git or google drive about any disruption and when the GitLab or google drive is most busy.
Physical tool failures such as laptop, charger, monitor... etc.	Keep records of when the tool is inspected and advise the team to raise the issue as soon as possible if the tool is not working properly.
Software failures such as chrome, IDE... etc.	Advise the team to raise an issue if the software is acting abnormally like slowing down or not very responsive.
Significant or additional changes of requirements that makes sprints go to waste. I.e. a change of scope.	Notify the product owner before starting the sprint and ask for any changes or additional requirements, if product owner has one.
Absence of essential Scrum members.	Record how frequently it happens and advise the member to notify the Scrum master as early as possible.
Mental or physical health issues of project timeline.	Constantly observe each team member if they are acting strangely. I.e. not performing well or is depressed. Advise the team member to raise any issue they are facing, emotionally or physically.
Poorly defined requirements.	Check user stories developed by the team after getting the requirement to see if the user stories is following INVEST. Observe if the product owner keep on changing the requirement or is not confident in the decisions he/she is making.
Optimistic schedule and deadlines.	Keep the Burndown chart updated and observe the trend. I.e. number of tasks remaining and number of

	days until the deadline. Observe if any team member is lagging behind due technical problems.
A project breaking bug in Python or it's libraries that stops development	Advise the developers to raise any issue if they encounter any using the libraries or Python and record the frequency at which it happens.
A project breaking bug in web platform that stops development	Advise the developers to raise any issue if any function cannot be implemented on web platform and keep a lookout for any updates.

Risk Mitigation

For more in depth discussion on alternatives due to a risk or issue arising on a certain feature, program or task, refer to the Analyses of Alternatives.

Generally, risks are mitigated by putting in place strategies to prevent it from happening or strategies that reduce its severity when affecting the team. Some of these categories include:

- Avoidance: Not implementing any risky plans that might backfire.
- Reduction: minimize likelihood or impact of risk
- Transfer: Transfer ownership of the problem to a third party through insurance.
- Acceptance: Integrate an unavoidable issue into the work plan and build strategies to work around it and achieve the same results as if it weren't there.

Risk factor	**Mitigation strategy**
Product owner pulls out or changes.	Establish with the current owner if they are set to stay. Not much can be done if urgent issues come up.
High turnover on the project team.	Allocate a few days to the new developer to view and study the activity record of the last developer (if available) or the previous items allocated to the last developer, to help get the new developer up to team pace. Replan the sprint backlog if necessary.
Gitlab or google drive blackout.	To minimize migration cost: Have a clone of GitLab repository on GitHub . Have a clone of Google drive on Microsoft OneDrive.
Physical tool failures such as laptop, charger, monitor... etc.	Put back up measures in place in the case of such failures, back up code and documents on a Hard Drive on a daily basis as well as the cloud.
Software failures such as chrome, IDE... etc.	Use alternative software (refer to alternative of analyses) And plan sprint backlog to allocate some time to account for migration cost.
Significant or additional changes of requirements that makes sprints go to waste. I.e. a change of scope.	Perform a Spike. In addition remove secondary items(if any), focus on primary items only and replan sprint backlog to minimize cost. Any salvageable code or features should be kept. Nothing should be permanently removed however.

Absence of essential Scrum members.	Add a few extra days to the project timeline. Replan sprint backlog and allocate this person's task to team members of similar skill set.
Mental or physical health issue of project timeline.	Have a break sprint to release the pressure on the development team. If avoidance is not possible, make sure that the task is not dependent on only one person and allocate this person's task to team members of similar skill set.
Poorly defined requirements.	Perform a Spike to clear out any specification confusion.
Optimistic schedule and deadlines.	There is always the risk that a member falls behind, so, the team will make sure that all the features needed for full functionality of the new feature will be finished. In this way, we can keep track of team members' work. It allows us to organise our sprints in a timely manner for the Project lead.
A project breaking bug in Python or it's libraries that stops development	Refer to alternative of analyses.
A project breaking bug in web platform that stops development	Refer to alternative of analyses.

Target IDE

Pycharm vs VS Code

The two IDE's we have identified as most compatible with our project are VSC and PyCharm. Our team has chosen to use both of these environments as not every member will need the same benefits offered by one that are not offered by the other.

Target Project Management System

Sprint Backlog

Risk Factor	**Risk Mitigation and Monitoring**
Issues arise with Asana. Likelihood : very low Impact : low	At the moment, the team is using Asana as the backlog. However, if we find ourselves with too many complicated tasks to do and not enough functionality in Asana to communicate all the complexities to do with the tasks, then we will switch to Google Sheets.

	In the case that Google Sheets becomes unavailable for our use due to an issue with Google Drive, we will switch to Microsoft Excel which has the same functionality and can be used immediately as documents just need to be transferred from the Drive to Excel.

Other Risks

Loss of Team Members:

This team has already experienced the loss of a team member. In order to have successfully mitigated the negative effects of such an event on our team, we have re-assigned the tasks equally amongst the remaining team members and changed any necessary tasks in which they were previously involved. The risk associated with this loss was minimal as it happened soon into the project effectively lowering its severity.

Organizational Risks

Risk	**Likelihood**	**Impact**
Failure to comply legal / licence	Very low	Medium
Team dissatisfaction or lack of motivation	Medium	Medium

Risk Factor	**Monitoring Strategy**
Failure to comply legal / licence	Keep a record of the tool used and its licence i.e. the expiry date, does all tools require licence and comply with the term and condition.
Team dissatisfaction or lack of motivation	Observe if any team member showing any sign of depression or not pushing frequently, lack of communication or interaction between team members or not paying much attention.

Risk Factor	**Mitigation Strategy**
Failure to comply legal / licence	In the case that a tool does expire, steps will be taken to immediately renew that tools usage or if that is not possible, the analyses of alternatives will be consulted in order to find a different tool which serves the same purpose and can be put to use immediately.
Team dissatisfaction or lack of motivation	If signs of dissatisfaction are remarked, the team will act as a whole to help that team member in anyway that is feasible.

Technical Risks

Risk	Likelihood	Impact
Technical debt	Very low	Medium
Misunderstanding the capability of the target language	Very low	Low

Risk Factor	Monitoring Strategy
Technical debt	Observe the team member if they fully understand the requirements and have enough knowledge to complete the task. See if the requirement is fully reviewed and ready to implement. Are there any last minute requirement changes or added? Are there enough test cases and documentation? Did the code goes through review process after completion?
Misunderstanding the capability of the target language	Observe if the team is reviewing the target language carefully or are they just assuming.

Risk Factor	Mitigation Strategy
Technical Debt	In order to avoid the formation of any type of technical debt it is essential that the team keep on top of any organisational priorities and documentation throughout the duration of the project. In the case that the team does fall behind on any required documentation, as done previously, the whole team is let know and the steps are taken to catch up.
Misunderstanding the capability of the target language	The mitigation strategy best suited is to ascertain that all team members are comfortable with the chosen language in order to avoid any misconceptions or problems arising due to a lack of understanding or knowledge about the language.