Project Plan

For 2101_Data_Analytics

Group 3

Table of Contents

1. Project Plan

1.1 Vision

For Ms Kamala and her colleagues
who want a software service that is subscription-based and displays data from different sources as well as predictions for the changes of figures in the future,
the 2101_Data_Analytics is a web application based on a multi-layered MVC
that adds values to the end-users with versatile widgets and by displaying multiple widgets on the dashboard, different types of data in both tabular forms & server-side charts, and analysis & predictions regarding those datasets so that end-users will not only be able to view the data in the past but also be able to access figures in the future and thus help them construct their business strategies, travelling plans etc.
Unlike Cambridge Analytica or Nielsen,
our product delivers server-side information to our target group without collecting and utilising users' data for commercial purposes i.e. we only save a minimal amount of user information that is required to identify users, and we are not going to use the information gained to make profits by analysing customer activities, by selling it to some third party organisations nor in any other possible way.

1.2 Team Members

Name	Contact Info	Roles
Sami Hussein	shus0013@student.monash.edu +971 50 726 3664	Product Owner Backend Developer
Shuta Gunraku	sgun0027@student.monash.edu +81 80 9287 1604	Scrum Master Project QA Frontend Developer
Bryan Lim	clim0100@student.monash.edu +60 189 020 304	Frontend Developer GIT Owner Product QA
Er Tian Ru	terr0001@student.monash.edu +60 174 473 875	Frontend Developer
Jason The	jthe0005@student.monash.edu +62 821 5796 6045	Frontend Developer
Pan Wei Hung	wpan0017@student.monash.edu +60 16 260 7232	Backend Developer

1.3 Methods of Communication

1. Slack
2. Whatsapp

1.4 Team Roles

1. **Product Owner**
   Interacts with the client to identify product specifications to be put into the product backlog. Their main role is to maintain the product backlog by deciding on its content and the priority of each product backlog item. They're responsible for informing the team what to build on each iteration.

2. **Scrum Master**
   The Scrum master's role is to "lead" the team by helping them come to a consensus in decisions and moderating the team. They also organize the sprint activities (product review, retrospective etc.) and help the product owner in managing the backlog.

3. **Quality Assurance / Tester**
   The quality assurance (QA) engineer will be responsible for the quality of the product (mainly on the quality of the code produced by the developer) and do testing on the product. So, their main responsibilities are to improve the quality of the product and prevent defects in production.

4. **GIT owner**
   The GIT owner is responsible for managing the GIT repository and handling merge requests.

5. **Development Team:**
   (Responsible for developing product functionalities and product testing. Each member is self-managing and cross-functional.)
   a. **Frontend**
      Developers that will work with the frontend of the web application (HTML, CSS, etc)

   b. **Backend**
      Developers that will work with the backend of the web application (server side APIs, database etc)

1.5 Team's process model

The process model chosen for this project is based on the agile-based scrum process model. Some differences however are that instead of holding scrum meetings and artefacts updates daily it will be biweekly, once during the tutorial and once on the weekends. Each sprint will last two weeks. Scrum also recommends the team to work together in the same location but due to the pandemic, that would be impossible and so that will also be cut from our process model.

The reason why do we choose Scrum for our project

In Scrum, every team member has clearly defined roles and responsibilities. In this way, it makes our work more organized, which means although risk will always exist, we can reduce it to some extent by having things in an organized way. Moreover, Scrum lets us embrace any changes made by the clients, which we know often happens in most software development processes. There are also daily standup meetings in Scrum, which will allow every member of the team to communicate with each

other about their progress and difficulties, and allow them to identify roadblocks experienced by some members. In this way, we are able to solve the problems.

Moreover, in Scrum, the development team can also join the meeting with the stakeholders, which may allow them to clearly understand the client's requirements and vision since they are also involved in the meeting. Other than that, the Scrum framework allows transparency where there is visibility on each team member's task and progress. Make it easier to know each other's progress, and help them in case someone has a block.

By adopting Scrum, we divide our project into sprints/iterations, which allows us to deliver value often to the client. At the end of each sprint, there will be some deliverables that we can present to the client so that they can make use of our product without having the product fully built (unlike a process model which relies on only one iteration to make a fully functional product, in this case, the client needs to wait for the product to be fully built and only by then they can use it, which is not very effective).

In each sprint, we will also have user stories that need to be implemented. By having them in small chunks, the development team can focus on the quality of each deliverable by having rigorous testing and attention to the detail of the deliverable. In each sprint review, we will also have a sprint retrospective which allows us to discuss what can be improved from the previous sprint, and what the clients do not like from the deliverables. Therefore, these all allow us to increase the quality of the deliverables.

Besides that, by building our product gradually and receiving feedback from our clients, we reduce the risk of building a product that does not meet the client's requirements. Moreover, having each sprint in a fixed duration also allows the stakeholders to predict the development cost of the features, and helps them to make a decision on which backlog item they should prioritize in the future iteration. In addition to that, these all benefits suit our needs to build an MVC web-based dashboard for data visualization and analysis.

How other process models differs from Scrum

Scrum is part of the Agile methodology. And Scrum framework allows us to embrace changes, involve the clients in the development process, allow task and progress transparency, and deliver minimum viable product (MVP) in each sprint to allow the client to make use of it.

There are many other software process models available. One of them is called the Lean software development process, which is also a part of the Agile methodology. The origin of Lean methodology was started from the industrial sector, the intention is to make more efficient production systems, while Scrum originated from the knowledge-intense environment of software development, which allows people to adapt to instant changes. Moreover, Lean is often applied on a company basis, while Scrum is usually applied on a team basis. And generally, Lean involves longer process periods and the manufacture of consistent end products. Other than that, Lean focuses on improving the quality of the product, eliminating unnecessary waste, amplifying learning, optimizing the system as a whole, and reducing production times and total costs while Scrum encourages the team to learn through experience, self organize when working with problems, and reflect on what they did well and what they did not.

Another software process model is called Crystal, which is also a part of the Agile methodology. This process model encourages the tasks to be done in a team, instead of as individuals, which is the opposite of Scrum where every member will choose which task they will work on (although the task can also be done collaboratively). Moreover, the crystal also enforces the product design, requirement documents and other documentation to be understandable and straightforward, so that everyone can understand it and work more efficiently.

In Crystal, each team member should respond as and when required. Developers and testers need to have a valid reason and right logic for their actions not to let their activity be considered a waste. And they should also be well aware of all the stages they performed to allow easy reversal and reconstruct coding. Most of these are not enforced in Scrum, and therefore we can see how they differ from each other.

1.6 Definition of Done

1) **Complete all user stories**
   The definition of done (acceptance criteria) in all the user stories should be completed.

2) **Deliver a reliable application**
   - **Pass QA**
     a) **Code has been peer-reviewed**
        Code from each member for each user story should be reviewed by another team member (usually the tester) to catch any mistakes.
     b) **Pass the code testings**
        The software should be sufficiently tested by the test frames and test cases derived.
        - Pass unit testing
        - Pass BlackBox testing
     c) **Cross-browser testing**
        Cross-browser testing should be conducted on major different browsers i.e. Chrome, Firefox, Safari and Internet Explorer.
        For each browser, make sure that all contents in the pages can be seen without any error and all buttons work appropriately.
     d) **Cross-device testing**
        Cross-device testing should be conducted on different OS i.e. Windows and Macintosh. For each device, make sure that all contents in the pages can be seen without any error and all buttons work appropriately.
        ※ c & d shall be conducted with all the combinations of browsers and devices.

1.7 How will the team allocate tasks to members:

Each member picks an item from the sprint backlog to complete

1.8 How your team will keep track of progress on your project:

- **Slack:** For communications related to tasks.
- **Trello:** Overview of tasks.
- **Git**: Version control of the files in the project directory.

Git message format :
(User Story number) - Task Name (Description of changes you've made)

1.9 How your team will store and manage your backlogs:

- Trello

1.10 How your team will keep track of time spent on project tasks:

Use the time tracking service called Toggl. (Expired)

Due to the expiration of the Toggl plan, we instead used Clockify to track the time each member spent on each task from iteration 3.

Clockify description format :
(User Story Number) Task Name. (Description of changes you've made)

2. Risk Management

2.1 Risk Register

ID	Risk description	Likelihood of risk occurring	Impact of the risk occurring	Monitoring strategy	Mitigation Plan
1	The client's requirements are undefined or not well understood	5	5	Regular meeting with clients to ensure client's requirement is met	Reduction: Make sure to involve the client in the development process. Regularly show them the program after each sprint to make sure it's heading in the right direction.
2	Project group members having conflict	3	4	Infer about any issues during the standup meetings.	Acceptance: The SCRUM master should help in resolving the conflict.
3	No in house expertise with chosen development software	4	4	Discuss expertise during project inception when choosing the software to work with.	Avoidance: Don't choose software that no team member is familiar with.

4	Lack of client availability	5	4	Make sure to keep close contact with the client to know if they're unavailable for any meetings.	Reduction: Make sure to inform the client regarding their role in agile development as an active participant and work out a meeting schedule beforehand.
5	Team members facing health issues (not available, sick, etc)	3	4	Keep close contact with team members to know if any of them would be unavailable.	Acceptance: Have another team member pick up the unavailable team member's work.
6	The client suddenly propose big/unrealistic changes	3	3	Have regular meetings with clients to see if they want to propose any changes.	Avoidance: Discuss civilly why it might be infeasible to make the change and talk them out of it.
7	Faulty communications equipment, computers, internet connection or hardware	2	5	Regular examination of the equipment.	Acceptance: Have another backup equipment/device ready
8	Locally saved file corrupted	2	2	Check local save files regularly.	Reduction: Work with Git repository to ensure that all file is saved online
9	Software libraries used were revealed to be unstable (cause security flaws, etc)	1	5	Keep up to date with the software being used	Acceptance: Have an alternative software option we can switch to.
10	Misuse of data	2	3	Ensure only using the data as what has been agreed on with the user	Acceptance: Create and follow a legal regulation for data handling and usage. Make sure the user agrees to the terms of service when using the application.
11	Team members not able to complete their tasks during a sprint	4	3	Keep in close contact with team members to see if they encounter any issues	Reduction: Discuss how everyone is doing in their tasks in the standup meetings and provide help if necessary.
12	Issues with the difference in the development environments	3	3	Find a way to minimise the impact of the difference in the development environments	Acceptance: Find a solution to solve the issue and reuse the solution when the issue occurs.

2.2 Scale of Impact

Impact	Description
5	Extreme ● May result in project failure ● Project late by more than 50% ● Could affect the ability of the team to continue functioning
4	High ● May result in a significant impact on expected features, functionality or quality ● Project late by more than 25%
3	Moderate ● Significant effects on the project are unlikely

	● Project or subsystem late by more than 10%
2	Nominal ● Does not require monitoring or review ● Project late by more than 5%
1	Minimal ● Little or no impact on any aspect of the project ● Should be reviewed quarterly

2.3 Scale of Likelihood

Probability	**Description**
5	91 – 100% or Very likely to occur
4	61 – 90% or Likely to occur
3	41 – 60% or May occur about half of the time
2	11 – 40% or Unlikely to occur
1	0 – 10% or Very unlikely to occur