THE PROJECT PLAN

Vision

The goal of our team is to provide our client, Ms. Kamalahshunee, a tutor at Monash University, with a software which assists her in marking students assignments. The software, temporarily dubbed ‘GitHub Inspector’, will extract data from a GitHub repository and provide a wide range of visual aids to more easily identify how much work a student has done. The software will be easily accessible from mobile devices and computers.

Team Members and their responsibilities

Group contact and sharing:
Facebook group chat will be the main form of communication. Google drive and Git Repository will be main forms or sharing information/files. Facebook messages are expected to be viewed and responded to as soon as possible, and at maximum, within a day.

Danesh:
Role: Programmer
Responsibilities: Checking of code, data manipulation
Contact:
    Email: dhew0003@student.monash.edu
    Whatsapp: +61 406604413
Yi Meng:
Role: Programmer
Responsibilities: Backend development
Contact:
    Email: ylau0009@student.monash.edu
    Whatsapp: +60122139807

Zi Ling:
Role: Scrum Master
Responsibilities: Assist the team in making decisions
Contact:
    Email: ztan0027@student.monash.edu
    Whatsapp: +60169516413

Process model breakdown

We will be using a variant of Scrum, as some aspects of it are unnecessary for our project.

- Sprint planning meeting
    a. At the beginning of each sprint, we will have a meeting with the client to determine which items from the backlog will be selected to be into development. In addition, any changes issues raised by the client should be resolved here.
    b. Tasks are also allocated during this time once items from the product backlog are chosen for the sprint.

- Sprint Review

    a. A sprint review is conducted in the presence of the client. The development team demos the working sections of the product to give the client an idea as to what the final product could look like, as well as the completeness of the project.

- Sprint Retrospective
    a. A sprint ends with retrospective. The client is not present for this. The scrum master and team go over the events of the week, discussing what worked well, what didn't, and how it could be improved on, in preparation for the next sprint.

- Individuals involved in the project will either be a Scrum Master, team member, or a client.
    a. The client is who the product is for. The general direction and decision on features of the product will be from them.
    b. The Scrum Master is the team leader and the mediator between the team members and the client. They will ensure the client and team get along, as well as negotiate on the terms of development.
    c. The team members are the core of the production. They will be responsible for the development of the product.

- Sprints
    a. Scrum team will be working on the project in a series of sprints. Each sprint is two weeks long.

- Agile Board
    a. The Agile board is to help keep track of the progress of the project. It will be split into three sections; To do, in progress, and done. The ‘to do’ and ‘progress’ columns will have a limit on the number of items in them to curb the number of features being worked on at any given time. In Scrum, there is no limit on each column.

- Group meetings
    a. Group meetings are no longer daily, as compared to Scrum, and instead will be held every three to four days. The amount of work that can be accomplished within a single day is not significant enough to warrant a daily meeting. In addition, due to the constraints of the client, they are not required to be present.
    b. The meetings would be held to discuss important and urgent matters and to keep up to date on each team member’s progress.

Task Allocation

Tasks will be allocated during the sprint planning meeting. The team will discuss which items in the product backlog should be done. If an item is considered too big, the team will have to break it down into smaller parts and select the parts they will need to do for the sprint. The tasks are then place them in the ‘to do’ column of their Agile board. They then decide and choose which tasks they would like to assign to themselves. In this way, this allows the team to gain experience in doing a diverse range of tasks in order to help them become familiar with the entire project. However, each member is advised not to take on too many at once or too few tasks. This is to ensure that everyone has a balanced workload and does a fair share of the work for the sprint. If the team has difficulty in deciding, the Scrum Master will help out. Any disputes will be settled by the Scrum Master as well.

Progress Tracking

The progress of the project will be recorded via Google Drive through changes made in any of the documents and the GitLab repository through commits. Access to Google Drive, the GitLab repository and Kerika will be shared with the client.

Weekly progress will mainly be tracked via Kerika. Using the website's Agile board, it will help the team track how many user stories are in the 'done' column at the end of the week, and allows progress to be tracked live through the usage of its columns.

The team will update on their individual progress and discuss any issues that arise and resolve them (if possible) during the weekly sprint meeting. Outside of meetings, team members will update and discuss mainly through the Facebook group chat.

Backlog Management

The product backlog (list of features desired by the client and user stories) will be stored using Kerika as it also doubles as the team's Agile board. Each item will be written as a task and added to the 'to do' column of the Agile board if they are chosen for the sprint. After these have been done, they are moved to the 'done' column. This process will be repeated for each sprint.

Time spent on project

Time spent on the project will be tracked either by hours or by days on a separate document.

In terms of hours, team members record the number of hours a day they work on the project. Within the day, they record how long they spent on each of their tasks and total up the time taken. This can help to identify which type of tasks are difficult and take up the most time to do so that the team can estimate and manage their time better throughout each sprint.

In terms of days, another way to keep track of the time spent is to check when team members move their tasks from 'in progress' to 'done'. The date they have moved them is recorded in the tasks' history tab on Kerika. Members should ensure that they do this immediately once they considered their tasks as finished.

In addition to the above, the number of commits throughout the sprint will also be inspected. The commits of each team member will be checked based on whether the changes they made are major or minor along with the time between their previous commit and the latest one they pushed.

However, each of the ways mentioned above requires the honesty of the members, as the number of hours or days worked on cannot be accurately gauged through GitLab, and nor is it an indication of the quality of the work.

Definition of Done (DoD)

Definition of Done (DoD) is a general criteria that is created to estimate the completion of a project. It defines a clear separation between work that is completed, and work that is incomplete and needs to be worked on, through a checklist or list, to provide quality assurance, reliability, and value to an artifact. By using a DoD, it provides a standardised expectation of quality to the team and product owner. The team has a clear vision of what is expected of them, and can easily demonstrate their capacity to complete assigned tasks, via use of the DoD. The DoD is intended to be used by the development team to assist them throughout the entirety of the project. It is a living document and can be edited as necessary to accommodate new requirements or analysis.

Definition of Done checklist for User Stories

- ☐ User story matches the INVEST criteria
- ☐ Assumption of the user story is met
- ☐ Features are approved by Product Owner
- ☐ Documentation of the code's user story is written and updated when needed
- ☐ QA is performed and issues are resolved
- ☐ The code produced works as intended for each required functionality

Definition of Done checklist for Sprint features

- ☐ "To do's" are completed for the latest Sprint
- ☐ Product backlog is updated
- ☐ Most bugs in the code are fixed
- ☐ Features for the Sprint are well documented in the code
- ☐ DoD for user stories are met
- ☐ Feature meets specified client requirements
- ☐ Unit testing/Integration testing/System testing pass without problems
- ☐ Testing on required platform passed
- ☐ Peer code review is performed

Definition of Done checklist for Release

- [ ] Code and required features are complete
- [ ] Acceptance criteria from the Product Owner is met
- [ ] All tasks in the product backlog has been moved to done
- [ ] Obtained approval from client (and other stakeholders if any)
- [ ] QA has been done and major issues have been solved
- [ ] No unintegrated work in progress has been left
- [ ] Final testing on required devices has been finished

Risk Register

	Negligible	Minor	Moderate	Significant	Severe
Very likely	Medium Low	Medium	Medium High	High	High
Likely	Low	Medium Low	Medium	Medium High	High
Possible	Low	Medium Low	Medium	Medium High	Medium High
Unlikely	Low	Medium Low	Medium Low	Medium	Medium High
Very Unlikely	Low	Low	Medium Low	Medium	Medium

The risk matrix above is used as a guide to estimate the likelihood and impact of a risk. The risk register is subject to change and any risks identified in the future will be added here.

Risk Factor	Likelihood and Impact	Contingency Plan	Prevention Strategy
Technical risk - Uncertainty in time taken to learn new techniques to implement features and APIs	Likely and moderate	In this case, team members can only try to learn as much and as fast as they can. If the API is well-documented, it may help to speed up the process of learning how to use and incorporate it. Otherwise, the team may have to set aside 2 or 3 days to study the documentation and experiment with the API before continuing with the	Before the next sprint begins team members could go through the latest set of features that need to be included in the next sprint and if possible search for APIs that match the need of the features.

		project.	
General risk - Requirements of the client might get misunderstood by the team	Possible and significant	If possible, arrange to meet with the client to ask about any requirements that have been misinterpreted. If the client is not available to discuss in person send a message or an email asking for more clarification.	Team members could go through all of the requirements that have accumulated throughout each sprint as well as any user stories with the client one more time at the end of every meeting or review if possible. If there is any uncertainty, the team can ask questions to help make the requirement more specific so that it is easier to identify features to be included in the code
Organisational risk - A team member falls sick in the middle of the project	Possible and moderate	Take medication or visit medical centre. Inform all team members. Split workload of sick member among other team members to assist them in regaining their health.	Ensure everyone has medical checkups, sleeps early, takes vitamins and has eaten proper meals.
Technical risk - New features being implemented may conflict with existing executables	Unlikely and moderate	Check through existing features in the code that could be closely related with any new requirements. If a feature becomes too big as a result, break it into smaller parts when implementing the new feature	The team should try to reduce dependency among features. Ensure that each feature in the code is mostly self-contained and that the new code does not break anything. If there are features that do depend on one another, ensure that

			there is minimum coupling among them.
Organisational risk - The risk management plan is forgotten and neglected	Unlikely and minor	Update the risk management plan As soon as possible.	Check and update the risk management plan as required weekly
Technical-Github servers go down	Likely and minor	Do your section of the project offline, then meet up to discuss on merging.	Ensure your code is backed up on your computer, and communicate with team members.