Assignment 1

Project Management Plan

VERSION 1.0.0

Linuka Ekanayake, Xiao Hao Zheng, Tran Nhat Trung, Jimmy Tran, Siam Ahmed

Last updated: 22 Sep 2021

Table of Contents

Vision Statement

This program is for anyone who wants to view predictive analytics for future Olympic games, the product is a predictive tool that allows users to view possible future winning countries.

Project Information

Process Model

The process model used for this assignment is a variation of Scrum which aims to facilitate each team member's needs. This process model has three sprints/iterations of length 2 weeks, which are due on the 3rd and 17th of September and the final iteration due on the 1st of October. In this process model there is no daily Scrum meeting but is instead replaced with two to three meetings per week. Unlike Scrum, our process model only requires stand-up meetings once a week with the project manager (client). These changes make it more feasible for this project because team members have other priorities and commitments, which make it difficult to meet every day. These meetings are more effective because by pre-planning what needs to be done in an upcoming meeting, it can be efficient and productive.

Scope

The goal of this project is to provide users with an idea of what country is going to win the Olympics. This will be done through a dashboard for users, which will allow them to view the predictions of which country will win based on the number of medals. The project will show the table with a list of countries and its predicted number of winning medals. Users will be able to sort the list of countries by name and past medals. In addition, there will be a list of players of a particular country that will participate in the Olympics. Users can get access to the list of players from the table of countries by clicking on a hyperlink of a specific country. The table of players can be sorted alphabetically, by country, by event, and by previous achievements. Also, users will be able to see a graph that will display the number of medals each country has won in the past. Lastly, users will also be able to view a profile summary after clicking on a player’s name. This will be accessible through a web page coded with JavaScript and HTML.

Definition of Done

Project is "done" if the following features are implemented:

- Table with the list of the countries and predicted number of the medals
- Sorting bar that will sort the list by the country name and the past medals
- Table of the players that are representing a particular country
- Table of players should be accessed by clicking the hyperlink of specific country that is shown in the "List of Countries"
- Sorting bar that will sort the list of players alphabetically, by country, and by previous achievements
- Graph that will show how many medals each country has won in the past
- Pop up of player information after clicking player name
- Predictive algorithm that predicts which team is going to win the Olympics
- Navigation bar that should be implemented in every webpage

Checklist Items	Status
Tests on each feature have been written and are passing	
Tests that relate to using multiple features have been written and are passing	
Cross-browser testing has been done on 4 different browsers	
Hardware testing done on 4 different laptops/computers	
Code peer-reviewed	
Documentation updated	
Acceptance Criteria have been met	
All deliverables have been submitted	

Personnel/HR Management

Siam Ahmed

Contact:

- 0406 157 615
- sahm0036@student.monash.edu

Role:

- Timekeeper
- Developer

Responsibilities:

- Tasks allocated at meetings
- Ensures meetings do not go over scheduled time
- Holds team members accountable so that deliverables are consistent with time frame

Jimmy Tran

Contact:

- 0451 617 199
- jtra0039@student.monash.edu

Role:

- Quality Assurance
- Developer

Responsibilities:

- Tasks allocated at meetings
- Test all components of program after each feature is developed
- Asking questions on behalf of the team in forums and helpdesks

Tran Nhat Trung

Contact:

- 0412 894 222
- ntra0043@student.monash.edu

Role:

- Meeting Minutes Recorder
- Developer

Responsibilities:

- Tasks allocated at meetings
- Record meeting minutes
- Inform absent team members on their tasks and responsibilities

Xiao Hao Zheng

Contact:

- 0447 119 813
- xzhe0014@student.monash.edu

Role:

- Editor
- Developer

Responsibilities:

- Tasks allocated at meetings
- Checks all written work
- Ensures written work is in a presentable format before submission

Linuka Ekanayake

Contact:

- 0478174723
- leka0001@student.monash.edu

Role:

- Meeting Facilitator (Scrum Master)
- Developer

Responsibilities:

- Tasks allocated at meetings
- Ensures meetings are productive and facilitate productive conversations
- Ensures team is on track and everyone is aware of their tasks

Time and Task Tracking

Tasks will be allocated equally to all members of the team. This will be done by reading through the specifications and the marking scheme and break different tasks up equally. They will then be equally distributed between the team members. If there are any concerns or difficulties of a given task, team members can message other team members on Messenger or Discord for help. If a longer discussion is needed, team members can organise a meeting to tackle the concerns and difficulties of the task.

Meeting minutes and time will be tracked by making a Google Doc in the Google Drive after every meeting. This will be titled ‘Meeting # Minutes’. Attendees and apologies will be recorded along with an action sheet from that meeting; concerns arising from the meeting; and the schedule for the next meeting.

We will record each member’s tasks in Trello under the corresponding Sprint dashboard. This will include each task the member has been allocated; the date assigned; the estimated time of completion; the actual time taken; the difficulty of the task; and the concerns of the task.

Analysis of Alternatives

Platform:

When designing any sort of application, the platform that will be used requires a great deal of thought and consideration, especially as many applications remain in use for long periods of time. As such, the best platform to use will depend on aspects such as its overall purpose, accessibility, and cost. For example, specialised applications such as animation software, are extremely niche and may not be require the same level of portability and accessibility as applications such as weather forecast. The main three application types considered were: desktop apps, mobile apps, and web apps. Each of these types were researched and their key aspects were summarised for evaluation.

Desktop applications are programs that need to be installed onto the user's machine prior to its use. As a result, desktop applications will generally not require an active internet connection as most of their assets are stored locally when installed. Having it installed has the benefit that the app can better utilise the machine's resources, thereby improving performance. Security for desktop applications is generally better than other platforms as they have less dependency on the internet. The costs for any additional services are minimal as they only need to be distributed on websites. On the other hand, being installed means that it takes up storage on the machine, which depending on the software's purpose may be negligible or significant. In addition, the application is generally non-portable, being confined to the machine that it was installed on. In summary, desktop applications have good performance, high security, low associated costs, but low portability.

Mobile applications are like desktop apps in that they are also installed, however they are installed on mobile devices such as phones and tablets. They can fully utilise the resources of the device such as the gyroscope and accelerometer. The portability of mobile devices means that the app is portable which can be extremely useful – a good example being a GPS app which allows the user to navigate in real time. Mobile applications can be distributed by their operating system's respective app stores – App Store for Apple, Google Play Store for Android – which have relatively low costs of distribution. However, mobile apps are not without disadvantages. The processing power for mobile applications is limited by the hardware of their devices, which in cases where intensive processing is required, may be unsuitable. Additionally, as there are multiple operating systems for mobile devices, iOS and Android being the foremost, the app will need to be optimised for each OS – this requires an increased amount of work to develop and maintain. Thus, mobile applications are as portable as the devices they are on; have decent performance but generally inferior to desktop apps; minimal operating costs but require increased effort in development and maintenance.

Web applications are accessed via an internet browser such as Google Chrome, Microsoft Edge, or Opera. This makes it fundamentally different from desktop and mobile applications which have them installed. This aspect of being accessed on the internet means they are extremely portable, able to be accessed on any device (desktop, laptop, mobile or tablet), and have low system requirements. They also do not require the user to continually update the app, as is the case with desktop and mobile apps, making it more user-friendly. However, this strength is only due to the application being hosted on a web server, which requires an internet connection. The application can only be accessed with a stable internet connection, which may not always be possible in more remote places in the world. The web server also has hosting costs and depending on the app's usage can be more than a desktop app's distribution website. Therefore, mobile applications have great portability and accessibility across devices; low system requirements; and require no installation by the user. However, they require a stable internet connection; suffer decreased performance for complex processes; and are less secure.

Considering the strengths and weaknesses of the platform types, an evaluation was made based on the suitability for the purpose and functionality of our project. Firstly, the predictive Olympics application being developed does not require a great deal of processing power, as it mostly concerns data manipulation and some non-intensive calculations – all three types are suitable in terms of performance. Secondly, the portability may improve the user experience, but is not essential – this leans the preference to either mobile or web as both allow for portability but does not discount desktop as it is not vital to the app's success. Security is not a concern for the application as it deals with public data and non-sensitive information, hence it is not required that desktop app has a high level of security - web and mobile would still be appropriate. An internet connection would be needed to load the data initially, however after that it should be able to function without it. As for costs, the most appropriate would be either desktop or web application as both would have low operating costs for a website and require less effort to maintain and develop compared to mobile.

Ultimately, the final decision for a platform was a web application. Since it is on the internet, it would increase accessibility to its users as they would not be required to install anything, thereby streamlining the process of using it. This means a wider range of people will be able to access it regardless of if they are on desktop or mobile. Moreover, due to the low requirements for the Olympics application, it will run smoothly. This benefits users who do not have access to hardware with good performance such as a desktop. Despite requiring internet access to load initially, it was determined that this drawback is negligible as its use is unlikely to be necessary in any remote area without internet access. The combined cost of maintenance and operation is the lowest for web apps since the server hosting the website will be relatively inexpensive for simple processes. The maintenance for a web application is less effort than a mobile app, especially if it is intended to be available on both iOS and Android. Therefore, while the other two options are not entirely unfeasible, a web application serves as the best platform for which the Olympics application will be developed.

Programming Language

One of the most important architectural decisions that needs to be made is the programming language(s) that will be used. With a plethora of programming languages available to programmers, we decided to filter it based on which languages we already knew. As such, we reduced to a few options; JavaScript/HTML, Python and Java. In order to evaluate the plausibility of each option, we considered the following selection criteria: type of application, programmer experience, ecosystem of libraries, and performance.

In terms of type of application, our most likely candidate is a website/web application. JavaScript and HTML are the most used programming languages if you want to create a website/web application and are the languages that web browsers use. This makes them both extremely feasible options to better suit the needs of our type of application. Python can be used for web development too, with its main use being server-side web applications. However, Python is not the language that web browsers run in so its feasibility for building websites is not as high. Similar to Python, Java is also a commonly used language for web development to make server-side applications, allowing us to create dynamic web pages for instance. However, since we will not be making server-side applications, its feasibility and usefulness is low.

In terms of programmer experience, this is an extremely important factor because there is not only a cost associated with learning a new language, but there is also the familiarity and experience with the programming language's syntax, data types, inbuilt functions etc. The language that all members were extremely familiar and comfortable with was Python. This means there would be a very low cost to use Python and all members would be extremely familiar with all of Python's inbuilt functions, syntax, and data types etc. This makes Python the most feasible option in terms of programmer experience. Most members of our team have used JavaScript/HTML before, with a few never having used it before. This means there would be a medium cost to using JS/HTML, since those members would need to learn it for the first time. Furthermore, most members who have used JS/HTML before, have not used it for a while and are not the most confident in their JS skills. However, this is a smaller cost as it would be relatively easy to relearn and become familiar with the syntax and inbuilt functions etc. Overall, the cost would be medium low and so it is still a relatively feasible option. Finally, Java was the language very few of us knew so there would be a very high cost to using it, making it extremely unfeasible.

In terms of ecosystems of libraries, this is dependent on what features our project would have as it would determine the types of libraries required. One of the features of our project is the predictive analysis algorithm, this means a language with more data science libraries would be better suited. Python is one of the most popular languages for data science, as it offers a massive catalogue of libraries for data science. This makes it an extremely feasible option for our project as there would be data processing involved. JavaScript does not offer many data science packages, so it's feasibility for data processing is low. Java offers many data science functions such as data analysis, data processing and statistical analysis etc. This makes Java a feasible option for our data processing too.

In terms of performance, this would directly impact the user experience as it would help reduce loading time. JavaScript is a very fast programming language as it has advanced multithreaded capabilities. Comparatively speaking, Python is slower because it processes requests in a single flow. Java has the fastest performance out of all the options since its bytecode is similar to native code. However, this depends on how optimally it's given tasks are managed by the host Java virtual machine. In the end, all three languages have very high performance, so they are all feasible options.

With our criteria of selection being type of application; programmer experience; ecosystem of libraries; and performance, there were many factors for us to consider for each of our options. Since the most likely candidate for our platform is a website, JavaScript/HTML is a requirement. Despite its shortcomings of being a programming language that some members are not familiar with, its cost to learn is not that high since very few of our members would need to learn it for the first time. However, it does have the disadvantage of not being well suited for data processing. Python has a massive library of data science packages making it the best choice for data processing. Python's biggest disadvantage is that it is not used by browsers, making it not suitable for creating websites. In order to overcome the shortcomings of JavaScript and Python, both programming languages can be used. That way the data processing can be done efficiently and effectively using Python and the data can be presented to the user on the website using JavaScript. Java is the most unfeasible option as most of our team members are not familiar with it, meaning there is a high cost to using that language. Furthermore, Java is not used by websites and the data science features are already available on Python. Therefore, the most feasible options for the programming languages for this project is a combination of both JavaScript/HTML and Python.

Risk Management

Risk ID	Date raised	Risk description	Likelihood of Risk	Impact of Risk	Risk Mitigation
1	26/08/21	Unfamiliarity with predictive algorithms	**High**	**High**	Communicate frequently with all team members and have a meeting to understand how predictive algorithms work
2	26/08/21	Illnesses and absences	**Medium**	**Low**	Make sure all members understand everyone's role in the project so that they can fill in if necessary
3	26/08/21	Difficulty finding appropriate data for the project	**Medium**	**High**	Make sure to thoroughly research the project topic
4	26/08/21	Code bugs	**High**	**High**	Members should know/learn how to debug code
5	26/08/21	Unfamiliarity with programming languages and syntax	**High**	**Medium**	Members should ask other members if unsure how to implement code or otherwise refer to online sources
6	26/08/21	Unavailability of product facilitator	**Low**	**High**	Make the most out of the meetings/workshops

					we have with the product facilitator when they are available by coming prepared
7	26/08/21	Crashes while using GitLab	**Low**	**High**	Ensure members have the most up-to-date versions of the repository to accommodate for GitLab problems
8	27/08/21	Poorly defined project prototype	**Low**	**High**	Create project management plan and well-defined user stories to make sure prototype is representative of final product
9	15/09/21	Inability to obtain detailed information for the player popup	**High**	**High**	Alter the player information popup that contains whatever information we have available, given it is still a valuable feature for the customer.
10	15/09/21	Less familiarity with the R coding language	**High**	**Medium**	Determining other ways to implement graph page e.g., Google charts
11	15/09/21	Not being able to deal with time constraints	**Low**	**High**	Properly planning sprints and allocating tasks to team members. Constantly keeping the team updated and asking other members for help when required.

12	15/09/21	Challenges with sorting	**Low**	**Low**	Determine most appropriate way to implement sorting for different object attributes
13	15/09/21	Creating a popup	**Low**	**Low**	Compromise and use a different method. E.g., hyperlink
14	15/09/21	Finding daily medals won data	**High**	**High**	Graph some alternate data with what we have