Friday.com

Group 19

Yan Zhen He, Victor Huang, Rehan Ali,
Melvin Pramode, Bingmo Li, Sutrisno

Table of contents

Project Plan

Our Vision

For companies and organisations who desire maximum proficiency. The chartZ intends to streamline the Scrum process to couple productivity with effectiveness and efficiency.

Team Members

Yan Zhen He:

- Contact: yhee0047@student.monash.edu
- Roles: Product Owner / Scrum Master
- Responsibilities: Helps make the team come to a consensus on decisions about the project, provides an environment for the team to thrive without any interferences, handle's disagreements within the team, and makes sure that there are appropriate resources available for the team to conduct project work.

Victor Huang:

- Contact: vhua0006@student.monash.edu
- Roles: Technical writer
- Responsibilities: Providing a clear and easy to understand manual or guide to using the program properly. Assisting with programming.

Rehan Ali:

- Contact: rali0010@student.monash.edu
- Roles: Frontend developer
- Responsibilities: Providing the clients with an easy to use and function app/website that meets all their requirements.

Melvin Pramode:

- Contact: mpra0021@student.monash.edu
- Roles: Programmer
- Responsibilities: Providing the base code of the application to ensure program functionality is achieved.

Bingmo Li:

- Contact: blii0059@student.monash.edu
- Roles: UX Specialist
- Responsibilities: UX Specialist are responsible for translating the user requirements into the software

Sutrisno:

- Contact: sutr0001@student.monash.edu
- Roles: Programmer
- Responsibilities: Assisting other programmers within the team and ensuring that the software follows design patterns to ensure an extensible and robust software.

Process Model

The model that we will be utilising as a team is a cut down Scrum model which is an agile framework that focuses on teamwork and collaboration. It is based on continuous improvement and adjustments where there isn't a rigid structure, this allows members to adapt to changing conditions and client requirements. The scrum processes consist of collaborating with the client and creating a product backlog which is a list of all the desired features that the client wants. Once the product backlog has been created sprints are conducted, which are short periods of time where a set amount of work is completed. Multiple sprints are conducted which have their own sprint backlogs. Once the sprints are completed the product is reviewed by the members and the clients to confirm that the requirements are met.
As scrum is a fluid structure it will allow us to tailor this model to our project's circumstances and the hurdles we face and depending on the size of the project we can alter the amount of sprints we conduct. Since we are regularly having discussions and reviewing the work completed, we can make sure we fill the clients requirements hence making sure that this process model will succeed.

Definition of Done - A checklist

- Finishing an allocated task, and making sure that it works without major issues
- Making sure that the code is readable, and there are comments to explain thinking
- Making sure that there is documentation regarding the author of the code
    - In the situation that more explanation is required for a better understanding, the author can be contacted
    - The usage of git to track the author could also be used

Task Allocation

Base the task allocation on team member's roles, as well as their strengths. All allocated tasks should be predicted to be completed at the end of the sprint.

Progress tracking

Have regular meetings with the team and make sure everyone is on track with their tasks (stories). In the meeting, issues regarding certain tasks as well as a general progress report will be made, allowing the team to check their progression as they approach the end of the sprint.

Backlog management

The more important functionalities of our software will be prioritised first and allocated to team members. Less important tasks are allocated to be completed after all important functionalities have been resolved.

Time tracking on project tasks

Time tracking on project tasks will be undertaken by commit messages. Based on the commit message policy given below, team members will be required to record and input the amount of time spent prior on the work within the commit.

These commit messages will also be reflected in a separate spreadsheet in the official documentation folder. Members will be required to write a single line description of what they achieved as well as the total amount of time in hours spent on the task.

GIT Policy

GIT Branch structure

Master branch stores the final working segments of the code.
Another branch for testing and writing code (“Prototype Branch”) for the software
Other individual branches for each member’s code.

Individual team members will write their respective code in their respective branches, once completed, the code will be merged into the prototype branch, where faults and failures can be fixed. Test cases will also be conducted in the prototype branch. Once the prototype is completed, it will be merged into the master branch as a “live” working version of the software.

Handling Merge Requests

Merge requests should be overlooked by affected team members,a consensus should be agreed upon by the team and finalised by the Scrum Master.

Commit Message Policy

The following agreed upon structure:

General Commit

[Working_status][File1/File2/File3][Time_spent]
One sentence summary of changes

- Brief change of each file

Bug Handling Commit

[Fix/Not_fixed][Associated Files][Time_spent]
Brief associated bug description and description of faults

- Changes to fix the fault/failure

Process to follow when merge conflict happens

1. Contact the Scrum Master and colleagues associated with the conflicted files. The rest of the team should then be notified.
    - All conflicts should be handled by colleagues associated with the conflicting files. Replies should be within the same day of the conflict occurrence.
2. Create a new branch from the testing branch labelled “merge-conflict”, all conflict resolutions should be made using this branch.
    - This allows the current conflict to not interfere with the work of unrelated colleagues
3. Wait for a reply and resolve conflict
4. Merge the conflict branch back into the testing branch

Analysis of Alternatives

Preferable programming language and platform

A web application utilising JavaScript and HTML will be used for the final product.

Alternative platforms

Creating a desktop or mobile application is not required for our end user’s use cases, and the complexity of desktop or mobile applications is too high for the team to be able to deliver working software in the current time constraints.

Alternative languages

Our team is more familiar with Python, however it is not supported by many web browsers, unlike JavaScript and HTML. Since our end users are utilising the web application of various browsers, ensuring that this application works on all of our clients’ systems will ensure the best impression of our software. The team is also more familiar with running JavaScript and HTML with browsers.

Targeted browsers

- Chrome: We choose Chrome because it is one of the most commonly used browsers in the world and all the members of the team have access to this technology which makes it suitable for testing and deployment.
- Safari: We choose Safari because it is the main browser for Macintosh computers.

Analysis of Risks

Risk table

Risk No.	Risk Description	Likelihood of risk	Impact if risk occurs	Severity	Mitigation
1	Not completing the project on time	Medium	High	Catastrophic	Creating a timeline and meeting all the deadlines
2	Poor quality of code	Medium	High	Hazardous	Programmers should have a meeting and refactor code
3	Not meeting requirements	Medium	High	Hazardous	Review requirements with team
4	Team members not communicating	Low	Medium	Hazardous	Consult the tutor for advice on how to communicate
5	Unforeseen circumstances affecting team member	Low	High	Medium	Adapt and compromise with other remaining team members
6	Team member not showing up to a scheduled meeting on time	Medium	Low	Minor	Team members being clear with when they are able to meet.
7	Work not being saved/stored safely	Low	High	Hazardous	Constantly save work. Turn on AutoSave
8	Git branches crashed	Low	High	Catastrophic	Refer to GIT management plan