



Trinity College Dublin

Coláiste na Tríonóide, Baile Átha Cliath

The University of Dublin

CSU33012 Software Engineering-Measuring Software Engineering

**Pradyumn Bhardwaj, Student Id
19321492**

January 3, 2022

Contents

1. Introduction

- **What is Software Engineering?**

2. Task

3. Different ways to measure software engineering activity

- 3.1 Number of commits and volume of code**
- 3.2 Code coverage and unit testing**
- 3.3 Health and productivity**
- 3.4 Bug in a program**

4. Platforms available

- 4.1 GitClear**
- 4.2 Code Climate**
- 4.3 Hackystat**
- 4.4 Other Programs**

5. Algorithmic Approach

6. Ethical Concerns

7. Acknowledgement

8. Summary

9. Reflection

10. References

1. Introduction

The focus of this assignment was to learn about how to measure an engineering activity. In this report, I will analyze different ways of collecting data sets and using them to measure different software engineering methods various advantages of these measurements.

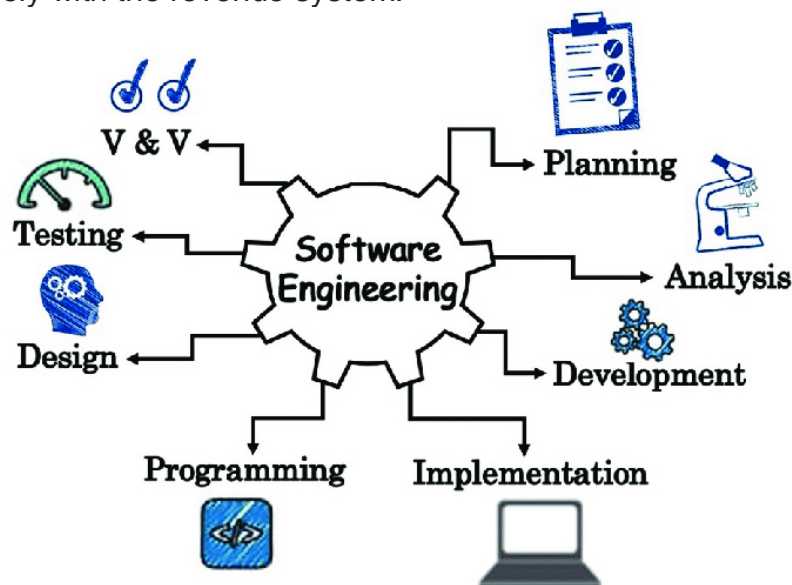
There are many different computational approaches which can be used for these measurements. In the end, I will conclude by outlining some ethical concerns which should not be overlooked and be taken care of.

What is Software Engineering?

Software engineering is defined as a process of collecting data from user such as the requirements and analyzing them and then use this data to look for the solution to the problem and then design and build a software application which will fulfil all the requirements of the user.

The field of software engineering is one of the areas of key importance in today's world. Most of the major companies are working in this sector and making the lives of people better by solving problems. They in turn also make a huge profit. Some of the major companies include Google, Meta, Amazon, Arm etc.

Most of the organisations now a days want to make huge profits. Therefore, many metrics are being developed by them so that they can measure the productivity of their engineers. But there is one problem. Most of the metrics which are being used have many flaws in them. They do not relate closely with the revenue system.



2. Task

The task of this assignment was to do proper research and assessment and form an opinion on whether it is reasonable to perform various kinds of analysis on the performance of software engineers as they go about their work or whether some of this crosses a line.

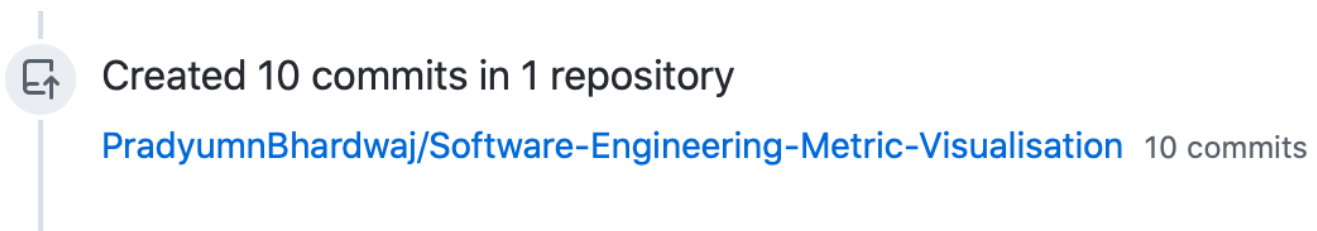
3. Different ways to measure software engineering activity

There are many ways in which one can measure the software engineering activity. But the question is which of the ways are most efficient. This section will include most common ways of measuring the activity and discuss both their benefits and drawbacks.

3.1 Number of Commits and volume of code

One of the most common ways to see how the engineer is working is through the number of commits an engineer makes. This method of measuring the productivity is being used from a long time. There are many softwares on which we can see the number of codes committed, number of lines added etc. But this method of measuring is very basic and does not properly fulfill our needs.

By seeing just the number of commits we cannot see whether the task is completed by the engineer or not. Just doing a lot of commits does not tell that the engineer is productive or not. However, when the work is being given to a team, we can see which engineer is giving the most output by this method.



*Example of number of commits

Also, the amount of code written is not proportional to the quality of code written. A high quality of code is the one which performs the same functionalities in less amount of code. For example, if an engineer writes a code in 100 lines and the other engineer writes the code in 40 lines and the functionality of both the code is same, the code of the second engineer will be highly appreciated because it will run in less time and will also be very much cost efficient.

If the volume of code is taken into count as a measure of productivity, then the engineers might add unnecessary lines to the code for inorganically increasing the volume of the code which won't help in long run.

3.2 Code coverage and unit testing

One another way of measuring software engineering is using unit testing and code coverage. In my opinion, it is a much more effective way than seeing the code being committed or the number of lines added. Code coverage means the amount of your code which being actually used by the unit tests. It can help to assess whether the code written by the engineer is useful or not.

Many a times engineers write a huge amount of code but when you test them against the unit cases, it fails and there is very little coverage so whole of the code and time gets wasted. It also makes sure that the code written is bug free. But this also has some minor problems. The test cases may pass a certain code which may be unnecessary in the future of the software.

```

package com.zereturnaround.blog;

public class BasketWeightCalculator {
    private int totalWeight = 0;

    public void addItem(int itemWeight) // Assume weight is always an integer
                                        // number
    {
        totalWeight = totalWeight + itemWeight;

        /* Feature request BWC-356. Weight can * never be negative */
        All 2 branches covered, right < 0) {
            totalWeight = 0;
        }
    }

    public int getTotalWeight() {
        return totalWeight;
    }
}

```

Element	Coverage	Covered Instructio...	Missed Instr
com.zereturnaround.blog	100,0 %	22	
BasketWeightCalculator.java	100,0 %	22	

*Example of a code coverage of 100%

3.3 Health and Productivity

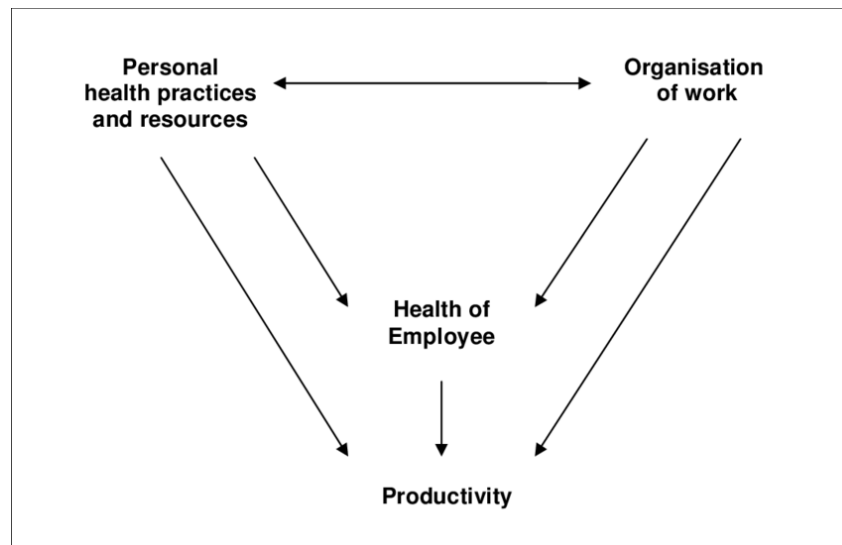
One of the key aspects which is overlooked by many people while measuring software engineering is the health of the engineer. Many a times a software engineer experiences a lot of workloads which directly affects his mental health.

And in turn they are not that productive and efficient and tend to make minor mistakes which they don't make otherwise. Keeping record of the health activities can be of a great benefit. It can tell at what particular time of a month or year an engineer is more or less effective.

Studying all the health-related data will help to make a more balanced life which in return lead to more productivity. There are many health apps which can be downloaded free of cost on both ios and android. These apps help to keep a track record of an engineer.

There are also many devices made for the same purpose, but they are especially designed for this work. Fit-Bit is one of the best examples. It is specially made to keep a track of fitness of an individual. These apps and devices remind the engineer of when to drink water, eat food, take a break from work, stand, breath deeply and sleep which all in turn provides better productivity. A fit bit is shown in the picture below.





3.4 Bug in a program

Whenever an engineer writes a code, he tries his best that the code does not have any bugs. But having bugs in a program is completely fine. No engineer should be scared of having bugs in his or her program. But the seriousness of these bugs should not be overlooked. It is obvious that the time taken to solve these bugs will be directly proportional to the experience of the engineer.

4. Platforms available

We have seen how a company profits by collecting and analysing the data which they gather from the engineers and increase their productivity. Nowadays, there are many platforms available where we can use many different metrics available and can make a reasonable opinion on whether the engineer is productive or not. In this section, I am going to highlight the most useful tools which I found in my research and are not like the old school platforms.

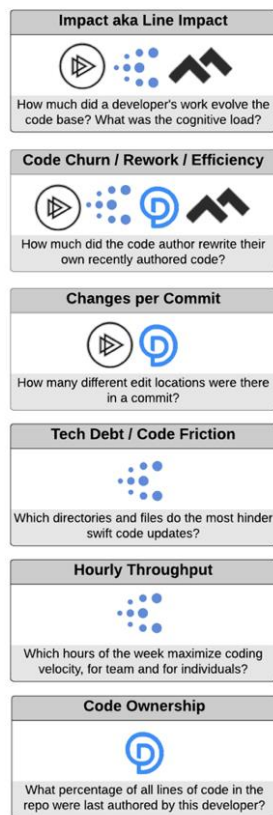
4.1 GitClear

GitClear was founded in year 2017 with the aim to become the best solution for technical managers to review the performance of their employees. Till now, gitclear has built many code reviewing tools which includes a graphical Directory Browser that pinpoints tech debt. GitClear focuses more on quality of the source data.

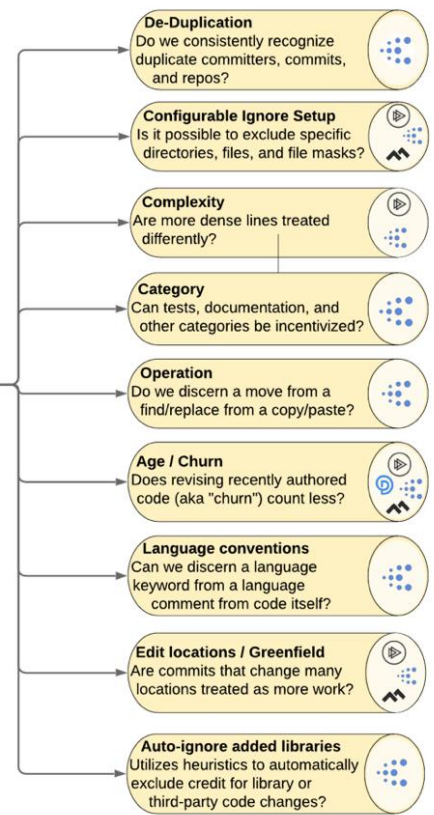
Most of the tech metric providers use the lines of code written or the number of commits made as the top basis of their analysis. In the above section, I have discussed both the benefits and disadvantages of using these as a criterion of measurement. In 2019, GitClear spent a lot of time and resources to prove that the data quality matters a lot.

The GitClear has succeeded to an extent to create a single reliable metric which the managers can use to capture the code activity. Using this, a manager can see the activity of each developer. The team at GitClear has worked a lot to refine the lines of code as much as they can into a consistent and reliable data source.

Metrics Featured in Products



Data Quality Factors



Pluralsight / GitPrime



GitClear



Pinpoint



Code Climate Velocity

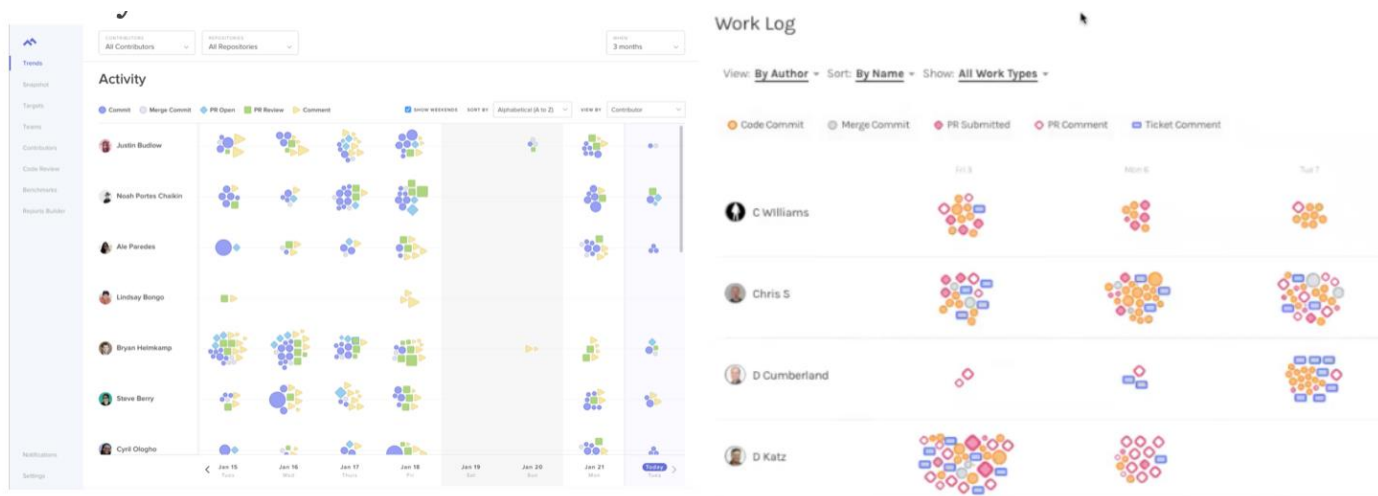
4.2 Code Climate

Code climate is a new platform which came into the market not long ago. It has a tool called climate which ensures that the code of best quality is being submitted by the engineer. It provides an extensible platform. It analyses thousands of codes each day and maintains to keep a good code health.

It has made lives of many software engineers very simple. When a developer writes code, there may be very small issues which he might not keep record of and proceeds to create a pull request for the code.

Using code climate quality, the software analyses each of the of the pull requests and traps the issues. Only after this is done, the code is integrated. Obviously, more the issues, the less productive the developer. So, it keeps track of which lines of code are being set for pull requests by which engineer. Thus, the manager can see which engineer is giving code with a smaller number of issues.

The very new tool introduced by code climate is the tool called cold climate velocity. Velocity takes all the data from pull requests and commits and then converts it to a useable data. This data then can be using for improvements which will later help with increasing productivity. It uses then data form DevOps tools like Jira and translates the data.



*Difference between Code Climate and GitClear

4.3 Hackstat















Hackstat is an open-source framework for collection analysis, visualisation, interpretation, annotation and dissemination of software development process and product data. It is a very advanced and helpful tool which is used by many of the researchers for software engineering experimentation, metrics visualisation.

The functioning of Hackstat is very different from other platforms. It uses a sensor which collects the raw data and sends it to a web service called the Hackstat SensorBase where it is stored. The engineer has to attach these sensors to their development tools. Hackstat is very portable. This means that we can easily integrate it to the code repositories on GitHub and other such platforms.

It collects the data in such a way that the user may not be even aware if his data is being sent to a server for analysis or not. The data collected by Hackstat includes the amount of time that is being spent, the amount of work to be done, code coverage, the number of commits and lines written and the complexity of the code.

But the working of this tool is not appreciated by most of the engineers as it gives their in-depth data to a server without their knowledge. The data is then collected by their managers which know exactly what each of the engineer in his team is doing. This increases the productivity of his team and in turn generates much more revenue.

But the working of this tool is said to be unethical which therefore has prevented the tool from being expanded more at the industry level.

Coverage		Complexity	
	 85.0		 2.4
	 81.0		 3.0
	 4.0		 2.9
N/A			 3.1
N/A		N/A	
N/A		N/A	

*Hackystat working

4.4 Other platforms

There are many other platforms which can be used for analyzing and measuring the software engineers. Some of the common platforms include Github stats, Flow, Uplevel, Waydev but most of these are still in their developing stage and henceforth we cannot rely on them for the data interpretation because misleading data may hinder in the working of the engineers which in turn give very less productivity.



5. Algorithmic Approach

There are many algorithmic approaches available to us which we can use to measure software engineering. Algorithmic approaches have very more advantages over normal approaches. When data is entered manually, there are many chances that some important data is lost or the person entering the data missed some crucial information while entering.

This in turn may provide inaccurate report which will instead of boosting the productivity of the engineer, put him under pressure and may result in deteriorating mental health. Algorithmic processes are much more fast, efficient, and descriptive. In the past few years, one of the information technology sectors is mostly in news. It is the artificial intelligence and machine learning.

Artificial intelligence refers to the intelligence demonstrated by the computers against the natural intelligence of humans. In this section, I will show how we can use machine learning approaches to measure software engineering and increase the productivity of the developers which will result in higher profits in less expenses.

5.1 Machine learning

Machine learning in my opinion is one of the best approaches. It is a branch of computer science which focuses on the way the algorithms and data is being interpreted in the same way the humans do using artificial intelligence.

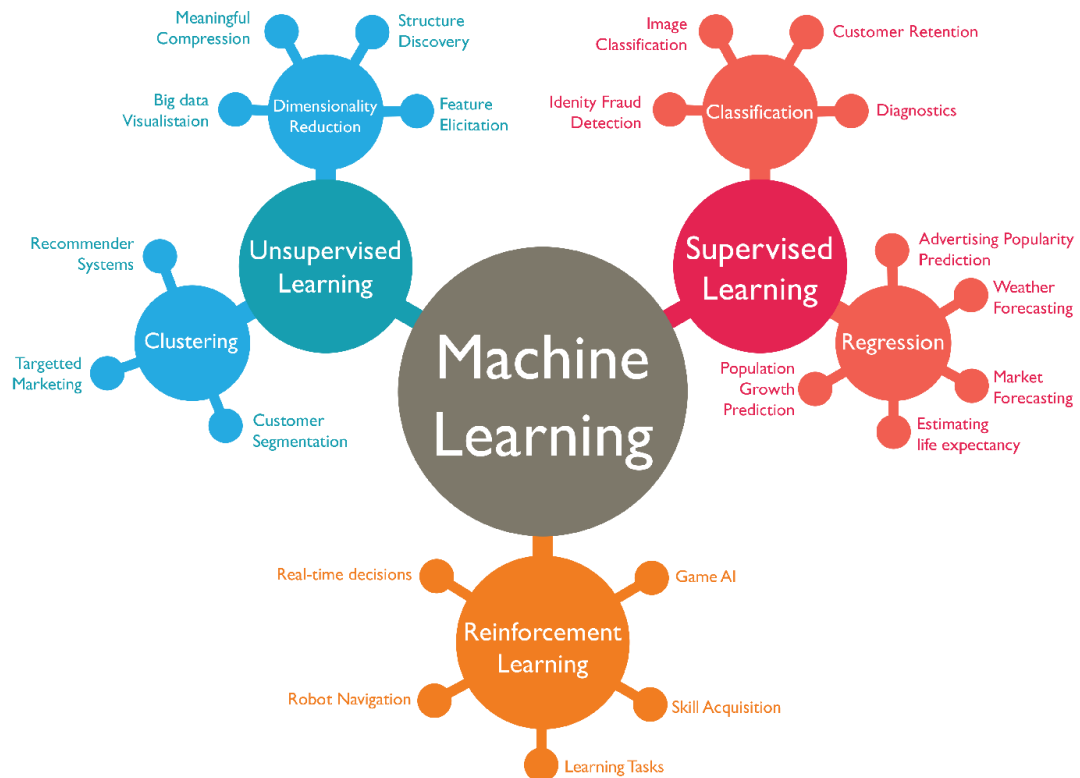
Machine learning is one of the most common branches of computer science and in the last few years, there has been a massive development in this field. Nowadays, self-driven cars are very common because of Tesla. They can achieve this only because of machine learning. To use machine learning, one must have very large data sets to analyze.

We have already seen that an experienced and better engineer is the one who writes fast and complex code. We can use machine learning in this case. We can use artificial intelligence to see which code is faster and performs the same functionality as the other engineer in less amount of time. Complex and rigorous testing will ensure that the best code is only chosen.

Machine learning is the future of data analytics with the world's most known and famous companies such as Google, Meta, Amazon, Tesla and many more investing huge sums of money in developing the machine learning algorithms.

There are mainly 3 main methods in machine learning:

- Supervised learning- Common algorithms include:
 1. K-nearest neighbors
 2. Linear Discriminant analysis
- Unsupervised learning- Common algorithms include:
 1. K means clustering
 2. Principal component analysis
- Reinforcement learning



6. Ethical Concerns

There are a lot of ethical concerns which come to our mind when we talk of measuring software engineering. Though the methods and practices which we have discussed above help in increasing the productivity of the company, it can have some serious ethical concerns with them which need to be taken care of.

When we apply these methods to measure the software engineering, the engineers might think that they are always being monitored even if they are not. This will always keep them in some kind of pressure which will definitely show in their work. To do the work on time and meet their deadlines, they might not give their hundred percent in delivering the product to the customer which will in turn be bad for the reputation of the company as a whole.

It completely defeats the whole purpose of the measurement. This will create a very unhealthy working environment within the organization. For complete measurement of a software engineer, we cannot use only one matrix. There should be different metrics and the result should be given by looking at all of them instead of one.

For example, let's say one company uses the volume of code written as a measure of productivity. Now one developer writes a code in a very concise manner and very less volume and other developer writes a code in very big volume. Both codes have the same functionality. In this case, the company will see the second developer as a more productive developer because he has written more amount of code than the first developer. We can see that the first developer is obviously a better developer because less lines means the code is cost effective and less amount of money will be spent by the company than is case of the second developer.

In the end of the day, the best use for this is to not make employment decisions but rather to make recourse decisions.

Another major concern doing this is the privacy concern. If the managers uses the sensors which they connect directly to the IDE of the developer like in the case of Hackystat, the developer will not even know that his data is being collected by his managers. There might be something private which he is doing that may be seen by his manager.

The manager must ensure that the data which is being collected is only that data which will help in decision making, growing business and generating more revenue. Confidentiality, integrity and honesty plays a key role in the performance of a software engineer. There must be strict line between the data which being collected and what that data is being used for.

There must also be complete transparency in data collection. Each and every employee must be informed if their data is being collected or not. And if it is being collected, whether the data is with the firm itself or it is being given to a third party for analysis. If there is improper interpretation of data, it can give very negative effects to the company.

7. Acknowledgement

I would like to thank my professor for helping me throughout the report. He cleared even the smallest or the silliest doubts that I had. The lecture videos provided me most of the information which was required to make this report. Other than that, I looked many videos on the internet and read a lot of documents to look for the required knowledge for the report. The face-to-face sessions were of a great benefit to me. Tutorials also helped me a lot where I listened to other people's doubts. Many a times I also had the same doubts but was nervous to ask them.

8. Summary

Here is a small summary of what all is covered in this report: -

- Measuring software engineering is not as simple as it looks. There are many technicalities which are involved in this. In my opinion, software engineering should be measured but not in a way that is violates someone's privacy. Measuring software engineering if done in a right manner can be very much beneficial for both the organization and the for the developer itself. It can generate huge revenues for the company.
- The first section included how one can measure engineering activity. there are a number of ways to measure this I have used number of commits and volume of code, user testing and coverage, health and productivity and bugs in a program.
- In the second section, I have shown the different platforms which one can use To gather data and perform calculations over these data sets. The important infrastructure here is the emergence of utility computing with supports the gathering of large volumes of data and the processing of this data via algorithms this infrastructure enables complex and computationally expensive calculations to be performed.
- In the third section, I have shown how algorithmic approaches can be used to solve the problems we face while measuring software engineering. I have used the machine learning algorithms and how they can help achieve our goals.

- Finally, I have raised some of the ethical concerns which should be taken care of and no matter what, should not be overlooked. We should be concerned with the ethics and legal or moral issues surrounding the processing of this kind of personal data.

9. Reflection

In the end, I am quite happy about how my assignment turned out. Last assignment helped me a lot to improve this assignment. Time management is one of the most important things which is required to do any of the assignment. I spent a lot of time and research while doing this assignment. I think I must improve my time management skills. Weekly tutorials helped me a lot in putting my features in this assignment. This assignment made me learn the value of time management. In my next assignments I will keep in mind how much time should I spend in doing the assignments. In the starting of the assignment, I started working without the proper knowledge which led to wastage of a lot of time. This assignment made me learn a lot of new things about how the software industry works and what all happens inside a company. Other than that, I learned a lot about how managers assess their engineers and give them work on what basis.

10. References

- <https://www.guru99.com/what-is-software-engineering.html>
- <https://www.getclockwise.com/blog/measure-productivity-development>
- https://www.gitclear.com/measuring_code_activity_a_comprehensive_guide_for_the_data_driven#can_developer_productivity_be_measured
- <https://hackystat.github.io>
- <https://www.scitepress.org/Papers/2017/63271/63271.pdf>
- <https://techbeacon.com/app-dev-testing/should-you-use-ai-make-decisions-about-your-software-team>
- <https://www.infopulse.com/blog/top-10-software-development-metrics-to-measure-productivity/>
- <https://www.3pillarglobal.com/insights/critical-metrics-for-measuring-software-development-team-performance/>
- <https://harness.io/blog/developer-productivity/>