**ORCHID INTERNATIONAL COLLEGE**

**(Faculty of Management)**

**Lab #1 PERT Chart**

**Date: 8th July 2019**

**Submitted By Submitted To**

**Name: Safal Lamichhane Department of BIM**

**TU Roll No: 6685/15**

**PERT (Program Evaluation and Review Technique) Chart**

A PERT chart is a project management tool that provides a graphical representation of a project’s timeline. The PERT are considered more preferable than Gantt Chat because they identify task dependencies but they are often more difficult to interpret.

Pert Chart uses 3 estimates for an activity duration. They are:

**Most likely (M):**

This estimate is based on the duration of the activity, given the resources likely to be assigned, their productivity, realistic expectations of availability for the activity, dependencies on other participants, and interruptions.

**Optimistic (O):** The activity duration based on analysis of the best-case scenario for the activity.

**Pessimistic (P):** The activity duration based on analysis of the worst-case scenario for the activity

**Items of pert chart:**

1. ES: Early Start Time

2. LS: Late Start Time

3. EF: Early Finish Time

4. LF: Late Finish Time

**Formula:**

1. EFi=ESi+Di

2. EFj=ESi

3. LSi=LFi-Di

4. LFi=LSj

Q. Prepare a PERT chart for given schedule:

|  |  |  |
| --- | --- | --- |
| Activity | Predecessor | Duration(Days) |
| A | - | 3 |
| B | A | 4 |
| C | A | 2 |
| D | B | 5 |
| E | C | 1 |
| F | C | 2 |
| G | D,E | 4 |
| H | F,G | 3 |

Table 1.1 Pert chart

Where:

A : Requirement Collection and Analysis

B : User Interface Design

C : Frontend Designing Task

D : Database Design

E : Backend Coding

F : Code Testing

G : Connection

H : Maintenance



**Fig 1.1 Pert Chart**

**ORCHID INTERNATIONAL COLLEGE**

**(Faculty of Management)**

**Lab #2 Gantt chart**

**Date: 10th July 2019**

**Submitted By Submitted To**

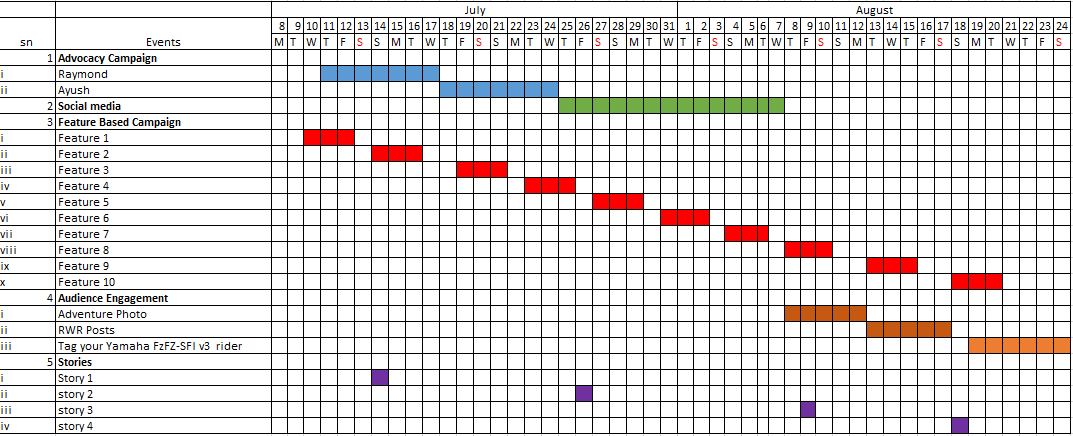
**Name: Safal Lamichhane Department of BIM**

**TU Roll No: 6685/15**

**Gantt chart**

A Gantt chart is a type of bar chart that illustrates a project schedule. This chart lists the tasks to be performed on the vertical axis and time intervals in horizontal axis. The width of the horizontal bars in the graph shows the duration of each activity. Gantt chart illustrate the start and finish dates of the terminal elements and summary elements of a project.

The advantages of using a Gantt chart is that it helps you monitor the progress of your project and to set priorities. The project manager and the team members can easily see what needs to be done at what time and if there are any delays the chart can easily be altered to balance it. This is important to remember, a Gantt chart is not carved in stone. Having a perfect chart does not guarantee a successful project execution on deadline, it is also crucial to compare the projected progress to the real progress and then update the chart accordingly.

****

**Fig 2.1: Gantt chart for new Project**

A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity. This allows you to see at a glance:

**ORCHID INTERNATIONAL COLLEGE**

**(Faculty of Management)**

**Lab #3 IEEE SRS**

**Date: 13th July 2019**

**Submitted By Submitted To**

**Name: Safal Lamichhane Department of BIM**

**TU Roll No: 6685/15**

**Software Requirement Specification**

A software requirements specification (SRS) is a document that describes what the software will do and how it will be expected to perform. An SRS describes the functionality the product needs to fulfill all stakeholders (business, users) needs.

A typical SRS includes:

* A purpose
* An overall description
* Specific requirements

The best SRS documents define how the software will interact when embedded in hardware or when connected to other software. Good SRS documents also account for real-life users.

Software Requirements Specification

For

Game Information System

Version 1.0 approved

Prepared by Safal Lamichhane

<OIC>

13th July, 2019

**Chapter-I** **Introduction**

**1.1 Purpose:**

The project “Game Information System” provides a detail or information about the game named “DOTA2”. As the game is very much difficult to learn and the new players have some problems understanding the game this system was created. The system provides guide for the Game heroes, items that are used with proper description about the game items according to the type of hero played.

The various purposes of developing Game Information System are:

* To help the guidance for the game easier and automated
* To make the existing manual system for guiding new players into computerized system
* To help the users find out the details of the heroes
* To add the heroes and items by the admin

**1.2 Intended Audience:**

This Software Requirements Specification document is intended for software engineers, system testers and software designers in developing, testing, and for the project. It is suggested to read the sections sequentially, and to reference the appendices as one progresses, in order to clarify jargon terms and definitions with finding out what are the requirements that should be maintained in the system.

**1.3 Scope of the Project:**

The proposed system will manage the recording of heroes and items details. The system is not able to keep the track of players who is playing certain heroes regularly. It only provides some basic extent guide and the users have to implement and play the game in order to fully understand what the guide is trying to say. The actual demo of the game is not provided by this system.

**1.4 References:**

Dota2. (1999, 1 16). Retrieved from Dota2Valve: <http://dota2.com/international>

DrawIO. (n.d.). Retrieved July 7, 2019, from DrawIO: <https://draw.io/>

How Can I Check Database. (2019, July 9) Retrieved from Stack Overflow:

<https://stackoverflow.com/questions/37373038/how-can-i-check-database>

Steam Community. (2007, 12 23). Retrieved from Steam https://steamcommunity.com/

**1.5 Abbreviations**

Admin: Administration

BIM Bachelor of Information Management

DOTA Defence of the Ancients

OIC Orchid International College

SP Summer Project

TU Tribhuvan University

**Chapter-II** **Overall Description**

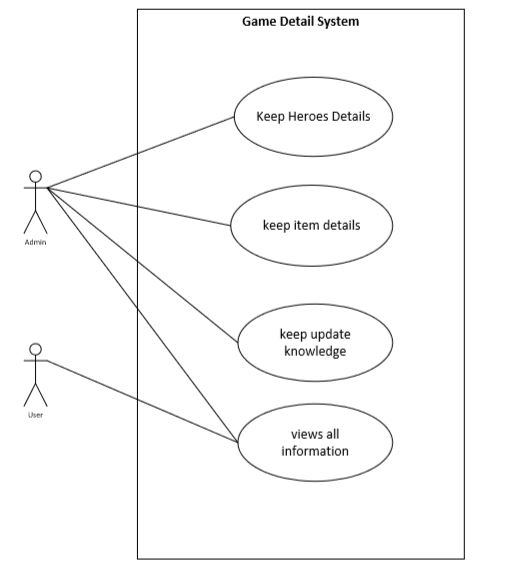
**2.1 Product Perspective:**

Game Information System is an Android application that provides the details of the heroes and items used by that heroes in game called DOTA. There are two actors: Admin who will upload the heroes and items and User who will view that information.

**2.2 Product Functions:**

The product function can be shown by its Use case diagram which has been shown.

Use Case of School Game Information System.



Use case description of UC-1

|  |  |
| --- | --- |
| Use Case Identifier | Uc1-Keep Heroes detail |
| Primary Actor | Admin |
| Secondary actor | NONE |
| Description: | The Detail of the heroes are stored and seen  like name of the hero ,type of the hero, complexity can be displayed |
| Pre-condition | The heroes Detail should not be stored already |
| Post-condition | The database must be updated |
| Success Scenario | The Heroes Information can be Seen |
| Failure Scenario | The Heroes details aren’t stored properly or there mayn’t be that hero |

Use case description of UC-2

|  |  |
| --- | --- |
| Use Case Identifier | Uc2-Keep item details |
| Primary Actor | Admin |
| Secondary actor | NONE |
| Description: | The detail of the item is recorded like item name and the type of the item whether it should be manually used or passive in nature |
| Pre-condition | The item detail should not be stored already and the user cant insert the item information |
| Post-condition | The database must be updated |
| Success Scenario | The details information can be seen |
| Failure Scenario | The item details are not stored properly or there mayn’t be such item |

Use case description of UC-3

|  |  |
| --- | --- |
| Use Case Identifier | Uc3-keep update knowledge |
| Primary Actor | Admin |
| Secondary actor | NONE |
| Description: | The update of the games are commonly occurred and the admin should be able to update the information or patches properly |
| Pre-condition | The update must have some changes in the game elements like heroes power, new items and so on |
| Post-condition | The database must be updated |
| Success Scenario | The updated Information can be clearly seen |
| Failure Scenario | The updated information is not properly shown and recorded properly |

Use case description of UC-4

|  |  |
| --- | --- |
| Use Case Identifier | Uc4-view information |
| Primary Actor | User |
| Secondary actor | Admin |
| Description: | All the information about heroes, items can be viewed by the user but is not able to make any changes on the contents. |
| Pre-condition | There must be information available for the user to view |
| Post-condition | The user must obtain the required information |
| Success Scenario | The user can see the particular needed information |
| Failure Scenario | The information is not properly shown or the information searched by user is not available |

**2.3 User Characteristics:**

a) The user is a new player who don’t have any knowledge about the items and heroes to be played according to the situations

b) The user should know general knowledge about the heroes and items that exists in the Dota value.

**2.4 Principal Actors:**

The principal actors of Game Information System are: Administrators, Teachers and System itself.

**2.5 General Constraints:**

a) Game Information System is internet connectivity application.

b) For this system to work fully, we must have interested gamers as users.

**Chapter-III**

**Requirements and Conclusion**

**3.1 Non-Functional Requirements:**

a) Security:

Game Information System should be secure enough. It is an application, which needs to be secured enough. The files regarding data of heroes should be made secured.

b) Fault Tolerance:

Game Information System should be fault tolerant. Data should not be corrupted in case of system crash or power failure.

d) Platform dependency:

Game Information System is not should not be dependent on single hardware as well as software platforms.

e) Accuracy:

Game Information System should be accurate. This system should perform its task accurately as far as practicable.

**3.2 Feasibility Analysis:**

The analysis of feasibility has conclude that the project is feasible with respect to time and cost. The technology used to develop are almost Open Source, therefore less cost for implementation and maintenance will be involved.

Game Information System is economically feasible to develop as it is a simple application which don’t require security and other things

Game Information System can be implemented in various game guides which are presently not generally available but is feasible

Operational feasibility is all about how well Game Information System will solve the problem of the new Gamers which is the major tasks achieved properly.

**3.3 Conclusion:**

Summer project has been a great opportunity through which various professional knowledge of developing software in real life scenario has been developed. This project gave a lot of knowledge about how to build a good desktop based application using java swing. In the course of doing this project, a lots of difficulties were experienced but all these difficulties were removed by getting help from all the stakeholders. Also, it is uniquely developed system. It contains unique design. This project has been very helpful to demonstrate how software is developed in the software industry. This system has been developed as a major project to get degree of BIM from Faculty of Management of TU.

To conclude, the project has been able to develop the adding and viewing of the entities in the database. It has been able to give a unique username and password to the main admin who can only add information’s and complete authority of the app is on his/her hands after building it. The application is actually able to give the required information for the users who are using the app and it is well designed according to the requirements stated by the Y2K Game Station for their users and the users requirements are also taken into consideration while developing the application.

# **Chapter IV: Other Non-Functional Requirements and Conclusion**

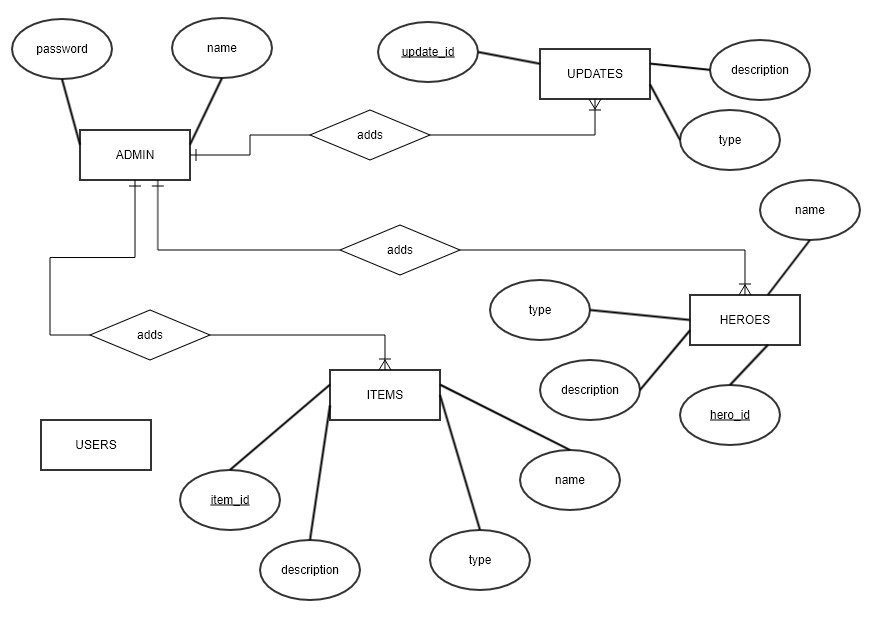
# **5.1 PERFORMANCE REQUIREMENTS**

The steps involved to perform the implementation of staff database are as listed below.

1. **E-R DIAGRAM**

The E-R Diagram constitutes a technique for representing the logical structure of a database in a pictorial manner. This analysis is then used to organize data as a relation, normalizing relation and finally obtaining a relation database.

* **ENTITIES:** Which specify distinct real-world items in an application.
* **PROPERTIES/ATTRIBUTES:** Which specify properties of an entity and relationships.
* **RELATIONSHIPS:** Which connect entities and represent meaningful dependencies between them.

****

**B) NORMALIZATION**

The basic objective of normalization is to reduce redundancy which means that information is to be stored only once. Storing information several times leads to wastage of storage space and increase in the total size of the data stored.

If a database is not properly designed it can give rise to modification anomalies. Modification anomalies arise when data is added to, changed or deleted from a database table. Similarly, in traditional databases as well as improperly designed relational databases, data redundancy can be a problem. These can be eliminated by normalizing a database.

**5.2 SAFETY REQUIREMENTS**

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

**5.3 SECURITY REQUIREMENTS**

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

**5.4 SOFTWARE QUALITY ATTRIBUTES**

* **AVAILABILITY:** The product should be available on the specified date and specified time as many customers.
* **CORRECTNESS:** The product should reach start to the correct destination as per the order made by customer.
* **MAINTAINABILITY:** The administrators should maintain correct record of the inventory and their suppliers
* **USABILITY:** The inventory level should satisfy a maximum number of customer’s needs.

# **5.2 Feasibility Analysis**

The analysis of feasibility has conclude that the project is feasible with respect to time and cost. The technology used to develop are almost Open Source, therefore less cost for implementation and maintenance will be involved.

FMS is economically feasible to be implemented because it is a desktop based application.

FMS can be implemented in various technologies which are presently available as well as in all technologies which will be implemented in future.

Operational feasibility is all about how well SSMS solves problem and takes advantage of identified opportunities during scope definition of the SSMS.

**ORCHID INTERNATIONAL COLLEGE**

**(Faculty of Management)**

**Lab #4 WBS**

**Date: 15th July 2019**

**Submitted By Submitted To**

**Name: Safal Lamichhane Department of BIM**

**TU Roll No: 6685/15**

**Work Breakdown Structure**

A **work-breakdown structure** (**WBS**) is a method for getting a complex, multi-step project done. It's a way to divide and conquer large projects so you can get things done faster and more efficiently. Work breakdown structure (or WBS) is a hierarchical tree structure that outlines your project and breaks it down into smaller, more manageable portions. In Wrike, you can build a WBS by creating folders and subfolders, and can even go further to divide individual tasks into subtasks. The goal of a WBS is to make a large project more manageable. Breaking it down into smaller chunks means work can be done simultaneously by different team members, leading to better team productivity and easier project management overall.

A work-breakdown structure element may be a [product](https://en.wikipedia.org/wiki/Product_(business)), [data](https://en.wikipedia.org/wiki/Data), [service](https://en.wikipedia.org/wiki/Service_(economics)), or any combination thereof. A WBS also provides the necessary framework for detailed cost estimating and control along with providing guidance for [schedule](https://en.wikipedia.org/wiki/Schedule_(project_management)) development and control.

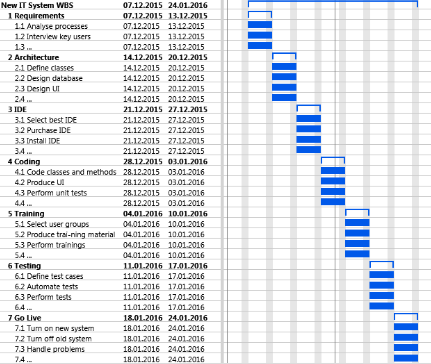


Fig 4.1: WBS for new System

**ORCHID INTERNATIONAL COLLEGE**

**(Faculty of Management)**

**Lab #5 VCS**

**Date: 17th July 2019**

**Submitted By Submitted To**

**Name: Safal Lamichhane Department of BIM**

**TU Roll No: 6685/15**

**Version Control System (VCS)**

**Version Control**, also known as **Revision Control** or **Source Code Management**, is the management of changes to documents, computer programs, websites or other collection of information.

These changes are often identified as numbers or letter code, termed as revision number. Each revision is associated with meta data like timestamp, author(a person who made changes), etc. Revisions can be compared, restored or merged.

**Benefits**

* **Version tracking and branching**

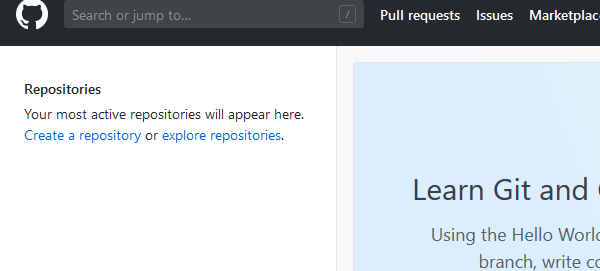
As each and every change is persisted, it allows you to compare files, identify changes and merge the changes if needed prior to committing the code. It also helps to identify which version is currently in development, QA or production. You can also identify easiily the changes which make or break the code.

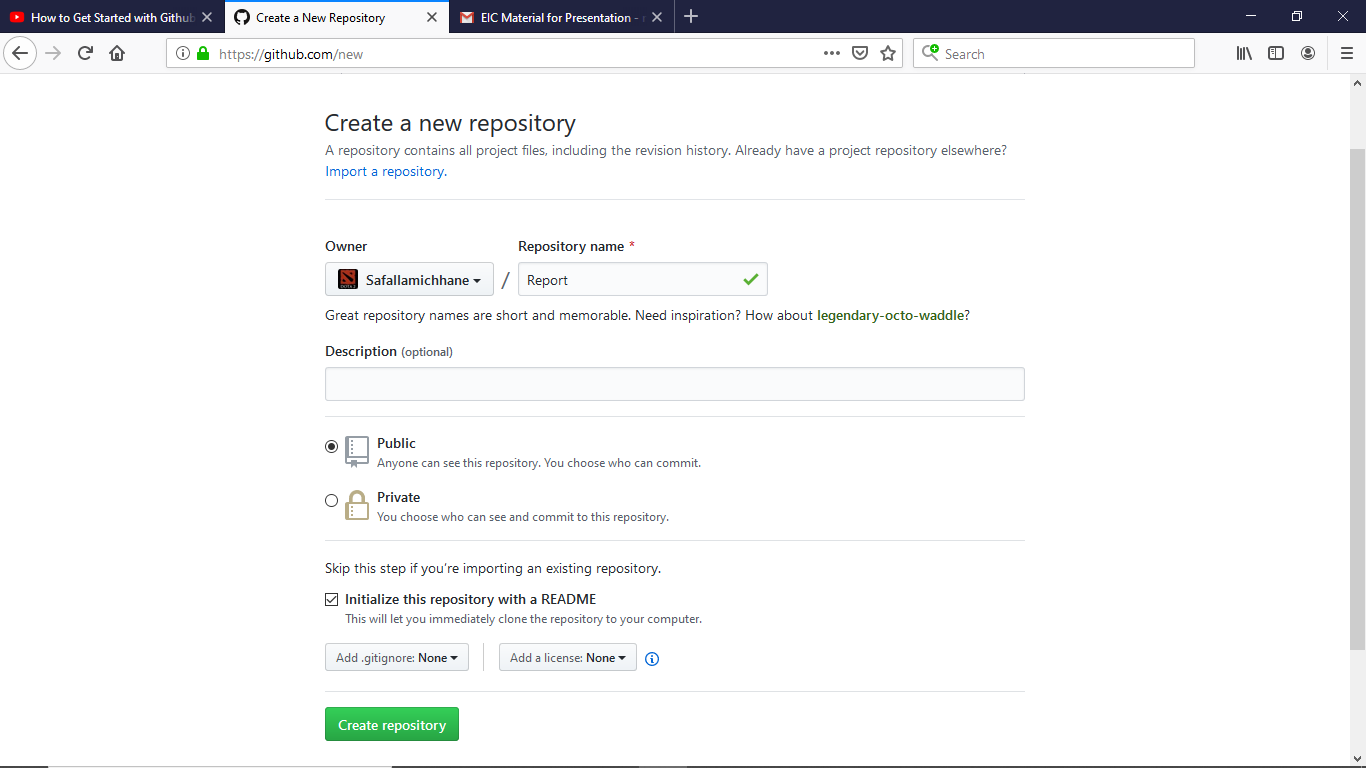
### **Step 0: Install git and create a GitHub account**

The first two things you'll want to do are install git and create a free GitHub account. Once you've done that, create a GitHub account [here](https://github.com/join).  (Accounts are free for public repositories, but there's a charge for private repositories.)

### **Step 1: Create a local git repository**

When creating a new project on your local machine using git, you'll first create a new [**repository**](https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository)(or often, '**repo**', for short). To begin, open up a terminal and move to where you want to place the project on your local machine using the cd (change directory)command.





### **Step 2: Initializing**



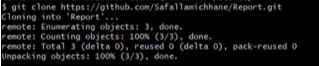
### **Step 3 : Create a commit**

It's time to create your first commit!

C:\Users\Safal\Desktop\2.PNG

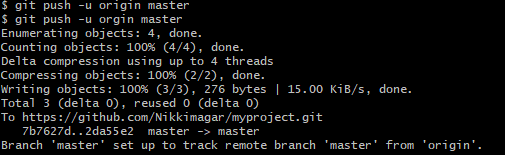
### **Step 4: Cloning in github**

Now that you've made a new commit, let's try something a little more advanced.



### **Step 5: Pushing to GitHub**

Now we’ll push the commit in your branch to you new GitHub repo. This allow other people to see the changes you have made, if they’re approved by the repository’s owner, the changes can then be merged into the master branch.



### **Step 6: Create a Pull Request (PR)**

A pull request (or PR) is a way to alert a repo's owners that you want to make some changes to their code. It allows them to review the code and make sure it looks good before putting your changes on the master branch.

This is what the PR page looks like before you've submitted it:

