Experiment - 1

Aim: To determine the nature of roots of a quadratic equations, its input is triple of +ve integers (say x,y,z) and values may be from interval[1,100] the program output may have one of the following:- [Not a Quadratic equations, Real roots, Imaginary roots, Equal roots] Perform BVA.

```
#include<iostream.h>
#include<conio.h>
void main()
{
	int a,b,c,d;
	clrscr();
	cout<<"The Quadratic equation is of the type a(x^2)+bx+c=0"<<endl;
	cout<<"Enter the value of a: "<<endl;
	cin>>a;
	cout<<"Enter the value of b: "<<endl;
	cin>>b;
	cout<<"Enter the value of c: "<<endl;
	cin>>c;
	d=(b*b)-4*a*c;
	if((a<0)||(b<0)||(c<0)||(a>100)||(b>100)||(c>100))
		cout<<"Invalid input"<<endl;
	else if(a==0)
		cout<<"Not a quadratic equation"<<endl;
	else if (d==0)
		cout<<"Roots are equal"<<endl;
	else if(d<0)
		cout<<"Imaginary roots"<<endl;
	else
		cout<<"Real Roots"<<endl;
	getch();
}
```

Output of the above program:

```
The quadratic equation is of the type a(x^2) +bx +c = 0
Enter the value of a:
50
Enter the value of b:
50
Enter the vaule of c:
0
Real roots
```

```
The quadratic equation is of the type a(x^2) +bx +c = 0
```

Enter the value of a:
50
Enter the value of b:
50
Enter the vaule of c:
50
Imaginary roots

The quadratic equation is of the type a(x^2) +bx +c = 0
Enter the value of a:
10
Enter the value of b:
40
Enter the vaule of c:
40
Roots are equal

The quadratic equation is of the type a(x^2) +bx +c = 0
Enter the value of a:
0
Enter the value of b:
50
Enter the vaule of c:
50
Not a quadratic equation

Boundary Value analysis: The basic idea of boundary value analysis is to use input variable at their manimum, just above manimum, normal value, just below maximum and maximum.

In this program, we consider the value as 0 (minimum), 1(just above minimum), 50 (normal), 99 (just below maximum) and 100 (maximum).

Test case ID	a	b	c	Expected output
1.	50	50	0	Real roots
2.	50	50	1	Real roots
3.	50	50	50	Imaginary roots
4.	50	50	99	Imaginary roots
5.	50	50	100	Imaginary roots
6.	50	0	50	Imaginary roots
7.	50	1	50	Imaginary roots
8.	50	99	50	Imaginary roots
9.	50	100	50	Equal roots
10.	0	50	50	Not a QE
11.	1	50	50	Real roots
12.	99	50	50	Imaginary roots
13.	100	50	50	Imaginary roots

Experiment- 2

Aim: To determine the type of triangle. Its input is triple of +ve integers (say x,y,z) and the values may be from interval[1,100].The program output may be one of the following [Scalene, Isosceles, Equilateral, Not a Triangle].Perform BVA

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
void main()
{
clrscr();
int ch;
char c;
float b,h,a;
1: cout<<"Enter your choice";
   cout<<"\n 1. Triangle";
   cout<<"\n 2. Square";
   cout<<"\n 3. Rectangle";
   cout<<"\n 4. Circle";
   cout<<"\n 5. Exit\n";
   cin>>ch;
switch(ch)
{
case 1: b: cout<<"\Enter the base of the triangle (1-200)";
              cin>>b;
              if((b<=0)||(b>200))
              {
              cout<<"\n Invalid entry for base \n";
              goto b;
              }
           h: cout<<"\n Enter the height of the triangle (1-200)";
              cin>>h;
              if((h<=0)||(h>200))
              {
              cout<<"\n Invalid height\n Enter the valid height (1-200)";
              goto h;
              }
           a= 0.5*b*h;
           cout<<"\n The area is : "<<a;
           cout<<"\n Want to enter more?(y/n)";
           cin>>c;
           if((c=='y')||(c=='y'))
           goto 1;
           break;
case 2 : s: cout<<"\n Enter the side of the squre (1-200)";
              cin>>b;
              if((b<=0)||(b>200))
              {
```

```
            cout<<"\n Invalid entry for base \n";
            goto s;
            }
        a=b*b;
        cout<<"\n The area is :"<<a;
        cout<<"\n Want to enter more?(y/n)";
        cin>>c;
        if((c=='y')||(c=='y'))
        goto 1;
        break;
case 3: d: cout<<"\n Enter the Base of the rectangle (1-200)";
            cin>>b;
            if((b<=0)||(b>200))
            {
            cout<<"\n Invalid entry for base \n";
            goto d;
            }
        p: cout<<"\n Enter the height of the rectangle (1-200)";
           cin>>h;
           if((h<=0)||(h>200))
           {
           cout<<"\n Invalid Height \n Enter the height (1-200)";
           goto p;
           }
        a=b*h;
        cout<<"\n The area is :"<<a;
        cout<<"\n Want to enter more?(y/n)";
        cin>>c;
        if((c=='y')||(c=='y'))
        goto 1;
        break;
case 4 : t: cout<<"\n Enter the radius of the circle (1-200)";
            cin>>b;
            if((b<=0)||(b>200))
            {
            cout<<"\n Invalid entry for radius \n";
            goto t;
            }
        a=3.14*b*b;
        cout<<"\n The area is :"<<a;
        cout<<"\n Want to enter more?(y/n)";
        cin>>c;
        if((c=='y')||(c=='y'))
        goto 1;
        break;
case 5 : exit(0);
          break;
default : cout<<"\n Wrong Choice:";
          goto 1;
          }
        getch();
```

}

Test cases: In Equivalence class testing, we find two types of equivalence classes; input domain and output domain;
Input domain is formed from one valid sequence and two invalid sequences. The output domain is obtained from different types of output of a problem.

For Triagle:

Input domain:::

I1 = {h : h<=0}
I2 = {h : H>200}
I3 = {h : 1<=h<=200}
I4 = {b : b<=0}
I5 = {b : b>200}
I6 = {b : 1<=b<=200}

Test cases:

Test case ID	h	b	Expected output
1.	0	100	Invalid input
2.	100	100	5000
3.	201	100	Invalid input
4.	100	0	Invalid input
5.	100	100	5000
6.	100	201	Invalid input

Output domain:::

O1 = {<h,b> : triangle in h > 0,b>0}
O2 = {<h,b> : Not a triangle if h<=0, b<=0}

Output screen shots for the triangle is:

Enter your choice:
1. Triangle
2. Square
3. Rectangle
4. Circle
5. Exit

1
Enter the base of the triangle (1 – 200) 0
Invalid entry for base
Enter the base of the triangle (1 – 200) 100
Enter the height of the triangle (1 – 200) 201
Invalid height
Enter the Height(1-200)
Enter the height of the triangle(1 – 200) 100
The area is 5000
Want to enter more? (y/n)y

For square:

Input domain:::

I1={s : s<=0}
I2={s : s>200}
I3={s : 1<=s<=200}

Test cases:

Test case ID	s	Expected output
1.	0	Invalid input
2.	100	10000
3.	201	Invalid input

Output domain:::

O1 = { <s> : square if s>0}
O2 = { <r> : Not a square if s<=0}

Output screen shots for the square is:

Enter your choice:

1. Triangle
2. Square
3. Rectangle
4. Circle
5. Exit

2

Enter the side of the square (1 – 200) 0
Invalid entry for side
Enter the side of the square (1 – 200) 201
Invalid entry for side

Enter the side of the square(1 – 200) 100
The area is 10000
Want to enter more? (y/n)y

For Rectangle:

Input domain:::

I1 = { l : l <=0}
I2 = { l : l>200}
I3 = { I : 1<=l <=200}
I4 = { b : b<=0}
I5 = { b : b>200}
I6 = { b : 1<=b<=200}

Test cases:

Test case ID	l	b	Expected output
1.	0	100	Invalid input
2.	100	100	10000
3.	201	100	Invalid input
4.	100	0	Invalid input
5.	100	100	10000
6.	100	201	Invalid input

Output domain:::

O1 = {<l,b> : rectangle if l > 0,b>0}
O2 = {<l,b> : Not a triangle if l<=0, b<=0}

Output screen shots for the Rectangle is:

Enter your choice:

1. Triangle
2. Square
3. Rectangle
4. Circle
5. Exit

3
Enter the length of the rectangle (1 – 200) 0
Invalid entry for length
Enter the length of the rectangle (1 – 200) 201
Invalid entry for length
Enter the length of the rectangle(1 – 200) 100
Enter the breadth of the rectangle (1 – 200) 100

The area is 10000
Want to enter more? (y/n)y

For Circle:

Input domain:::

I1 = {r: r<=0}
I2 = {r : r>200}
I3 = { r: 1<=r<=200}

Test cases:

Test Cases	r	Expected output
1.	0	Invalid input
2.	100	31400

3.	201	Invalid input

Output domain:

O1 : {<r>: Circle if 1<=r<=200}
O2 : {<r>: not a circle if r<=0}

Output screen shots for the Circle is:

Enter your choice:

1. Triangle
2. Square
3. Rectangle
4. Circle
5. Exit

4
Enter the radius of the circle (1 – 200) 0
Invalid entry for radius
Enter the radius of the circle (1 – 200) 201
Invalid entry for radius
Enter the radius of the circle(1 – 200) 100
The area is 31400
Want to enter more? (y/n)y

EXPERIMENT-3

Aim: Calculate the value of a^b. Perform Decision Table based testing on it.

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
void main()
{
  clrscr();
  int a,b;
  float c;
  char ch;
1: cout<<"To Calculate 'a to the power b' \n";
   cout<<"Enter the value of 'a' \n";
   cin>>a;
   cout<<"Enter the value of 'b' \n";
   cin>>b;
   c=pow(a,b);
   cout<<endl<<c;
   cout<<"\n Want to enter again? (y/n)";
   cin>>ch;
   if((ch=='y')||(ch=='y'))
   goto 1;
   getch();
}
```

Test cases: Decision Table Based testing is useful for describing situations in which a number of combination of actions are taken for different conditions. There are four parts of a decision table; condition stub, action stub, condition entries and action entries.

Test case ID	A	B	Expected output
1.	2	3	+ve result
2.	-1	3	-ve result
3.	-2	-4	+ve result
4.	0	1	Result is 0
5.	0	0	Domain Error
6.	-1	-0.6	Result is 1

Decision table is:

Conditions are:::
C1 : a = 0, b = 0
C2 : a = -ve, b = +ve
C3 : a = +ve, b = -ve
C4 : a = -ve, b = -ve
C5 : a = +ve, b = +ve
C6 : a = 0,b = integer

C7 : b = 0, a = integer
C8 : a = -ve, b= -ve odd

Actions:::

A1 : Domain error
A2 : Negative output
A3 : output =1
A4 : positive output
A5 : output = 0

Condition	R1	R2	R3	R4	R5	R6	R7	R8
C1	T	-	-	-	-	-	-	-
C2	-	T	-	-	-	-	-	-
C3	--	-	-	T	-	-	-	-
C4	-	-	--	-	T	-	-	-
C5	-	-	-	-	-	T	-	-
C6	-	-	-	-	-	-	T	-
C7	-	-	-	-	-	-	-	T
C8	-	-	T	-	-	-	-	-
Action								
A1	X							
A2			X		X			
A3								X
A4		X		X		X		
A5							X	

The output for the above program is:

To calculate’a to the power of b’
Enter the value of ‘a’
0
Enter the value of ‘b’
0
Pow: Domain error
1
Want to enter again? (y/n) y
To calculate’a to the power of b’
Enter the value of ‘a’
2
Enter the value of ‘b’
3

8
Want to enter again? (y/n)n

Experiment- 4

Aim: write a program in C/C++ to compute previous data. We are given the present date as input. Also perfome the robust case analyses on it.

```
#include<stdio.h>
#include<conio.h>
int main()
{
int day, month, year, flag=0;
printf("Enter the day value:");
scanf("%d",&day);
printf("Enter the month value:");
scanf("%d",&month);
printf("Enter the year value:");
scanf("%d",&year);
if(year>=1900 && year<=2025)
{
  if(month==1||month==3||month==5||month==7||month==8||month==10||month==12)
   {
    if(day>=1 && day<=31)
      {
            flag=1;
      }
      else
      {
      flag=0;
      }
   }
   else if(month==2)
   {
   int rval=0;
   if(year%4==0)
           {
           rval=1;
           if((year%100)==0 && (year%400)!=0)
            {
            rval=0;
            }
   }
   if(rval==&&(day>=1 &&day <=29))
     {
     flag=1;
     }
  else if (day>=1 &&day <=28)
     {
     flag=1;
     }
  else
     {
     flag=0;
```

```
    }
}
if(flag)
{
  if(day==1)
  {
   if(month==1)
   {
    year--;
    day=31;
    month=12;
   }
   else if(month==3)
   {
    int rval=0;
    if(year%4==0)
    {
          rval=1;
          if((year%100)==0 &&(year%400)!=0)
          {
           rval=0;
           }
          }
          if(rval==1)
          {
           day=29;
           month--;
           }
           else
           {
           day=28;
           month--;
           }
          }
          else if(month==2||month==4||month==6||month==9||month==11)
          {
            day=31;
            month--;
          }
          else
          {
          day=30;
          month--;
          }
   }
   else
   {
   day--;
   }
   printf(" The next data is : %d-%d-%d",day,month,year);
}
```

```
  else
  {
   printf(" The entered data is invalid: %d-%d-%d",day,month,year);
  }
  getch();
  return 1;
}}
```

Test cases:

Test Case ID	Month	Day	Year	Expected output
1.	June	1	1964	31 May, 1964
2.	June	31	1984	Impossible
3.	May	1	1945	30 April, 1945
4.	March	31	2007	30 March, 2007
5.	August	29	2007	28 August, 2007
6.	Februry	29	1962	impossible

The output for the above program is:

Enter the day value; 3
Enter the month value ; 3
Enter the year value ; 2000
The previous day is 2 – 3 – 2000

Experiment- 5

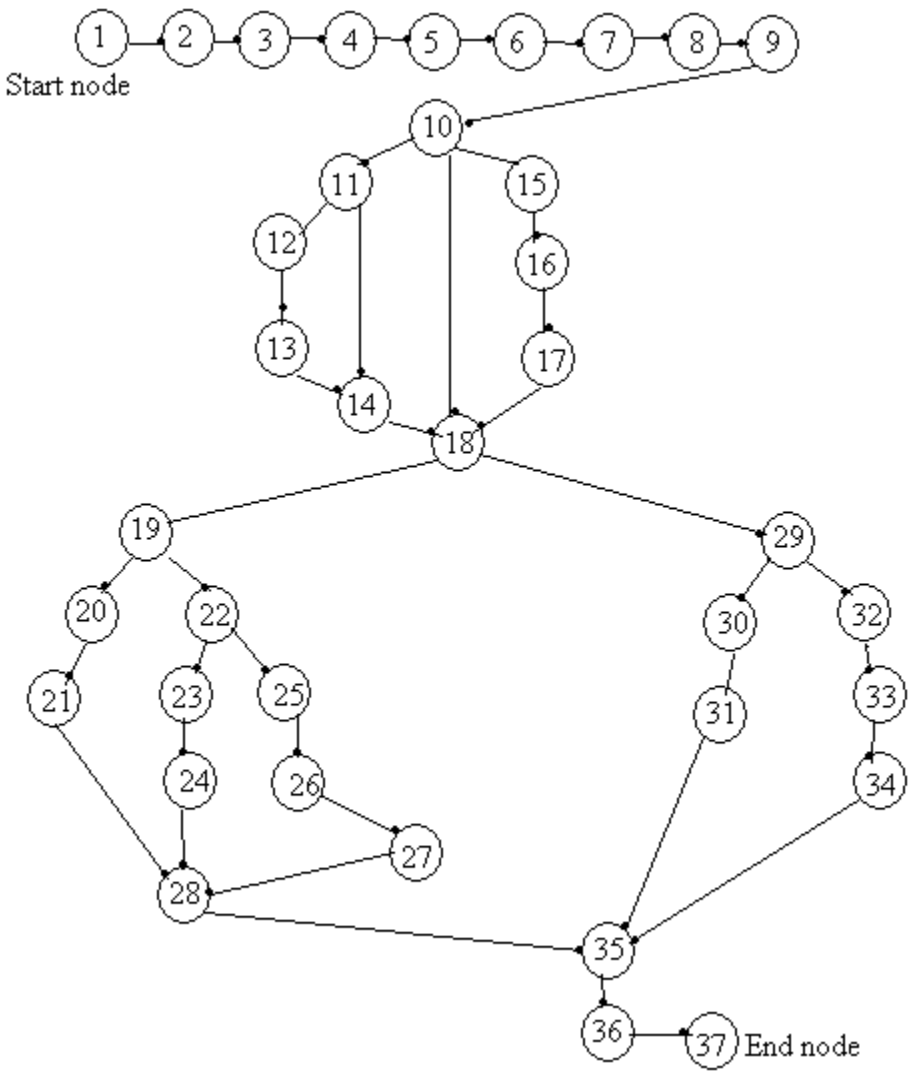
Aim: Read three sides of a triangle and to check whether the triangle is Isoceles, equilateral or scalane. Perform path testing on it and deduce:

1. **Flow graph**
2. **DD path graph**
3. **Independent path**
4. **Cyclomatic complexity**

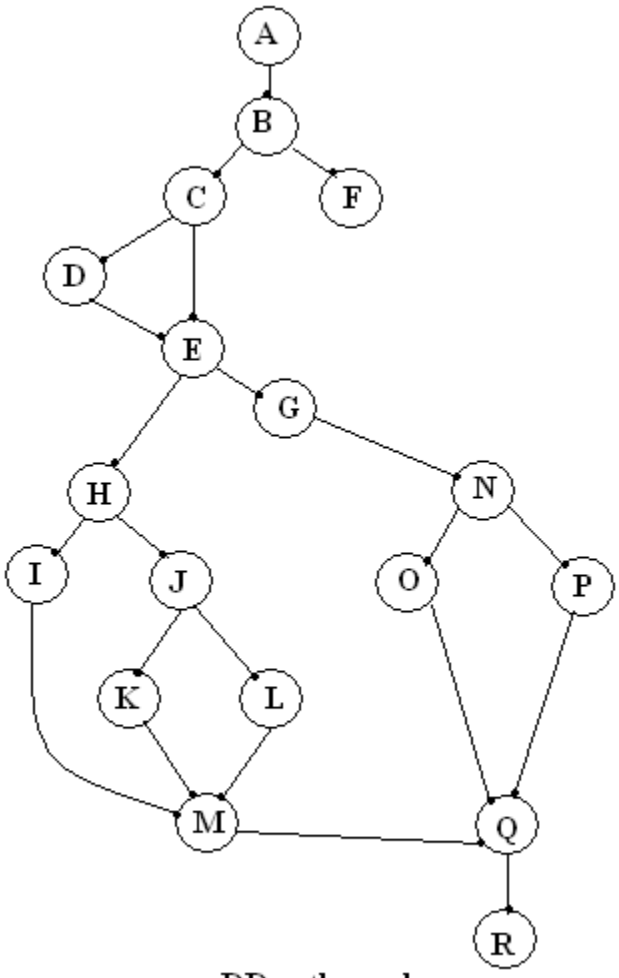
```
#include<stdio.h>
#include<conio.h>
int main()
{
	int a,b,c,validinput=0;
	printf("Enter the side ‘a’ value");
	scanf("%d",&a);
	printf("Enter the side ‘b’ value");
	scanf("%d",&b);
	printf("Enter the side ‘c’ value");
	scanf("%d",&c);
	if((a>0)&&(a<100)&&(b>0)&&(b<100)&&(c>0)&&(c<100))
	{
		if(((a+b)>c)&&((c+a)>b)&&((b+c)>a))
		{
			validinput=1;
		}
	}
	else
	{
		validinput=-1;
	}
	if(validinput==1)
	{
		if((a==b)&&(b==c))
		{
			printf("The triangle is equilateral");
		}
		else if((a==b)||(b==c)||(c==a))
		{
			printf("The triangle is isosceles");
		}
		else
		{
			printf("The triangle is scalene");
		}
	}
	else if(validinput==0)
	{
		printf("The value do not constitute the triangle");
	}
```

```
        else
        {
                printf("The input belongs to invalid range");
        }
        getch();
        return 1;
}
```

1. **Flow Graph:** It shows that how the control and data is flow from one node to another node. The flow graph for the following program is given below:



2. **DD path graph:**



DD path graph

Flow Graph Nodes	DD Path Graph Corr. Nodes	Remarks
1 to 9	A	Sequential
10	B	Decision
11	C	Decision
12,13	D	Sequential
14	E	Two edges are joined here
15,16,17	F	Sequentian nodes
18	G	Decision nodes plus joining of two edges
19	H	Decision nodes
20,21	I	Sequential nodes
22	J	Decision nodes
23,24	K	Sequential nodes
25,26,27	L	Sequential nodes
28	M	Three edges combined here
29	N	Decision nodes
30,31	O	Sequential nodes

32,33,34	P	Sequential nodes
35	Q	Three edges comibined here
36,37	R	Sequential nodes with exit node

3. **Independent path** : It is a path in the flow graph that has at least one edge that has not been traversed before in other path.

1. ABFGNPQR
2. ABFGNOQR
3. ABCEGNPQR
4. ABCDGINOQR
5. ABFGHIMQR
6. ABFGHJKMQR
7. ABFGHJLMQR

4. **Cyclomatic complexity**: To calculate the cyclomatic complexity we calculate the following formula:

$$\mathbf{V(G) = e - n + 2P}$$

Here : e = 23
n = 18 and
p = 1
so the total no of independent paths are **23 – 18 + 2 = 7**

The output for the above program is :

Enter the side ‘a’ value: 12
Enter the side ‘b’ value: 12
Enter the side ‘c’ value: 12
The triangle is Equilateral

Enter the side ‘a’ value: 45
Enter the side ‘b’ value: 45
Enter the side ‘c’ value: 23
The triangle is isosceles

Experiment- 6

Aim: Compute total salary of an employee given his/her basic salary. The slab is given below:

HRA = 30 % of basic
DA=80% of basic
MA = Rs. 100/-
TA = Rs. 800/-
I.tax = Rs. 700/-
PF = Rs. 780/- Also find out the path graph and cyclomatic complexity.

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
void main()
{
        clrscr();
        cout<<"Enter the basic salary of the employee";
        float bs;
        cin>>bs;
        int hra,da,ma=100,itax=700, pf=780,ta=800;
        hra=0.3*bs;
        da=0.8*bs;
        cout<<"\nHouse allowance: rs."<<hra;
        cout<<"\nDarkness allowance: rs."<<da;
        cout<<"\nmedical allowance: rs."<<ma;
        cout<<"\ntravel allowance: rs."<<ta;
        cout<<"\nIncome tax rs."<<itax;
        cout<<"\nprovidend fund: rs."<<pf;
        float netsal;
        netsal=(bs+hra+da+ta-itax-pf);
        cout<<endl;
        cout<<"The net selary of the employee is : "<<netsal;
        getch();
}
```

The output of the above program is :

```
Enter the basic salary of the employee10000

House allowance: rs.2999
Darkness allowance: rs.8000
medical allowance: rs.100
travel allowance: rs.800
Income tax rs.700
providend fund: rs.780
The net selary of the employee is : 20319
```

Path Graph

The **Cyclomatic complexity** for the above path graph is:

To calculate the cyclomatic complexity we use the formula:

$$V(G) = e - n + 2P$$

Here: e = 15
n = 16
p = 1

hence V(G) = 1

Practical 7:

AIM: To study Test Complete

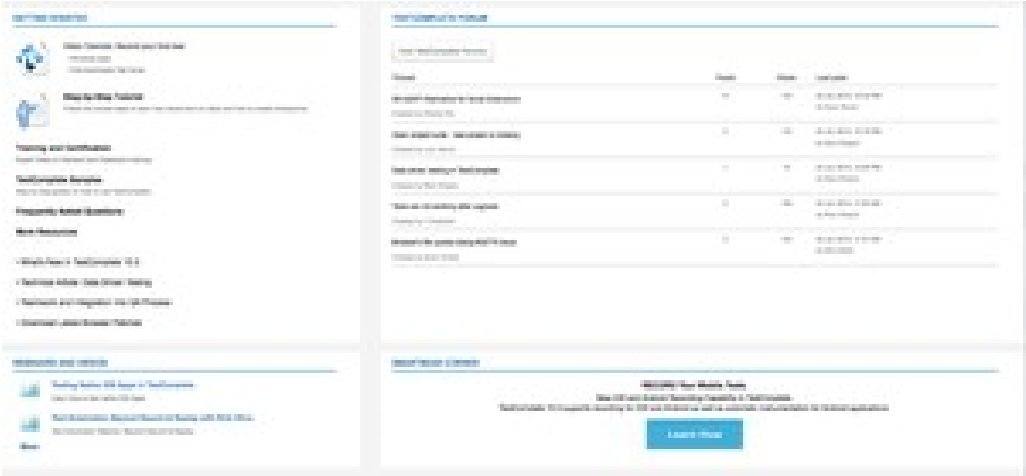
TestComplete is a functional automated testing platform developed by SmartBear Software. TestComplete gives testers the ability to create automated tests for Microsoft Windows, Web, Android (operating system), and iOS applications. Tests can be recorded, scripted or manually created with keyword driven operations and used for automated playback and error logging.

TestComplete is broken out into three modules:

- Desktop
- Web
- Mobile

Each module contains functionality for creating automated tests on that specified platform.

TestComplete is used for testing many different application types including Web, Windows, Android, iOS, WPF, HTML5, Flash, Flex, Silverlight, .NET, VCL and Java. I t automates functional testing and back-end testing like database testing.



Main Features

- *Keyword Testing*: TestComplete has a built-in keyword-driven test editor that consists of keyword operations that correspond to automated testing actions.
- *Scripted Testing*: TestComplete has a built-in code editor that helps testers write scripts manually. It also includes a set of special plug-ins that help.
- *Test Record and Playback*: TestComplete records the key actions necessary to replay the test and discards all unneeded actions.

- *Distributed Testing*: TestComplete can run several automated tests across separate workstations or virtual machines.
- *Access to Methods and Properties of Internal Objects*: TestComplete reads the names of the visible elements and many internal elements of Delphi, C++Builder, .NET, WPF, Java and Visual Basic applications and allows test scripts to access these values for verification or use in tests.
- *Bug Tracking Integration*: TestComplete includes issue-tracking templates that can be used to create or modify items stored in issue-tracking systems. TestComplete currently supports Microsoft Visual Studio 2005, 2008, 2010 Team System, BugZilla, Jira and AutomatedQA AQdevTeam.
- *Data-driven testing*: Data-driven testing with TestComplete means using a single test to verify many different test cases by driving the test with input and expected values from an external data source instead of using the same hard-coded values each time the test runs.
- *COM-based, Open Architecture*: TestComplete's engine is based on an open API, COM interface. It is source-language independent, and can read debugger information and use it at runtime through the TestComplete Debug Info Agent.
- *Test Visualizer* – TestComplete automatically captures screenshots during test recording and playback. This enables quick comparisons between expected and actual screens during test.
- *Extensions and SDK* - Everything visible in TestComplete — panels, project items, specific scripting objects, and others — are implemented as plug-ins. These plug-ins are included into the product and installed on your computer along with other TestComplete modules. You can create your own plug-ins that will extend TestComplete and provide specific functionality for your own needs. For example, you can create plug-ins or use third-party plug-ins for:
    - Support for custom controls
    - Custom keyword test operations
    - New scripting objects
    - Custom checkpoints
    - Commands for test result processing
    - Panels
    - Project items
    - Menu and toolbar items

Program no 8

Aim: Learn how to raise and report Bugs using Bug tracking tool (Bugzilla,Jira using QA Complete)

Introduction- What is Bugzilla

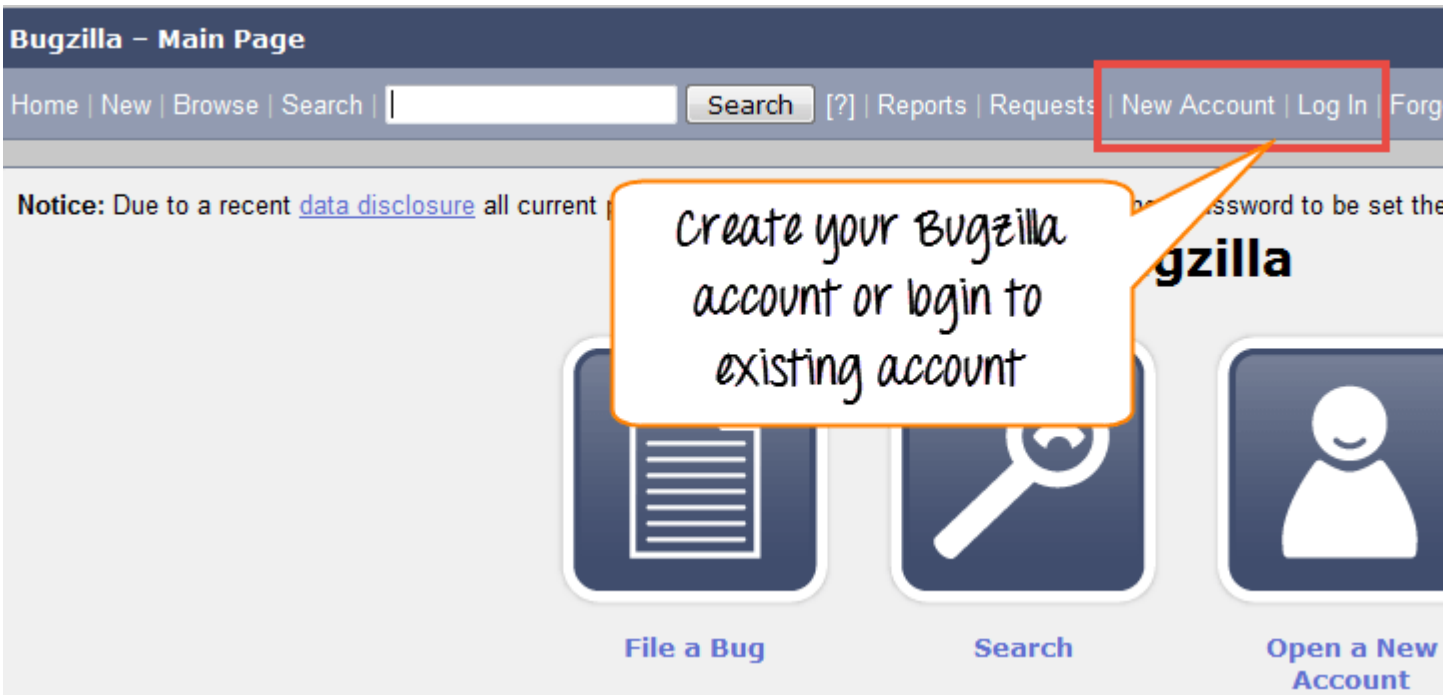
Bugzilla is an open-source issue/bug tracking system that allows developers effectively to keep track of outstanding problems with their product. It is written in Perl and uses MYSQL database.

Bugzilla is a Defect tracking tool, however it can be used as a test management tool as such it can be easily linked with other Test Case management tools like Quality Center, Testlink etc.

This open bug-tracker enables users to stay connected with their clients or employees, to communicate about problems effectively throughout the data-management chain.

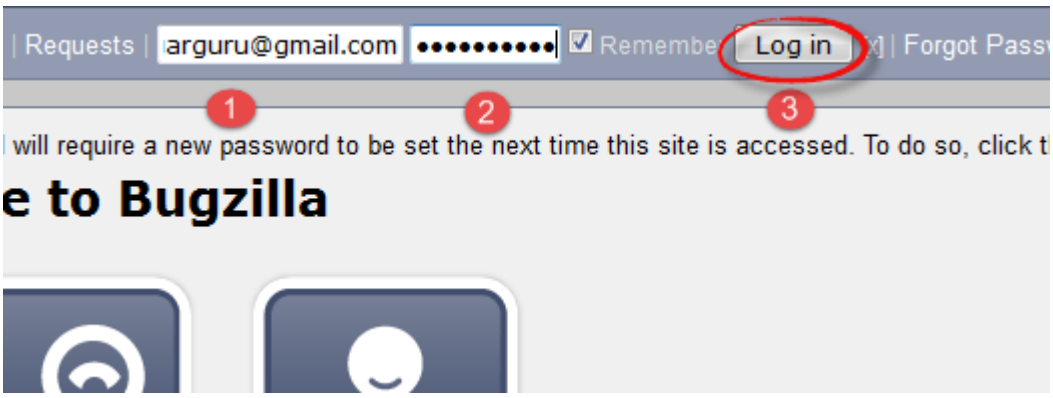
How to log-in to Bugzilla

Step 1) Use the following link for your handons. To create an account in Bugzilla or to login into the existing account go to **New Account or Log in** option in the main menu.

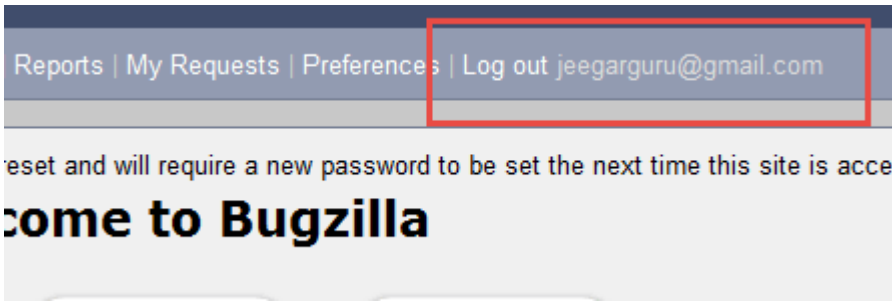


Step 2) Now, enter your personal details to log into Bugzilla

1. User ID
2. Password
3. And then click on **"Log in"**



Step 3) You are successfully logged into Bugzilla system



Creating a Bug-report in Bugzilla

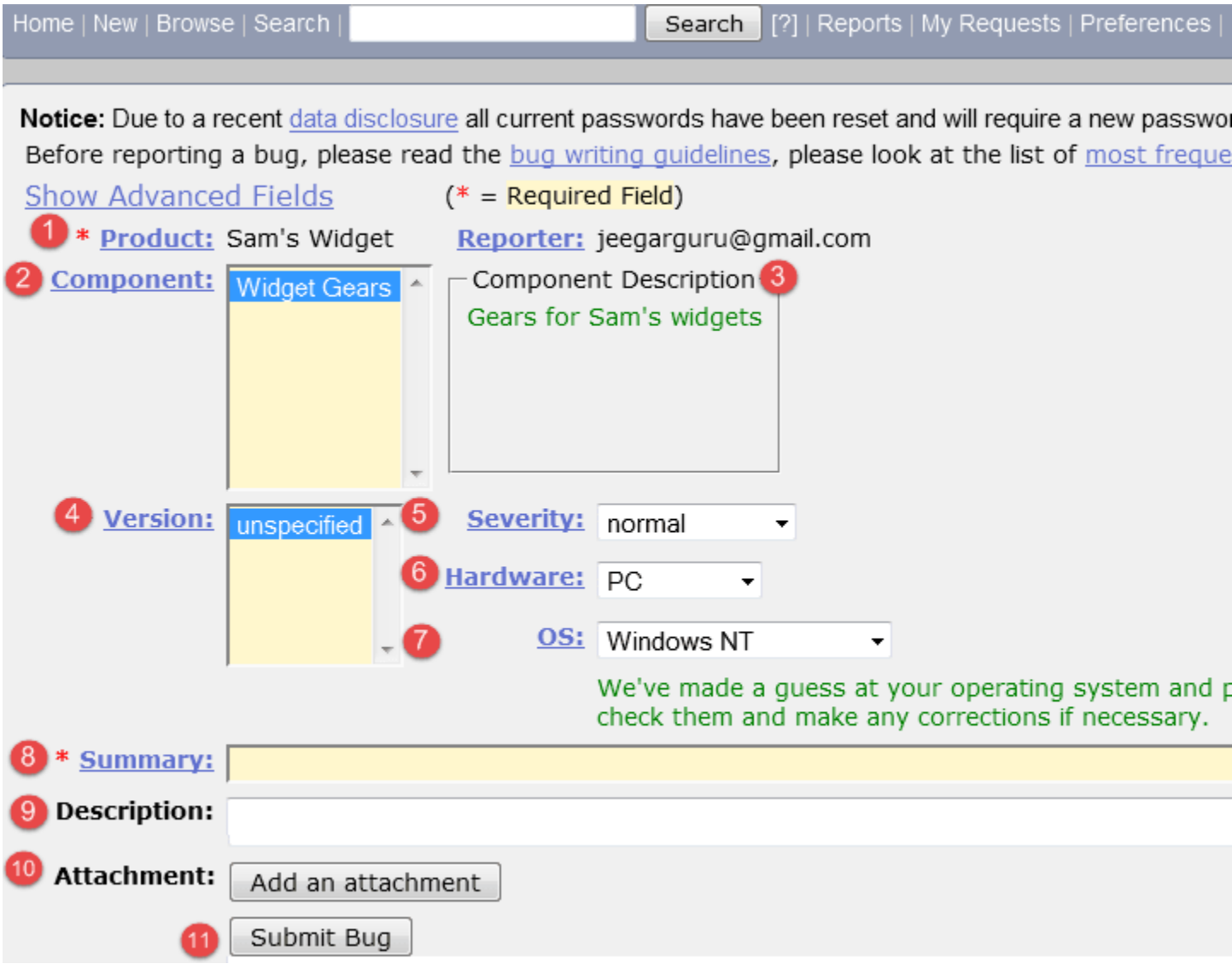
Step 1) To create a new bug in Bugzilla, visit the home-page of Bugzilla and click on **NEW** tab from the main menu

Step 2) In the next window

1. Enter Product
2. Enter Component
3. Give Component description
4. Select version,
5. Select severity
6. Select Hardware

7. Select OS
8. Enter Summary
9. Enter Description
10. Attach Attachment
11. Submit

NOTE: The above fields will vary as per your customization of Bugzilla



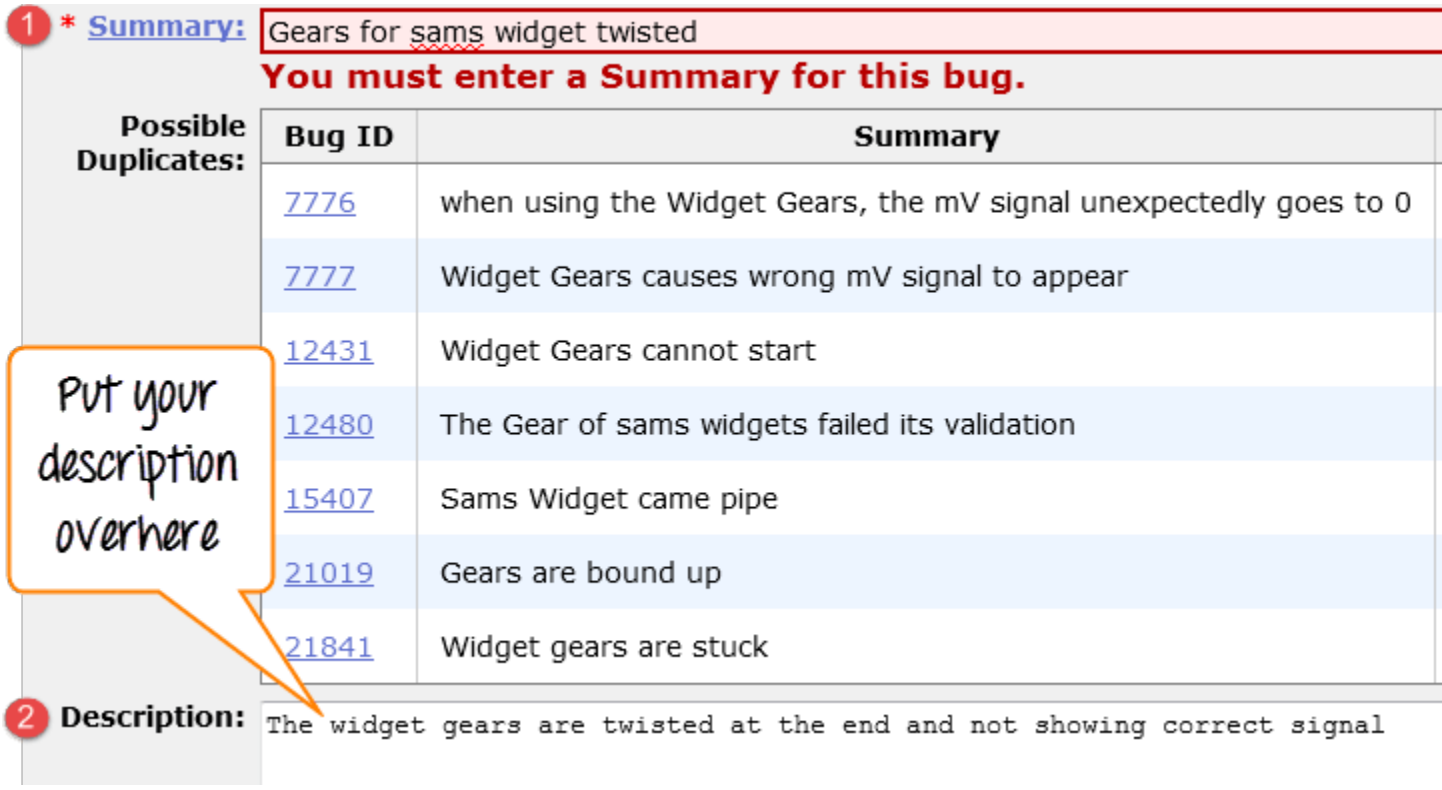
NOTE: The mandatory fields are marked with *.

In our case field's

- Summary
- Description

Are mandatory

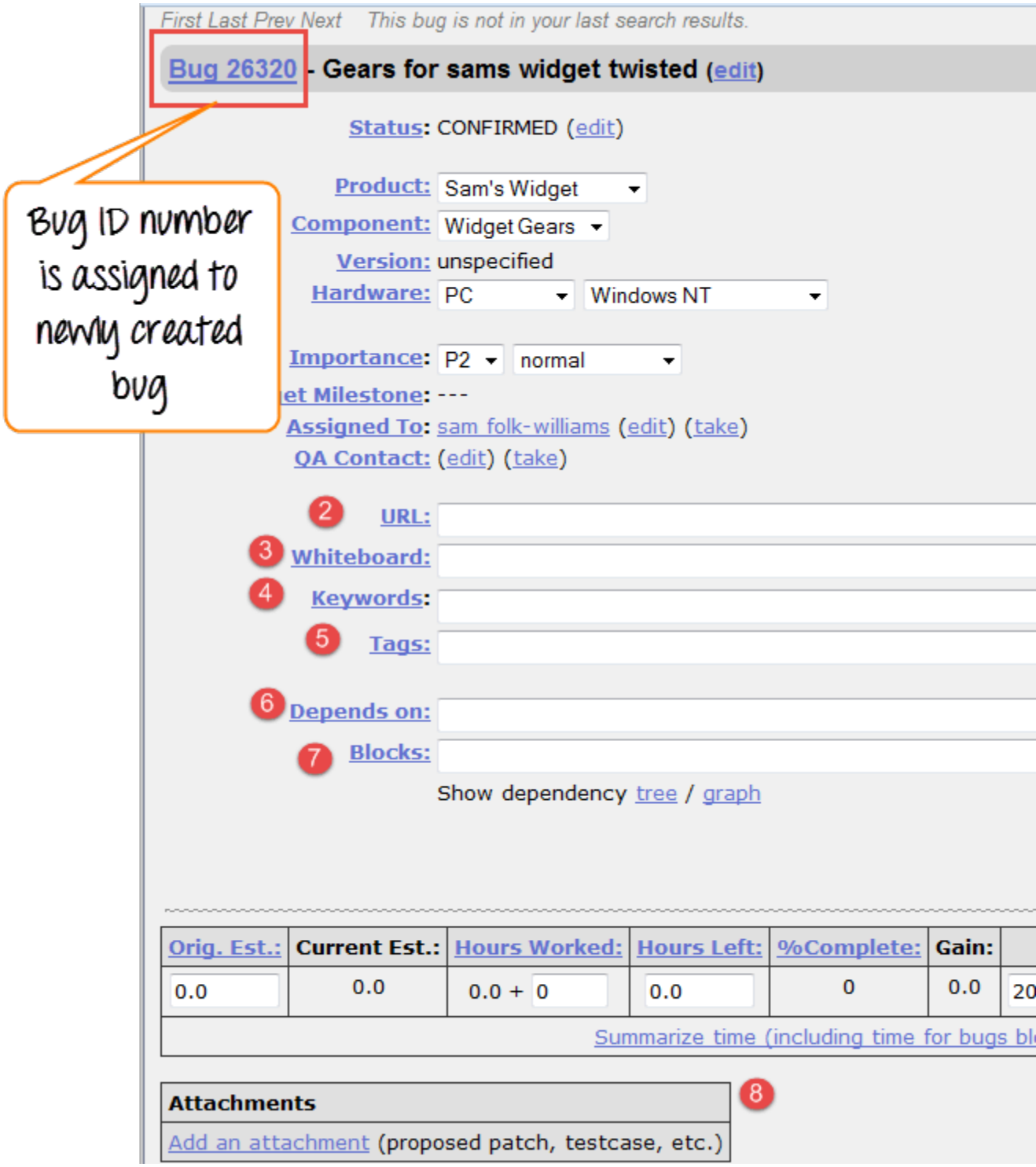
If you do not fill them you will get a screen like below



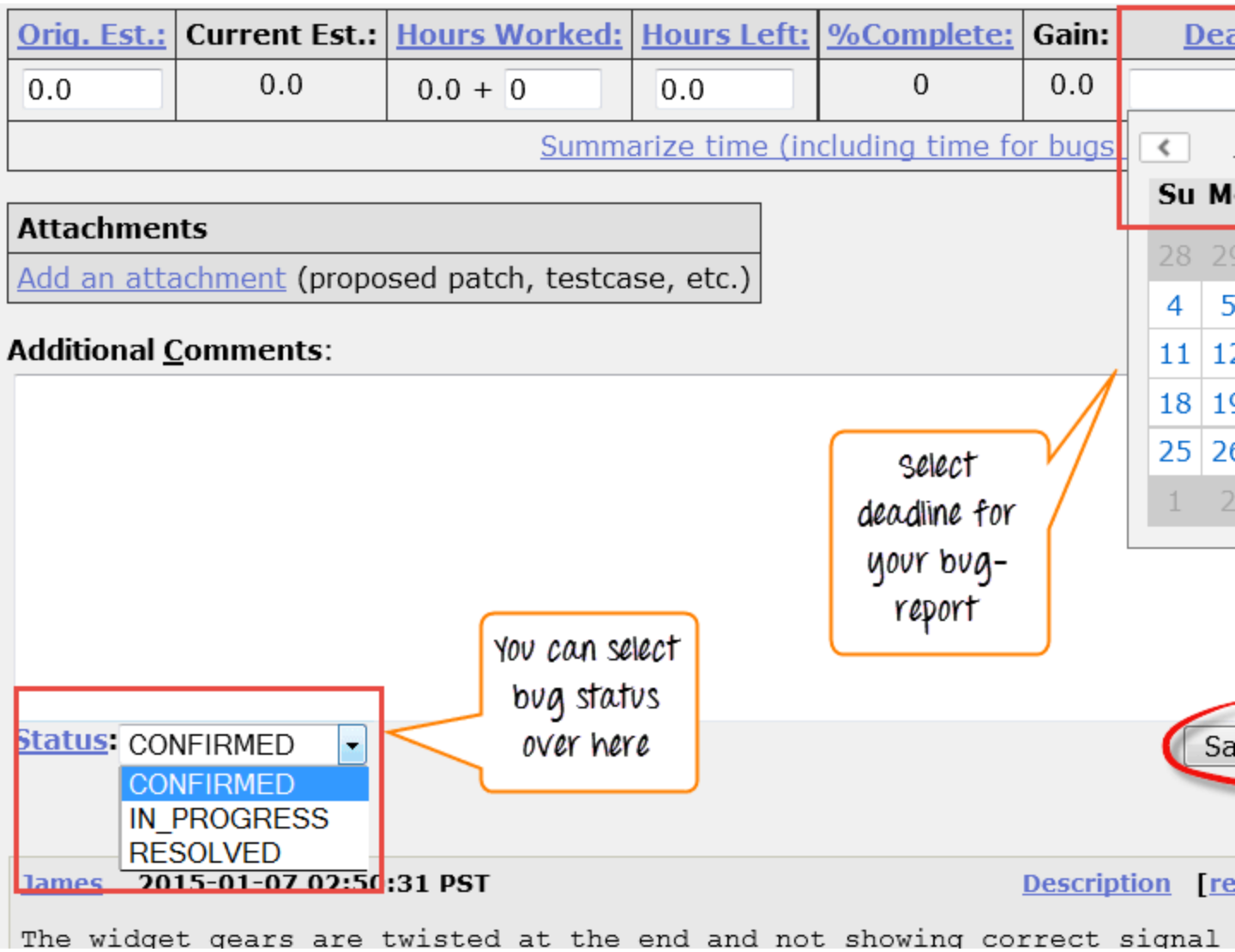
Step 4) Bug is created ID# 26320 is assigned to our Bug. You can also add additional information to the assigned bug like URL, keywords, whiteboard, tags, etc. This extra-information is helpful to give more detail about the Bug you have created.

1. Large text box
2. URL
3. Whiteboard
4. Keywords
5. Tags
6. Depends on
7. Blocks

8. Attachments



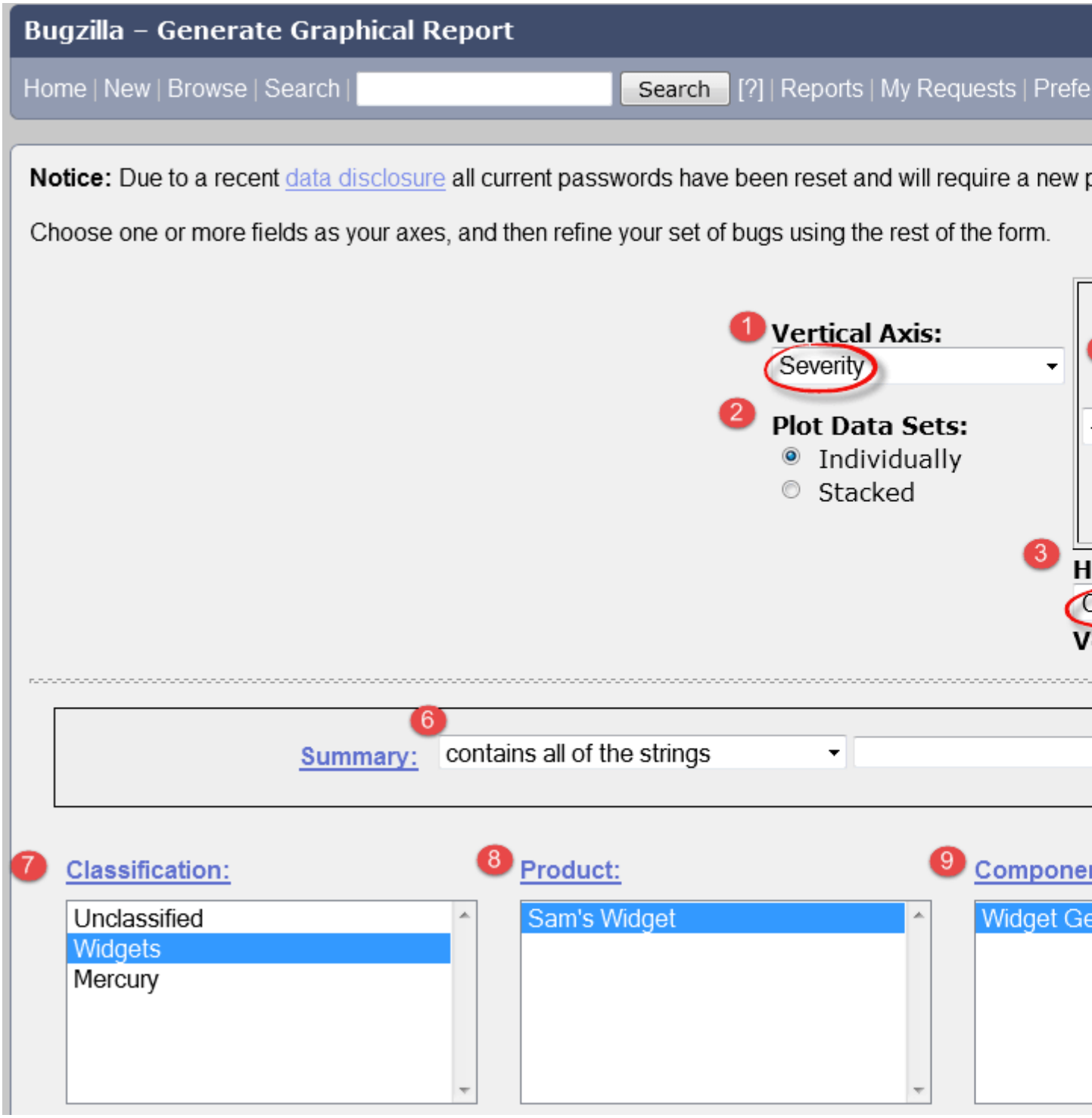
Step 5) In the same window if you scroll down further. You can select deadline date and also status of the bug. **Deadline in Bugzilla usually gives the time-limit to resolve the bug in given time frame.**



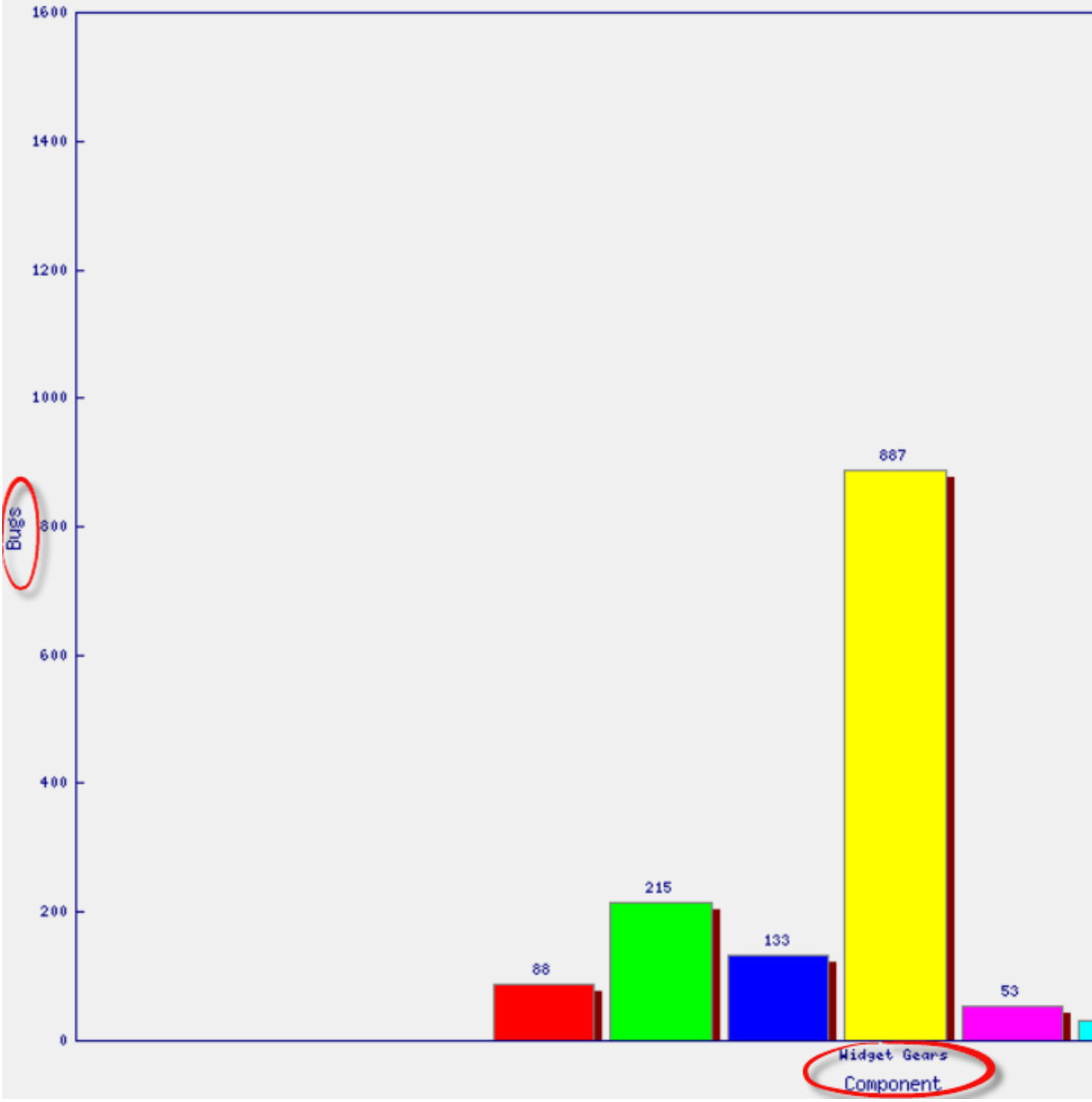
Create Graphical Reports

Graphical reports are one way to view the current state of the bug database. You can run reports either through an HTML table or graphical line/pie/bar-chart-based one. The idea behind graphical report in Bugzilla is to define a set of bugs using the standard search interface and then choosing some aspect of that set to plot on the horizontal and vertical axes. You can also get a 3-dimensional report by choosing the option of "Multiple Pages".

Reports are helpful in many ways, for instance if you want to know which component has the largest number of bad bugs reported against it. In order to represent that in the graph, you can select severity on X-axis and component on Y-axis, and then click on generate report. It will generate a report with crucial information.



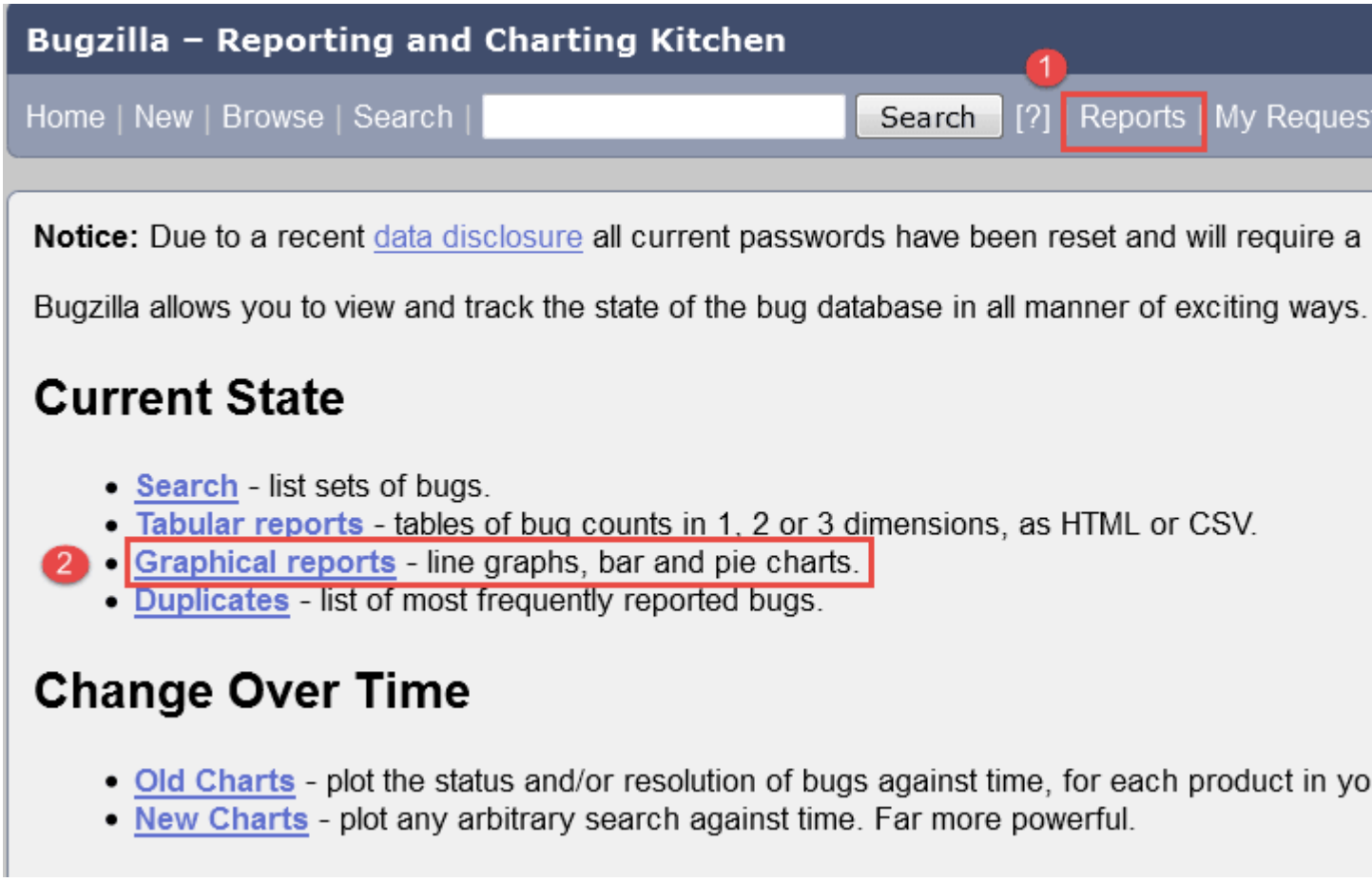
The graph below shows the Bar chart representation for the Bugs severity in component **"Widget Gears".** In the graph below, the most severe bug or blockers in components are 88 while bugs with normal severity is at top with 667 number.



Likewise, we will also see the line graph for **%complete Vs Deadline**

Step 1) To view your report in a graphical presentation,

- Click on Report from Main Menu
- Click on the Graphical reports from the given option



Step 2) Let's create a graph of **% Complete Vs Deadline**

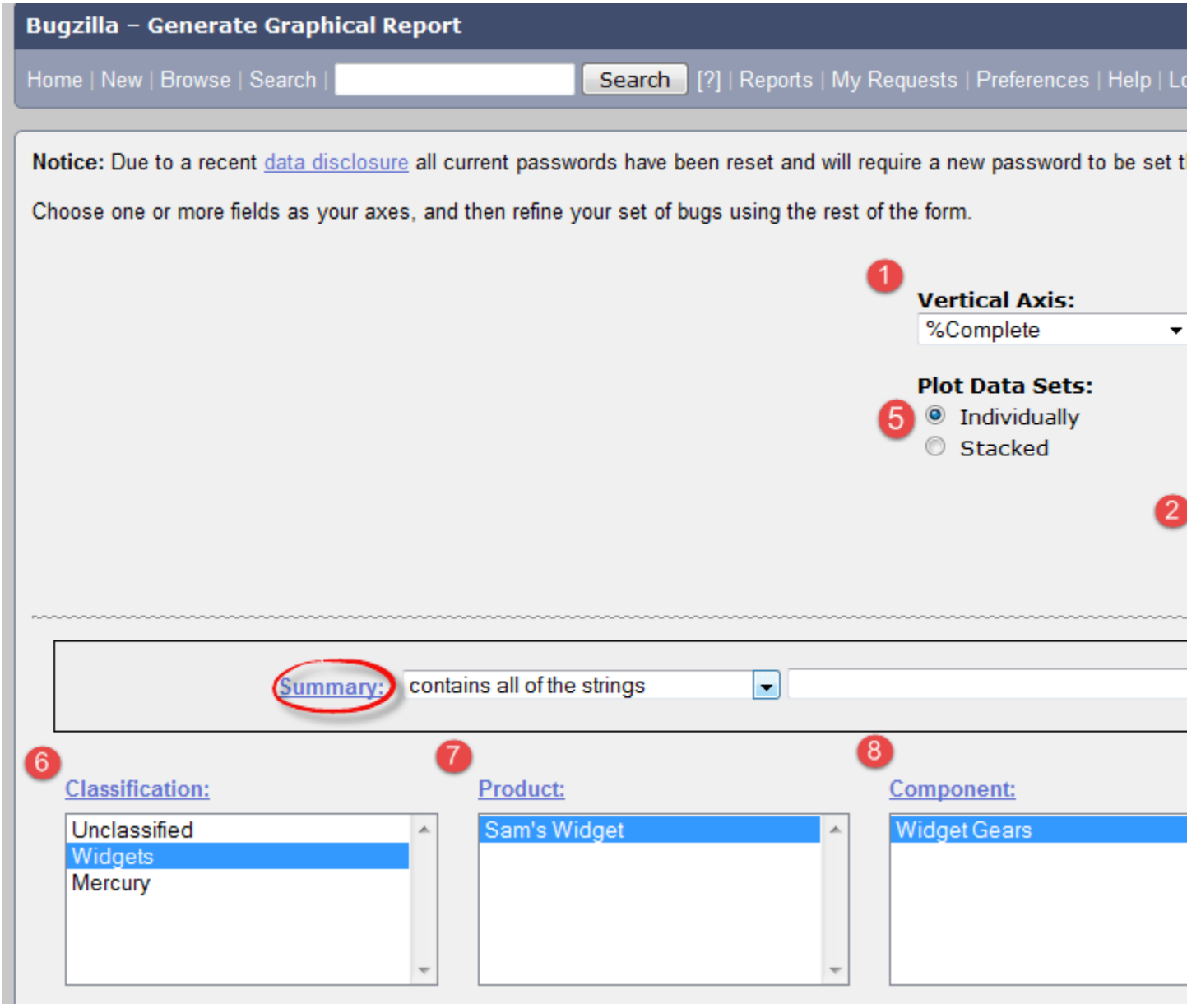
In here on the vertical axis we chose **% Complete** and on our horizontal axis we chose **Deadline**. This will give the graph of amount of work done in percentage against the set-deadline.

Now, set various option to present reports graphically

1. Vertical Axis
2. Horizontal Axis

3. Multiple Images

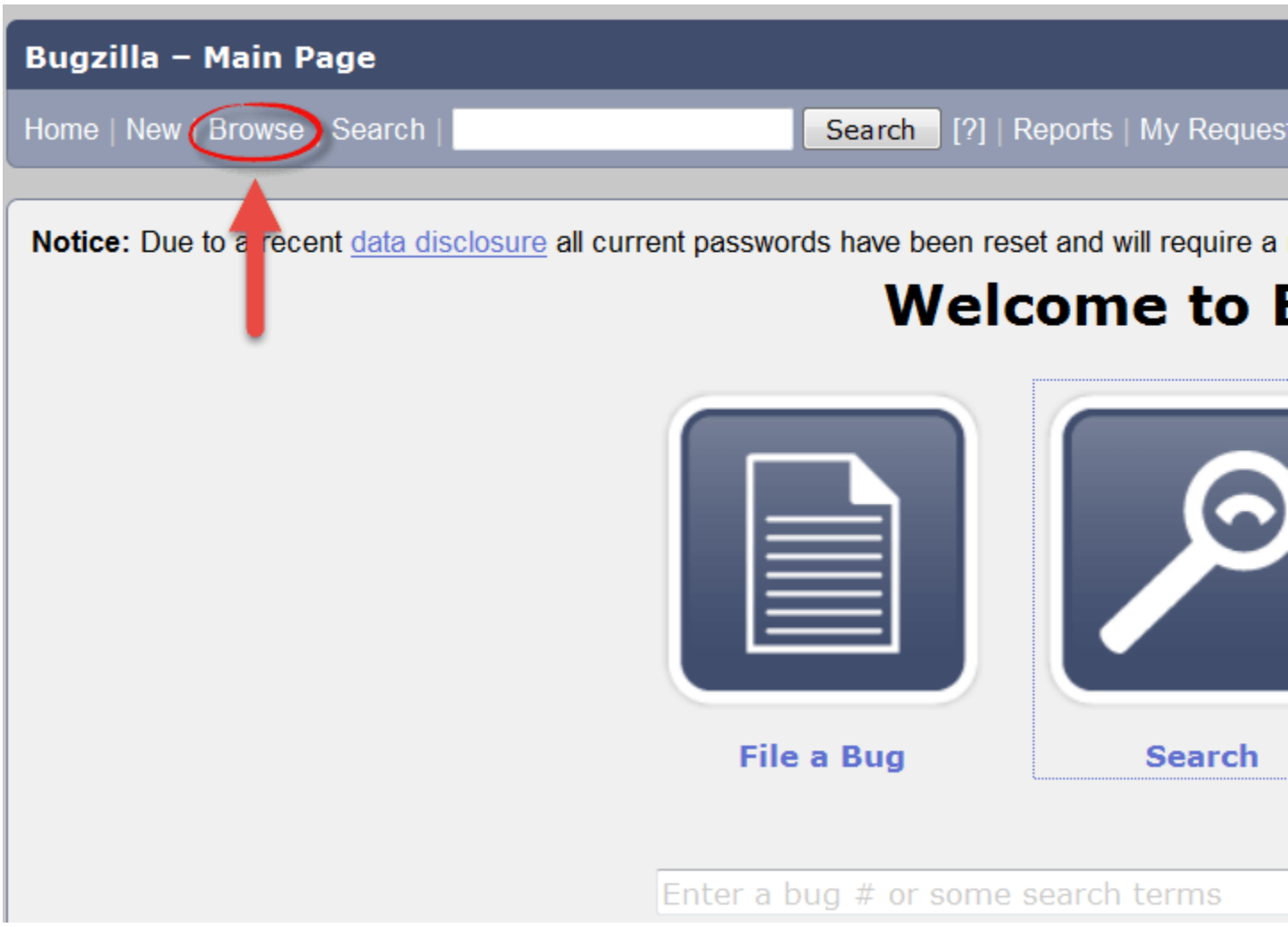
4. Format- Line graph, Bar chart or Pie chart
5. Plot data set
6. Classify your bug
7. Classify your product
8. Classify your component
9. Classify bug status
10. Select resolution
11. Click on generate report



The image of the graph will appear somewhat like this

Browse Function

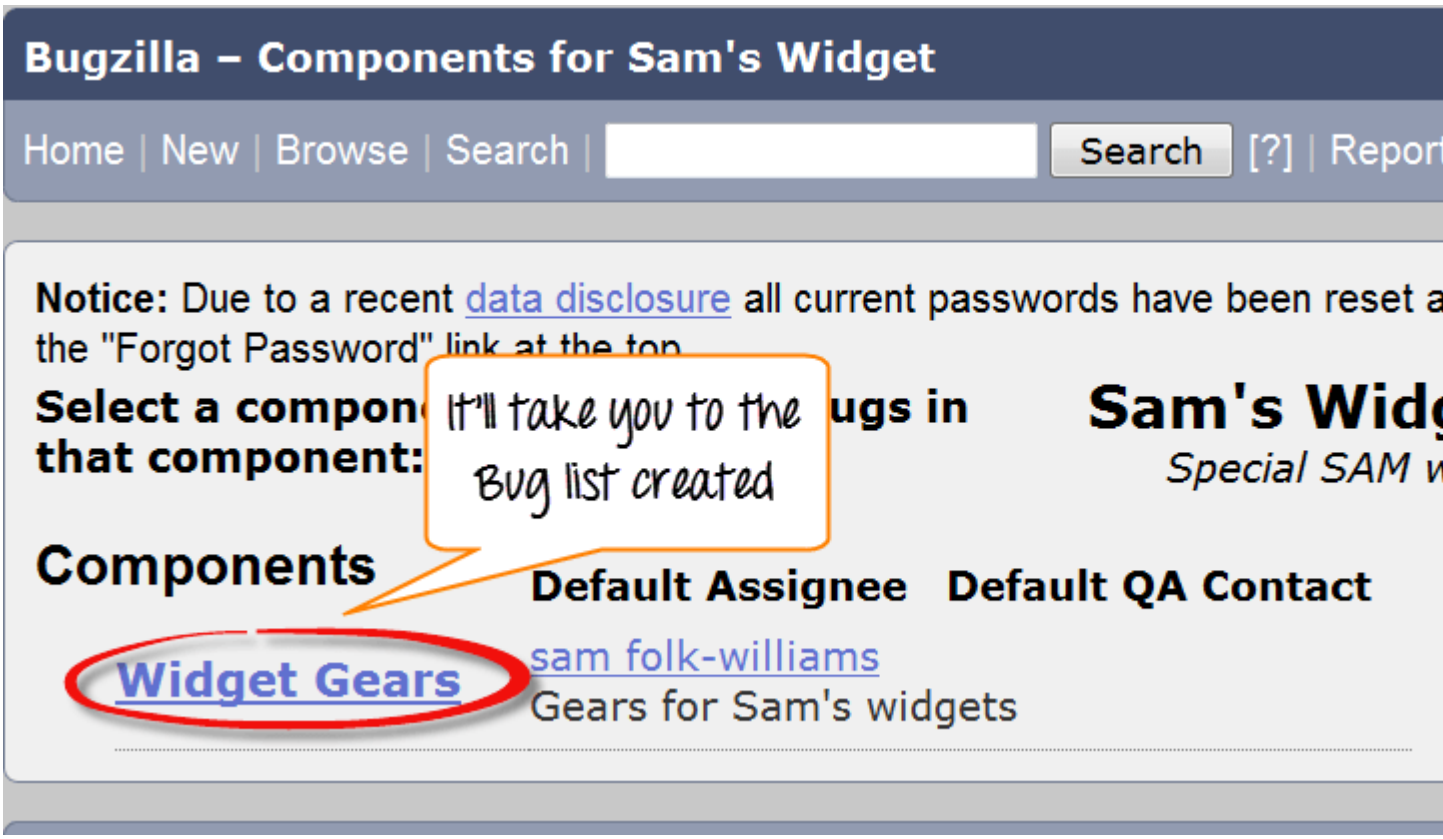
Step 1) To locate your bug we use browse function, click on **Browse** button from the main menu.



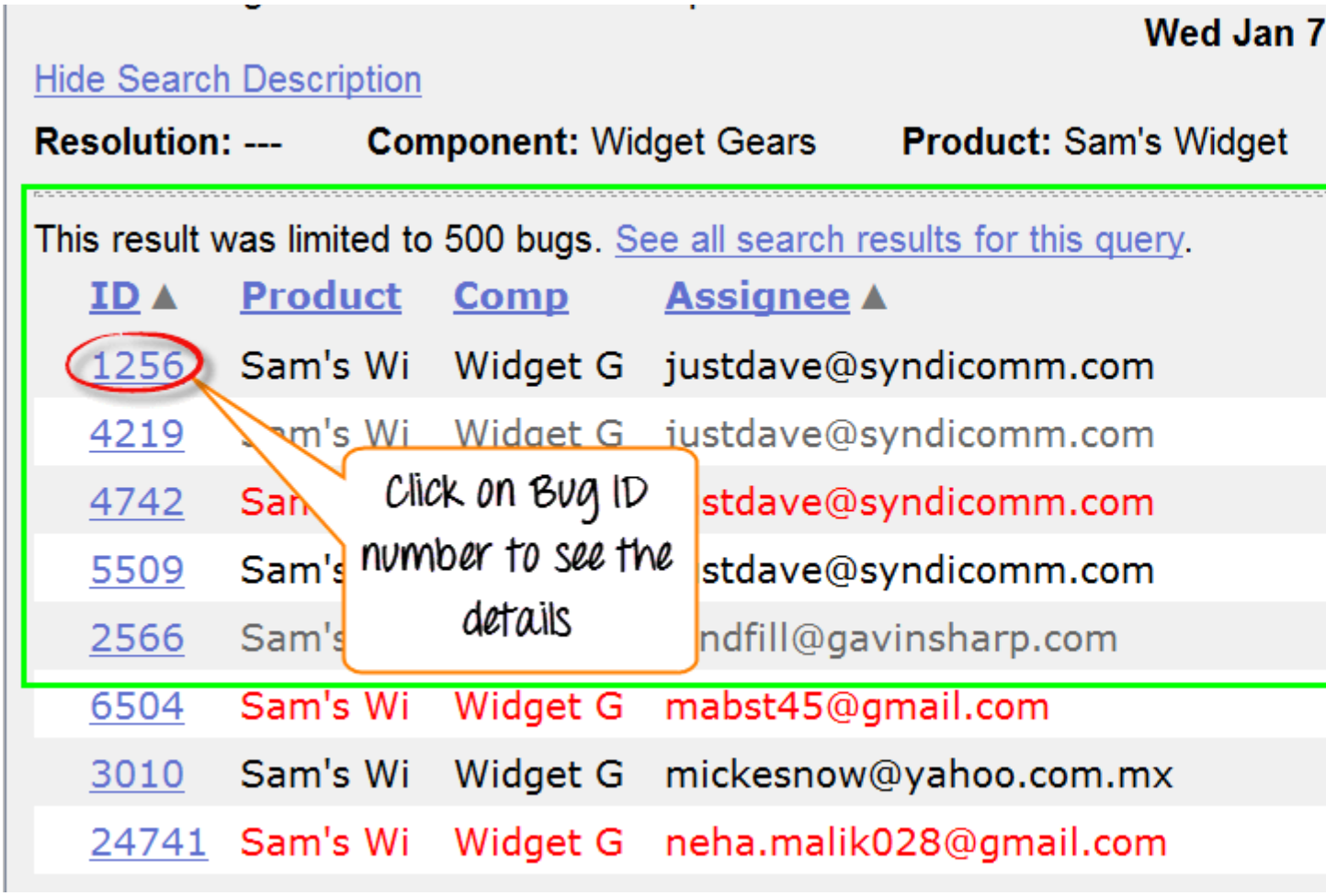
Step 2) As soon as you click on browse button a window will open saying **"Select a product category to browse"** as shown below, we browse the bug according to the category.

- After clicking the browse button
- Select the product "Sam's Widget" as such you have created a bug inside it

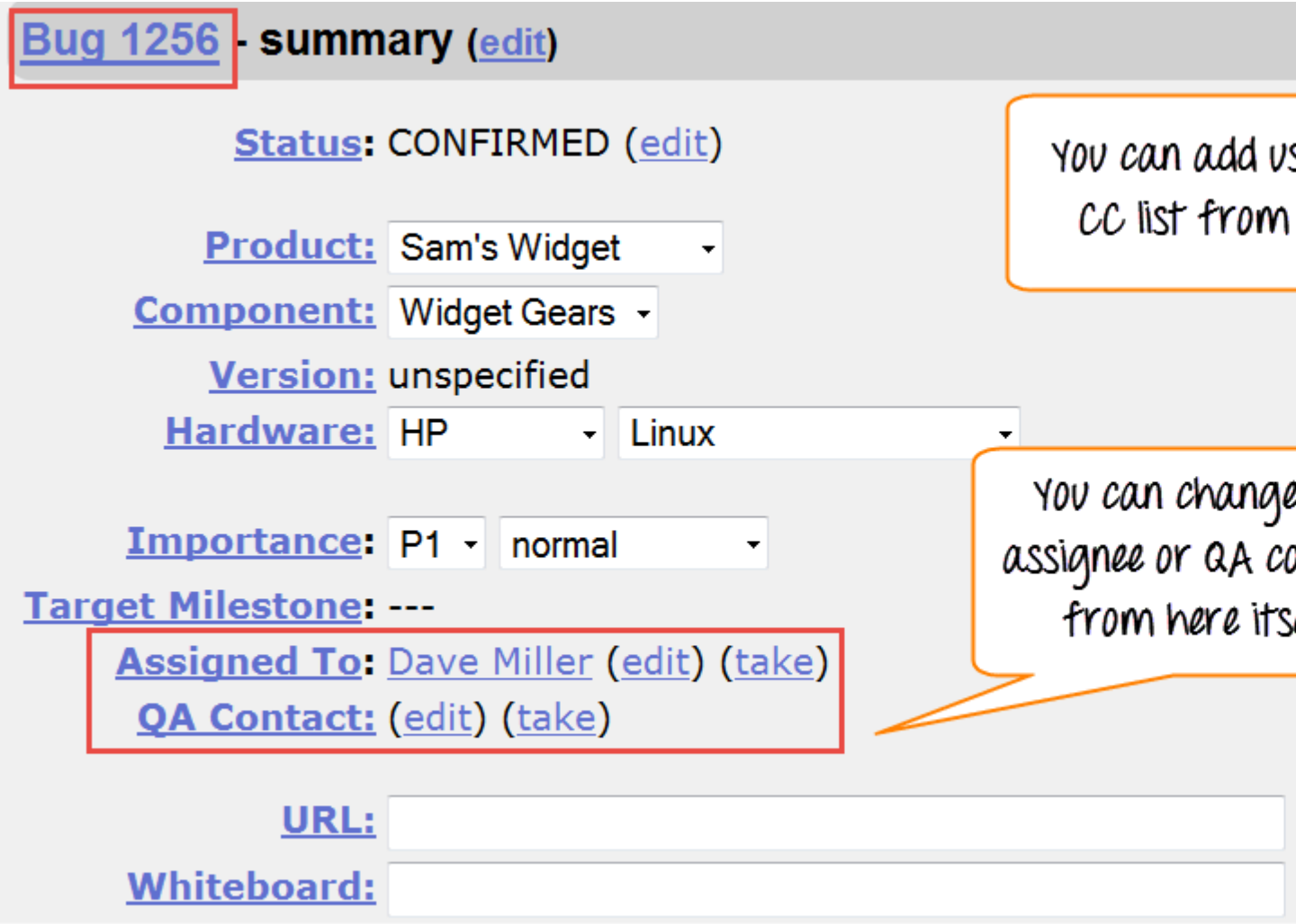
Step 3) It opens another window, in this click on component **"widget gears"**. Bugzilla Components are sub-sections of a product. For instance, here our product is **SAM'S WIDGET** whose component is **WIDGET GEARS**.



Step 4) when you click on the component, it will open another window. All the Bugs created under particular category will be listed over-here. From that Bug-list, choose your Bug#ID to see more details about the bug.



It will open another window, where information about your bug can be seen more in detail. In the same window, you can also change the assignee, QA contact or CC list.



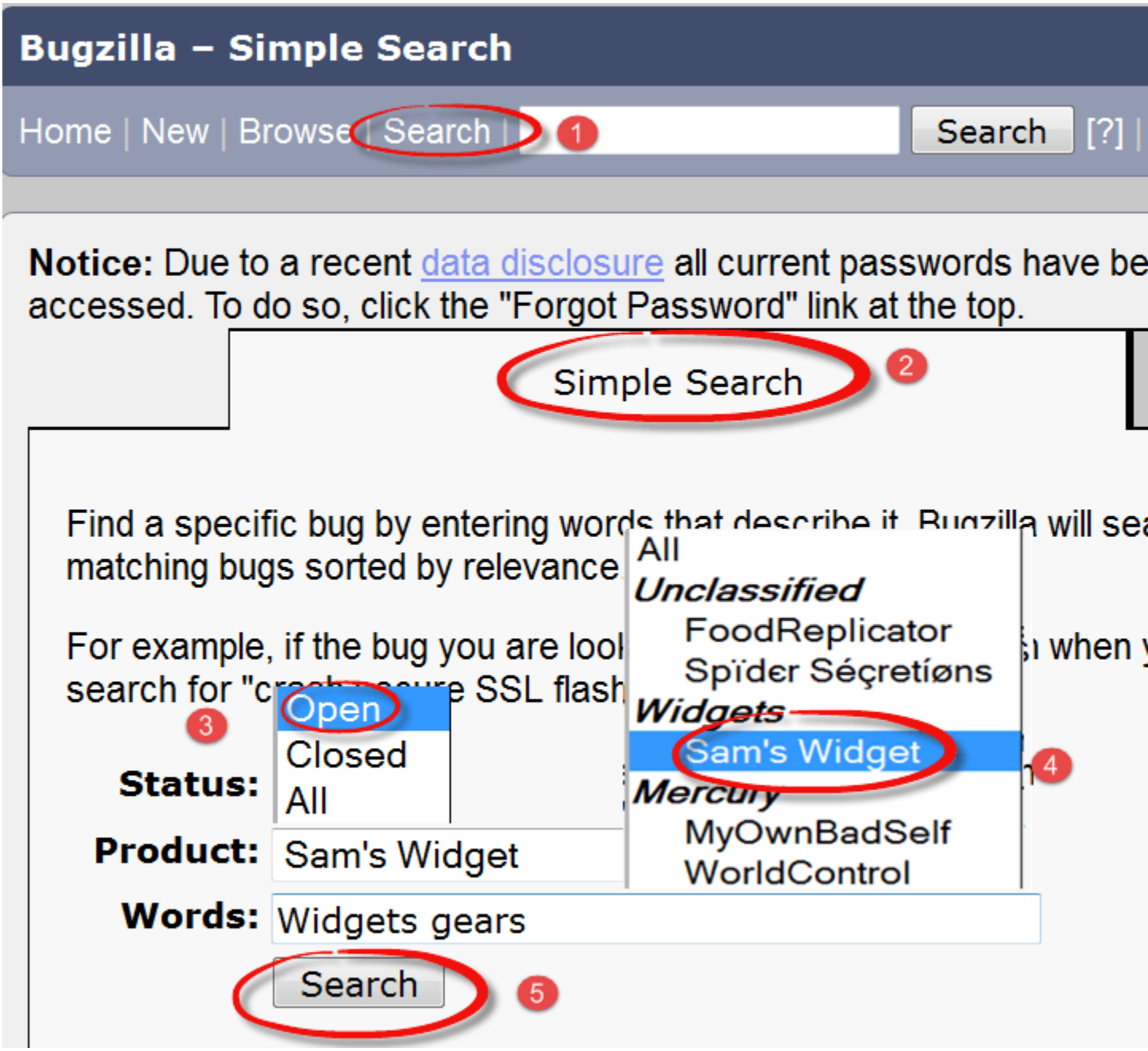
How to use Simple search option in BugZilla

Bugzilla provides two ways of searching bugs, they are **Simple Search** and **Advance Search**methods.

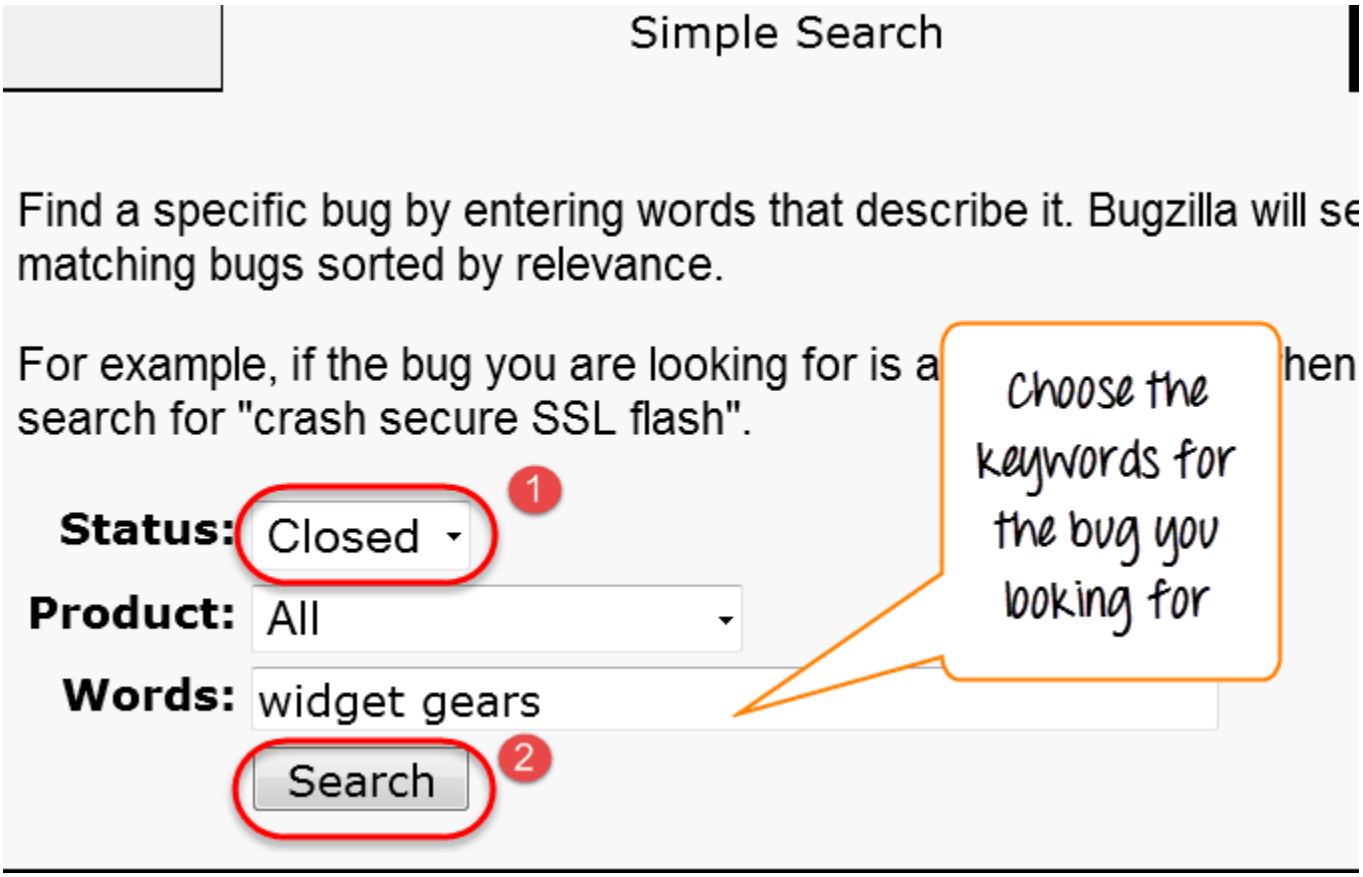
Step 1) We will first learn the **"Simple Search"** method. Click on search button from the main menu and then follow these steps

1. Click on "Simple Search" button
2. Choose the status of the Bug – choose Open if you are looking the bug in Open status and closed for bug in closed status
3. Choose your category and component, and you can also put keywords related to your bug

4. Click on the search



Step 2) Here we will search for both option **open** and **closed** status, first we have selected closed status for bug and clicked search button.



For closed status, it fetched 12 bugs.

12 bugs found.

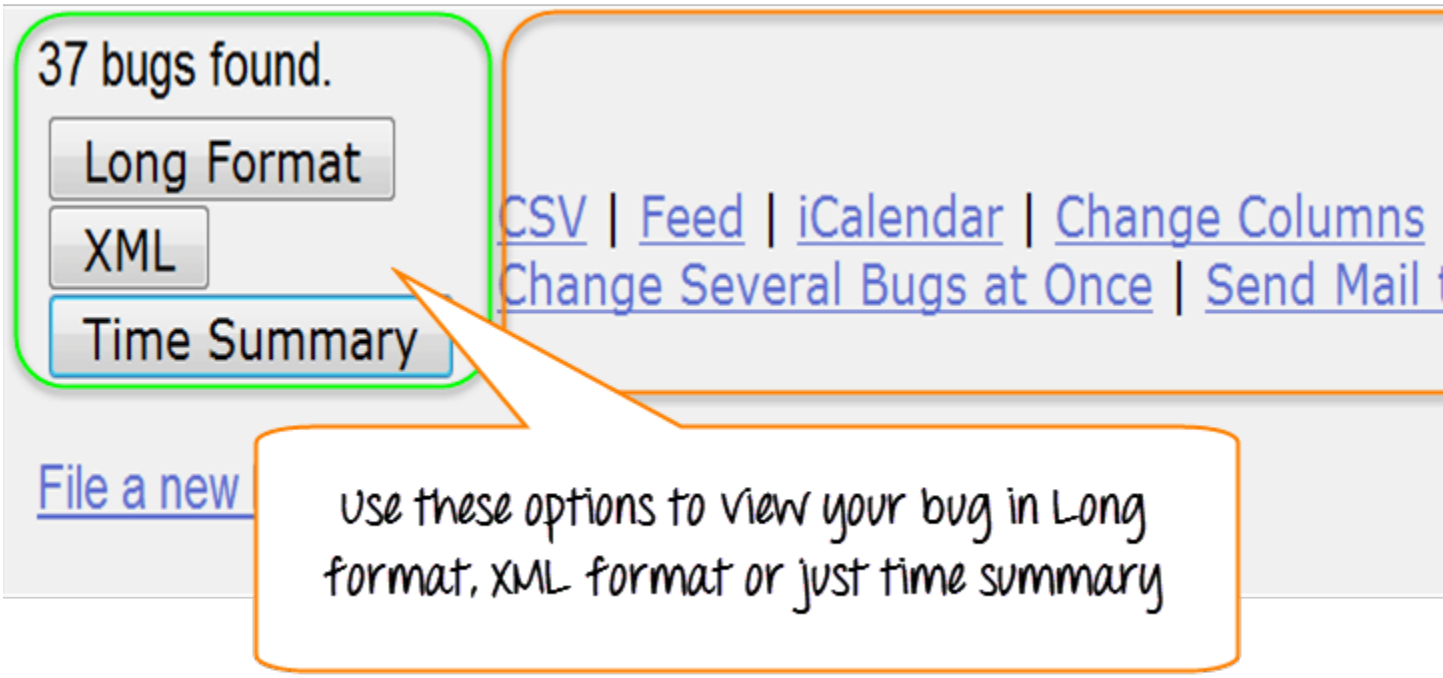
ID	Comp	Assignee	Status
12412	Widget G	sam.folkwilliams@gmail.com	RESO
2998	WeatherC	peter.rutherford@cnh.com	VERI
15235	WeatherC	tara@bluemartini.com	RESO
4297	Widget G	justdave@syndicomm.com	RESO
6312	Widget G	sam.folkwilliams@gmail.com	RESO

Step 3) Likewise we have searched for Open status as well, and it has fetched 37 bugs related to our queries.

37 bugs found.

ID	Product	Comp	Assignee
12431	Sam's Wi	Widget G	sam.folkwilliams@gmail.com
12561	Sam's Wi	Widget G	sam.folkwilliams@gmail.com
7359	Sam's Wi	Widget G	raydevereaux@bc.com
8907	Sam's Wi	Widget G	sam.folkwilliams@gmail.com

Also, at the bottom of the screen you have various options like how you want to see your bug - **an XML format, in Long format or just Time summary**. Apart from that you can also use other option like **send mail to bug assignee, change several bugs at once or change column of the screen,** etc.



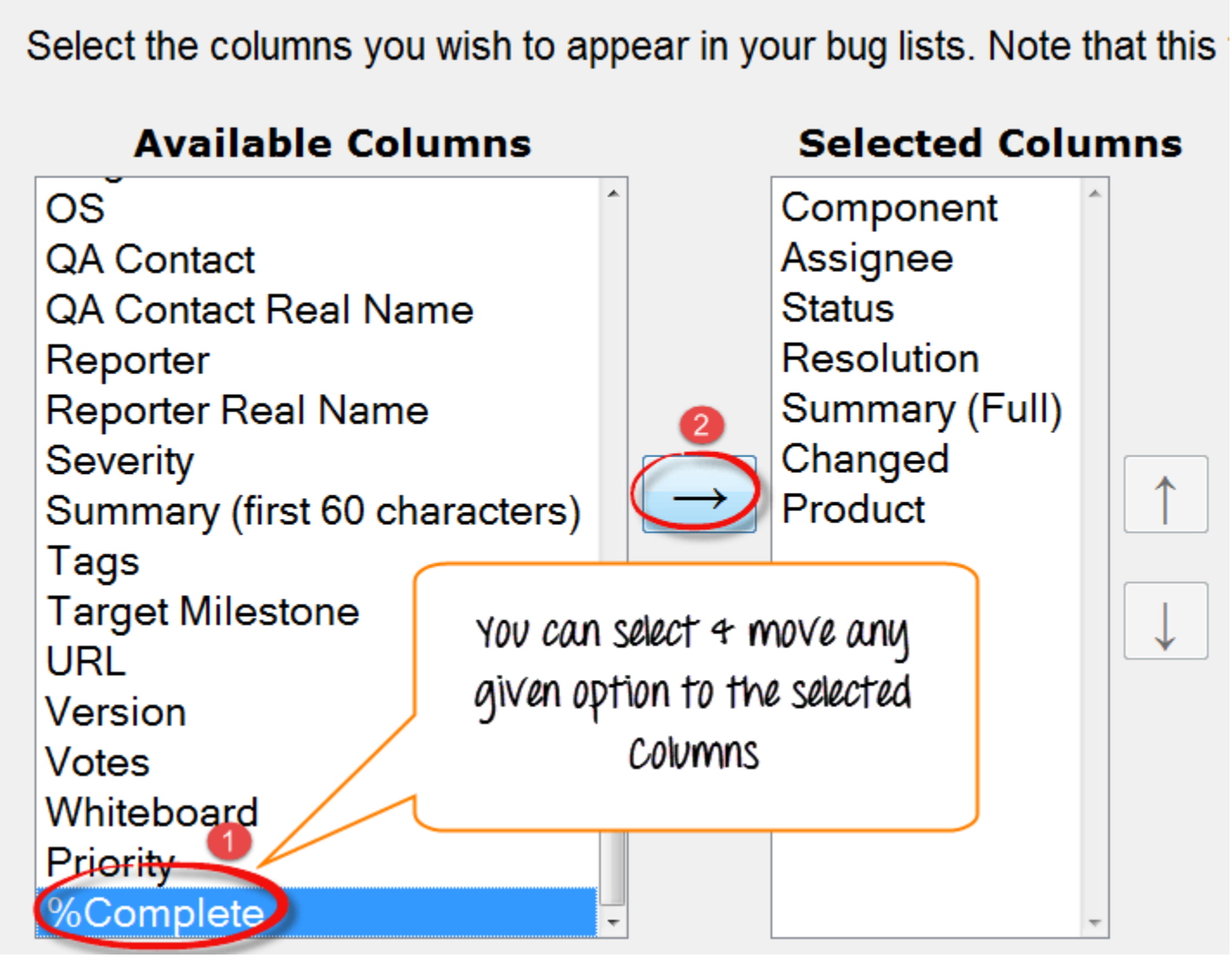
In next step, we will demonstrate one of this function **change column of the screen**, through which we will learn how to add or remove the column to the existing column.

How to add or remove column to default search screen

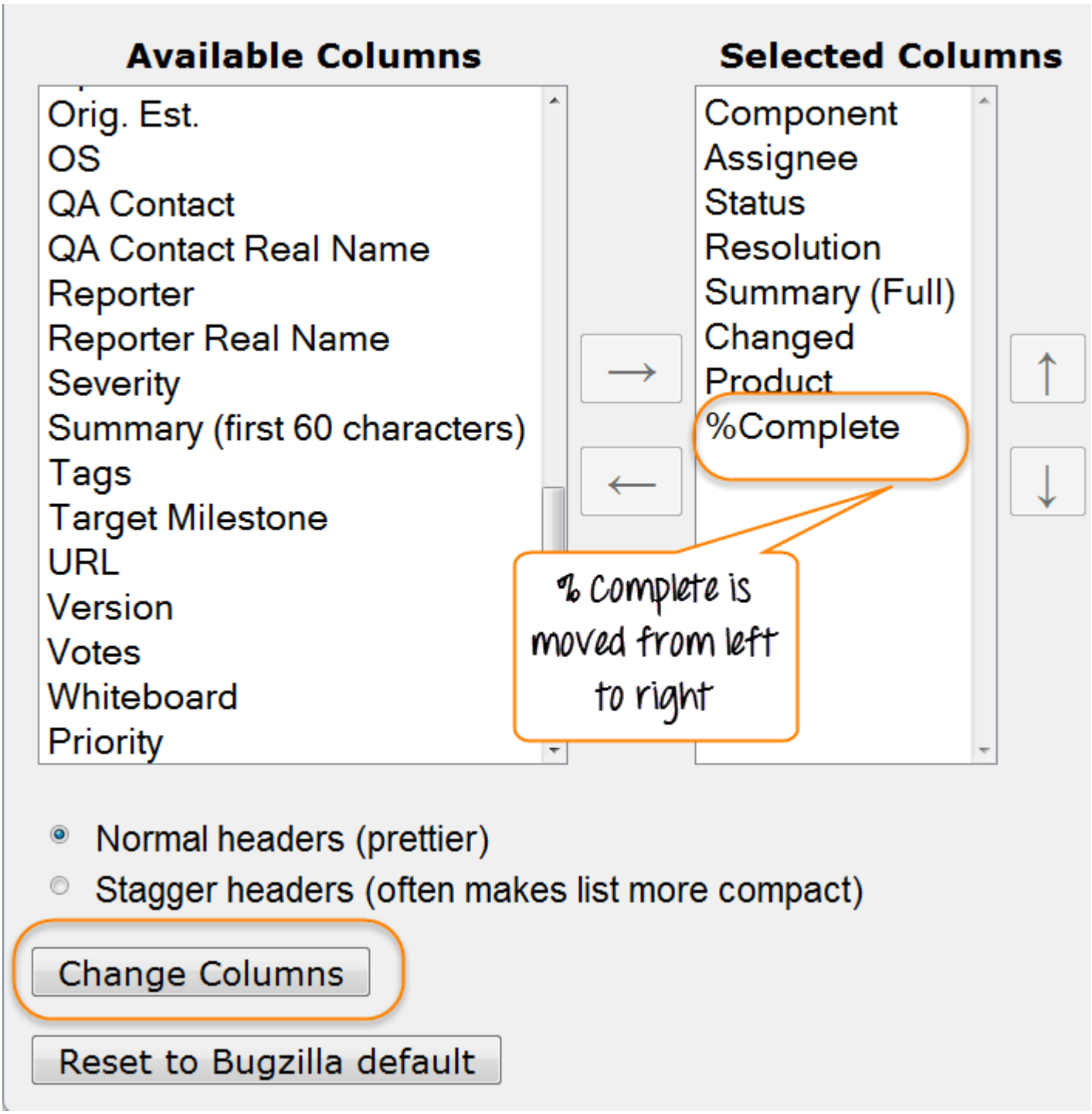
Step 1) Click on the **Change Column** as shown in above screen-shot. It will open a new window where you have to follow these steps.

- Select any given option from the column you want to appear in the main screen - here we have selected **% complete**
- Click on the **arrow button**, it will move % complete column from → **Available Column** to the **Selected column**

These steps will move the selected column from left to right.



The % complete is moved from left to right as shown below, and once we click on **change column**it will appear in the main screen



Before- Search result screen before using "Change Column" option-

- There is no % complete column appears in search screen result as shown below

ID	Product	Comp	Assignee
12431	Sam's Wi	Widget G	sam.folkwilliams@gmail.com
12561	Sam's Wi	Widget G	sam.folkwilliams@gmail.com
7359	Sam's Wi	Widget G	raydevereaux@bc.com
8907	**Sam's Wi**	**Widget G**	**sam.folkwilliams@gmail.com**

After- Search result screen after using "Change Column" option

- You can see **% complete** column added to the extreme right in the existing column in the main screen, which was not their previously.

37 bugs found.

ID	Comp	Assignee	Status
12431	Widget G	sam.folkwilliams@gmail.com	IN_P
12561	Widget G	sam.folkwilliams@gmail.com	CONF
7359	Widget G	raydevereaux@bc.com	IN_P
8907	**Widget G**	**sam.folkwilliams@gmail.com**	**CONF**

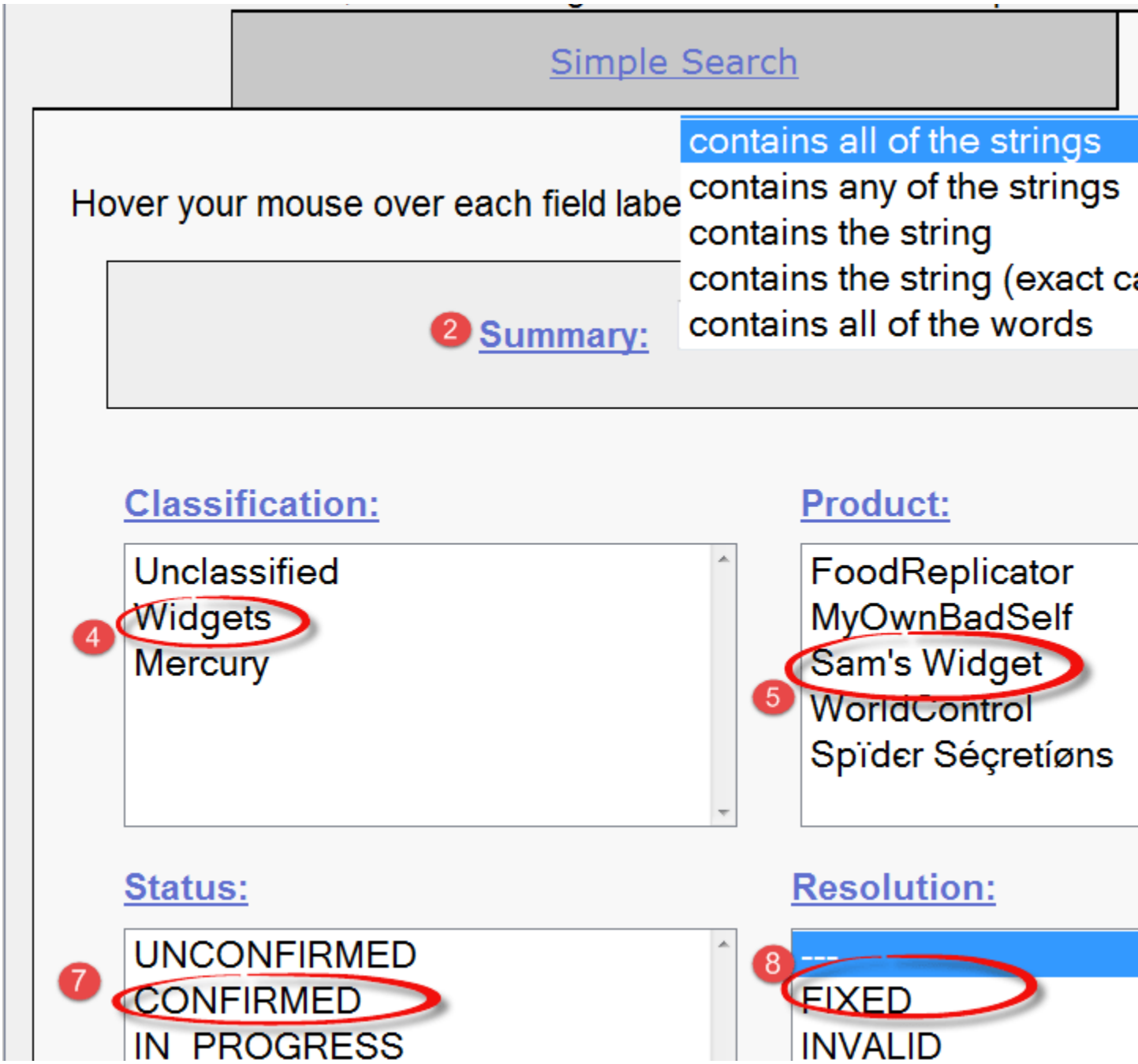
NOTE: Likewise you can remove or add any column you want.

How to use Advance Search in BugZilla

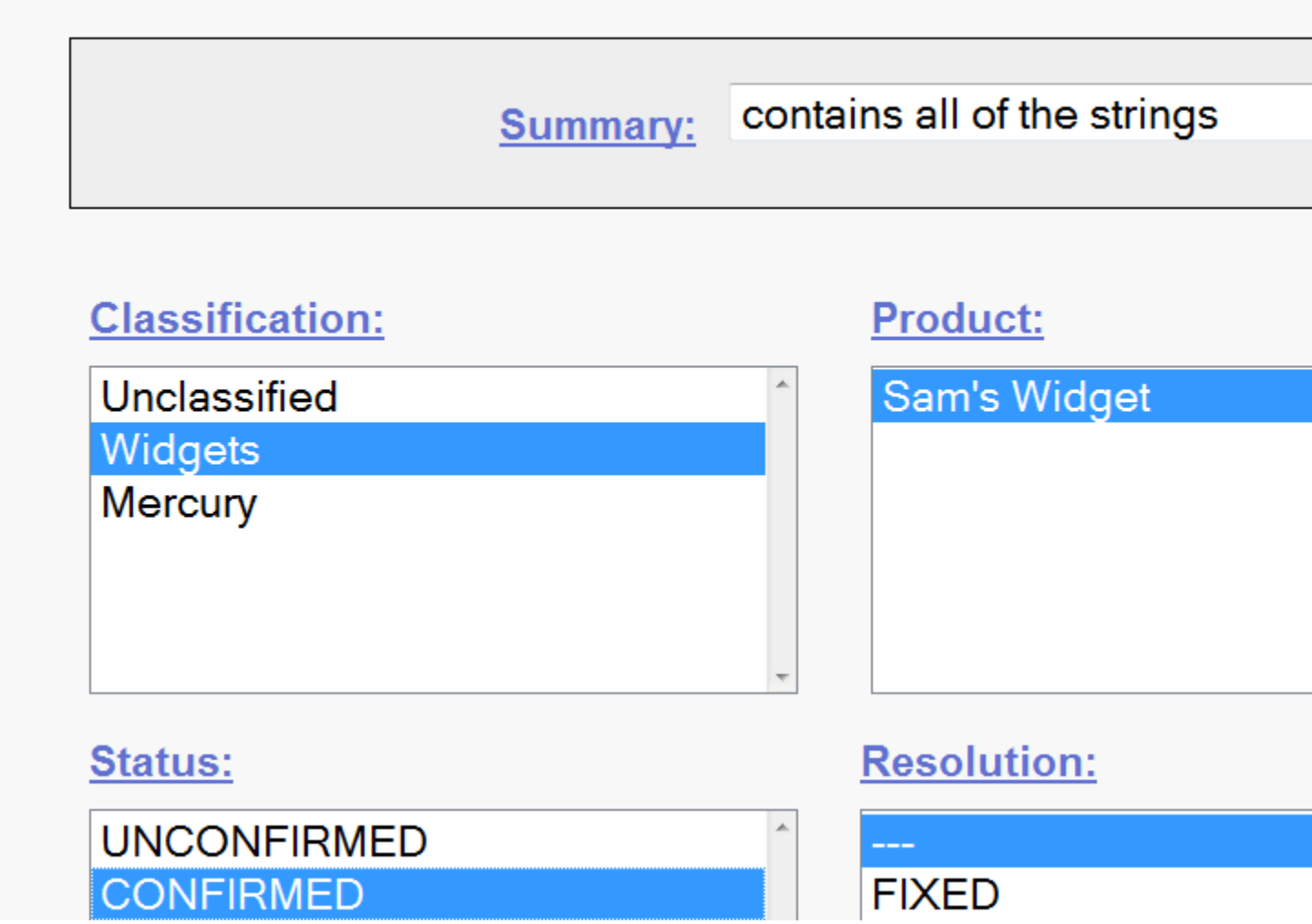
Step 1) After Simple search we will look into **Advanced Search** option for that you have to follow the following steps.

1. Click on advanced search option

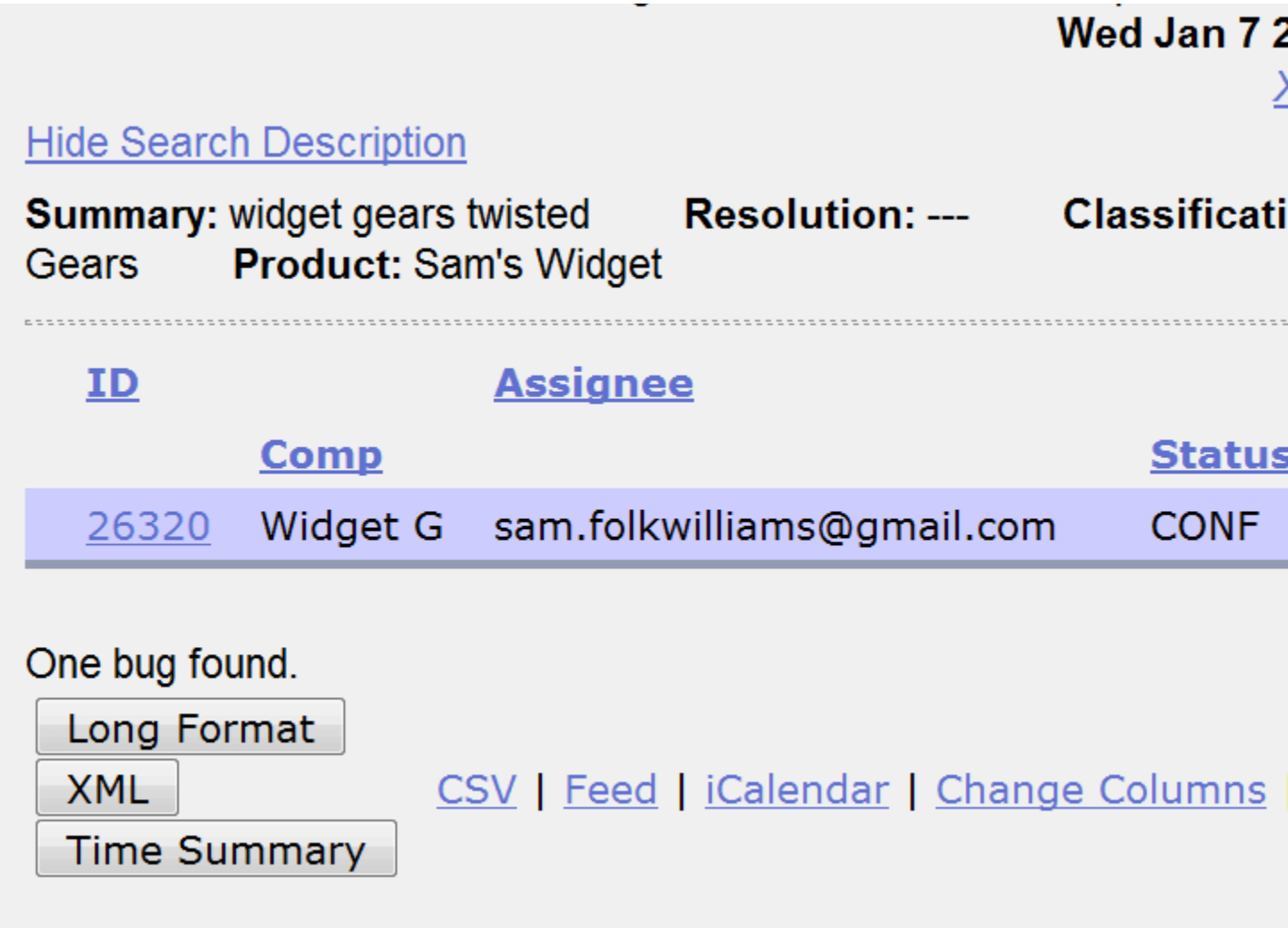
2. Select option for summary, how you want to search
3. Enter the keyword for your bug- for example, **Widget gears twisted**
4. Select the category of your Bug under classification, here we selected Widget
5. Choose your product under which your Bug was created- Sam's Widget
6. Component- Widget gears
7. Status- Confirmed
8. Resolution



Step 2) Once you select all the option, click on search button. It will detect the bug you created



The advance search will find your bug, and it will appear on the screen like this



How to use preferences in BugZilla

Preferences in Bugzilla is used to customize the default setting made by Bugzilla as per our requirement. There are mainly five preferences available

- General Preferences
- E-mail Preferences
- Saved Searches
- Account Information
- Permissions

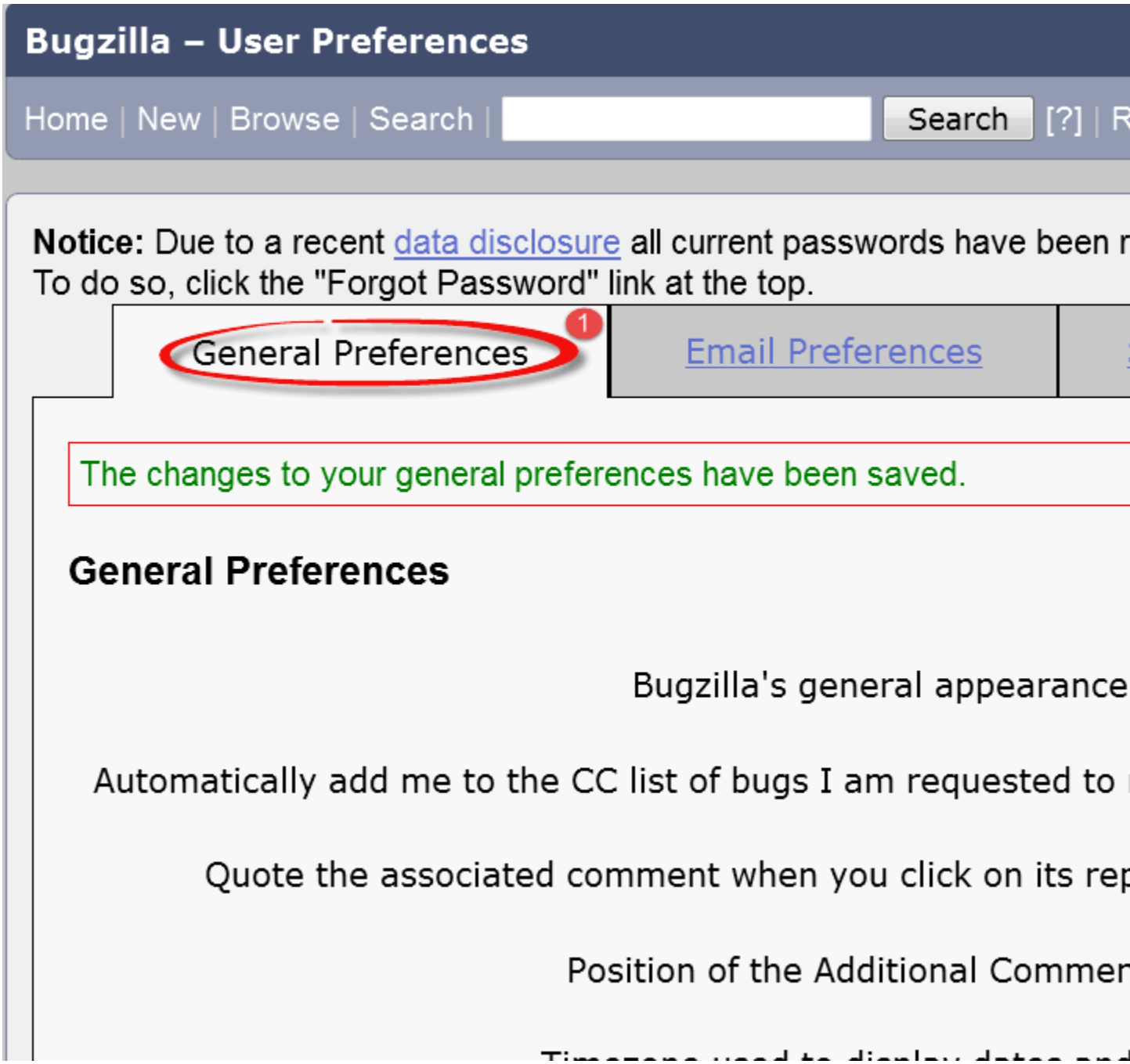
General Preferences

For **general preferences**, you have various option like **changing Bugzilla general appearance, position of the additional comment box, automatically add me to cc,** etc. Here we will see how to change the general appearance of the Bugzilla.

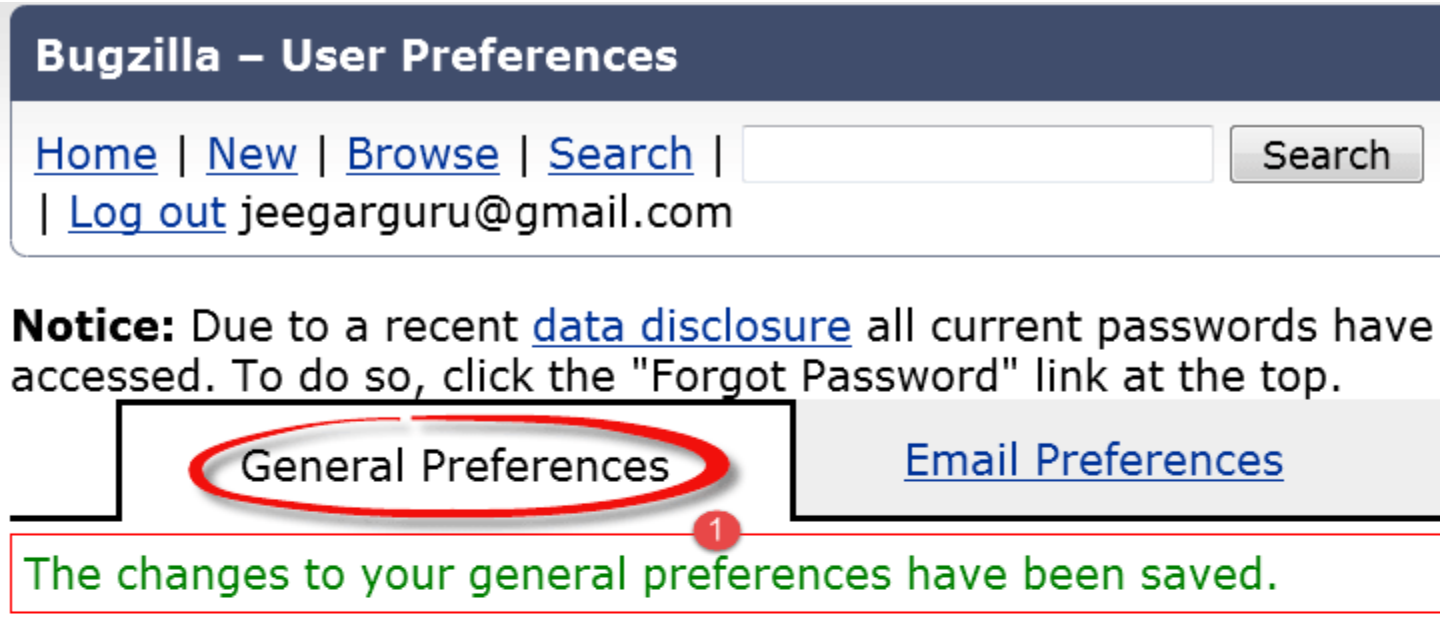
There are many changes you can do which are self-explanatory, and you can choose the option as per your requirement.

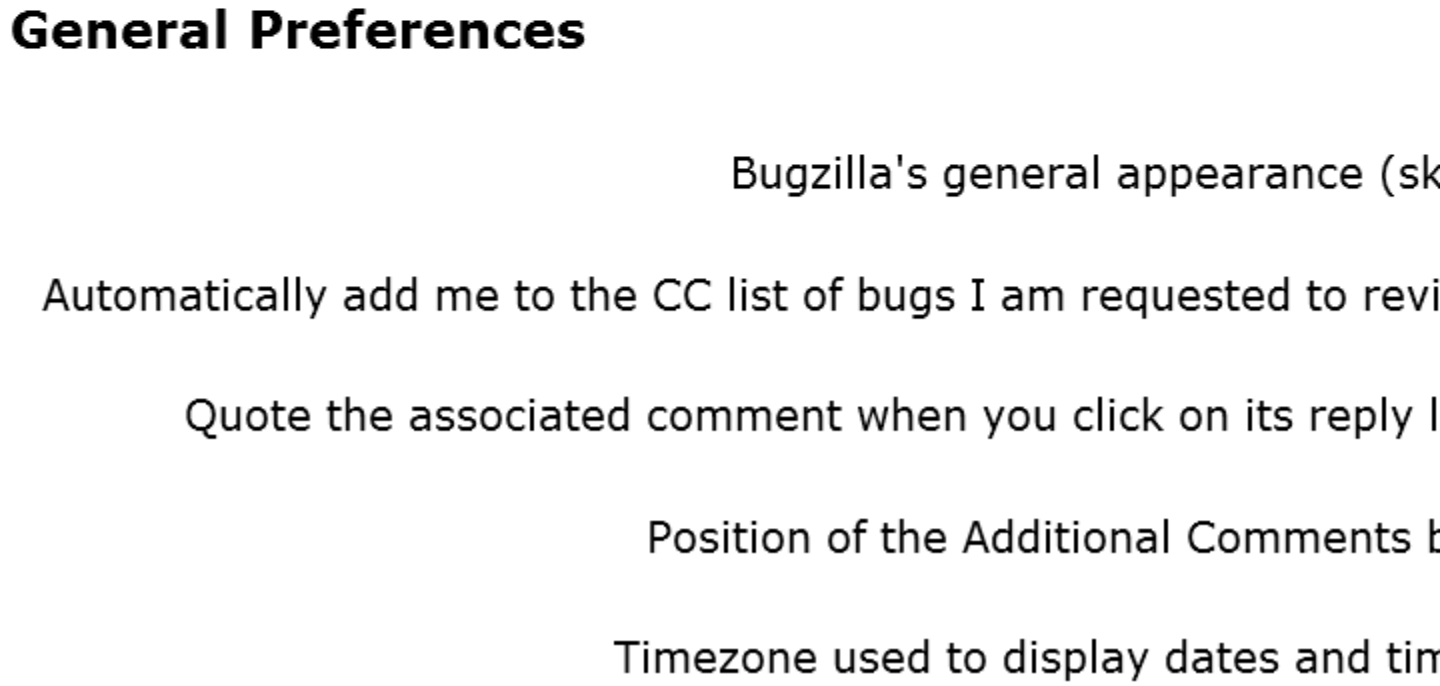
Step 1)

- To set the background Skin of Bugzilla
- Go to Bugzilla general preference (Skin)
- Select the option you want to see as a change and submit the change (Dusk→Classic)
- A message will appear on the window saying changes have been saved, as soon as you submit the changes



After the skin preference is changed to Classic from Dusk, the back-ground color of the screen appears white





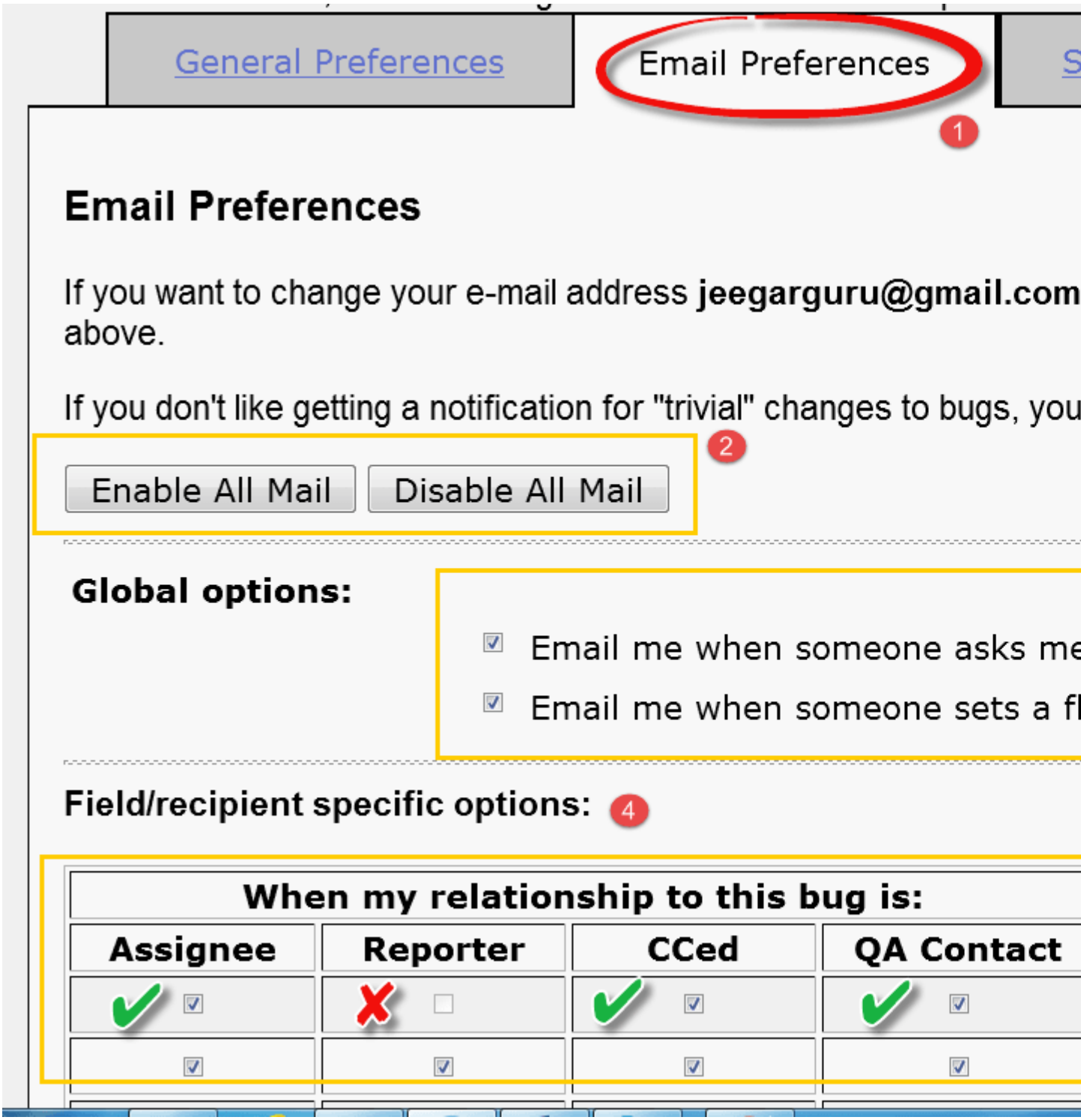
Likewise, for other default settings changes can be done.

E-mail preferences

E-mail preferences enable you to decide how to receive the message and from whom to receive the messages.

Step 1) To set the e-mail preferences

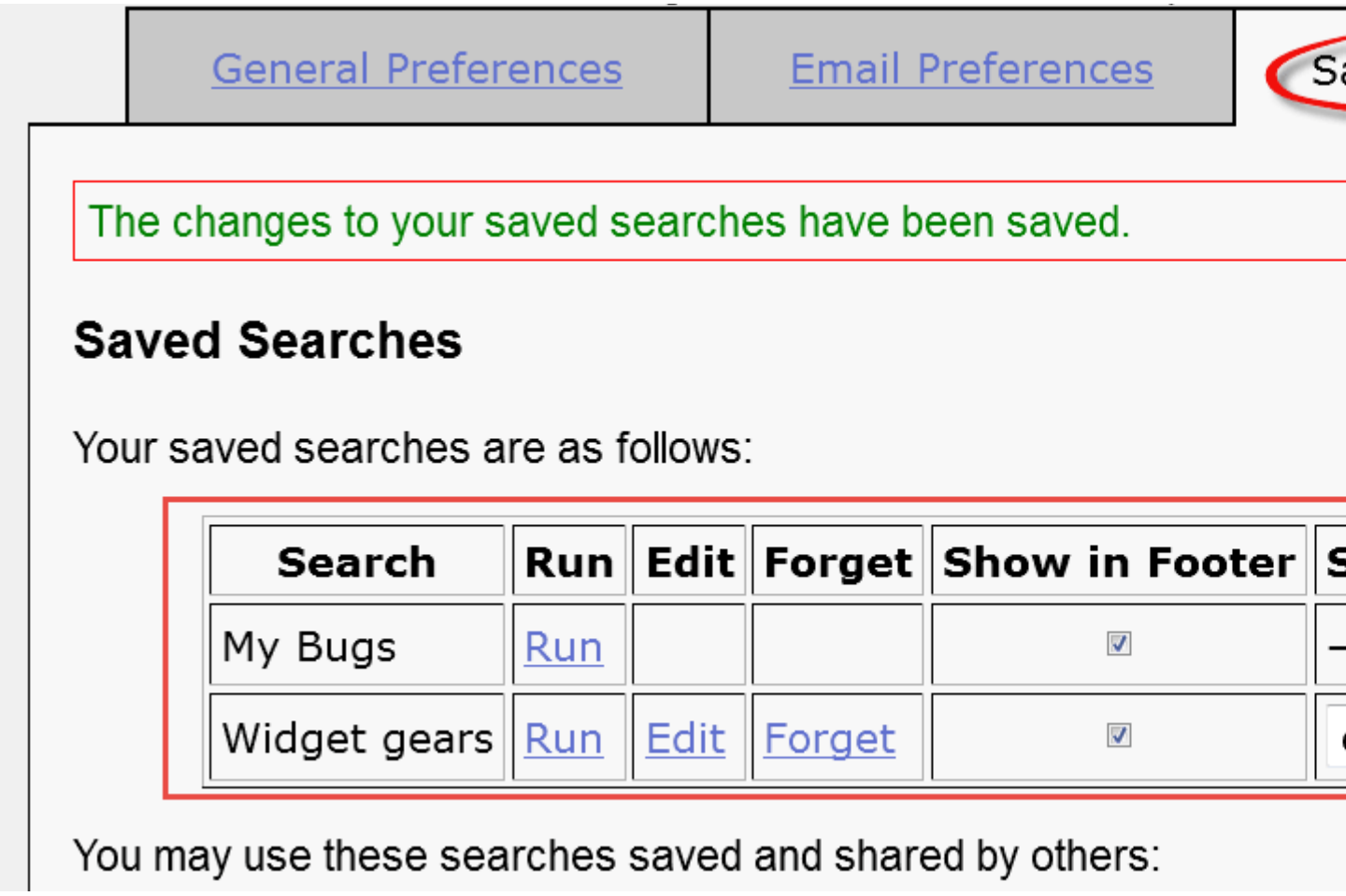
1. Click on e-mail services
2. Enable or disable the mail to avoid receiving notification about changes to a bug
3. Receiving mail when someone asks to set a flag or when someone sets a flag you asked for
4. When and from whom you want to receive mail and under which condition. After marking your option at the end, submit the changes.



Saved Searches Preference

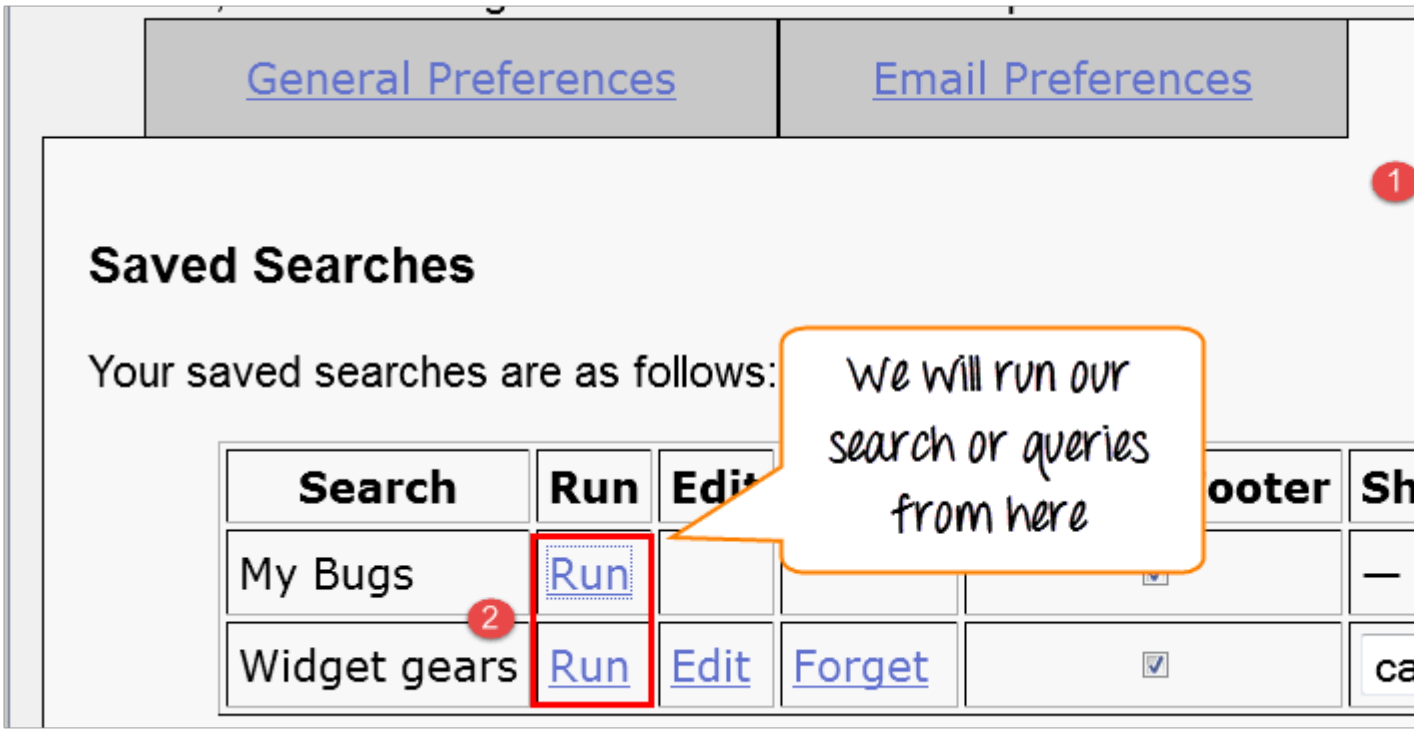
Saved searches preference gives you the freedom to decide whether to share your bug or not to share.

Step 1) Click on saved searches, it will open window with the option like **editbugs, don't share, canconfirm, etc.** Choose the option as per your need.

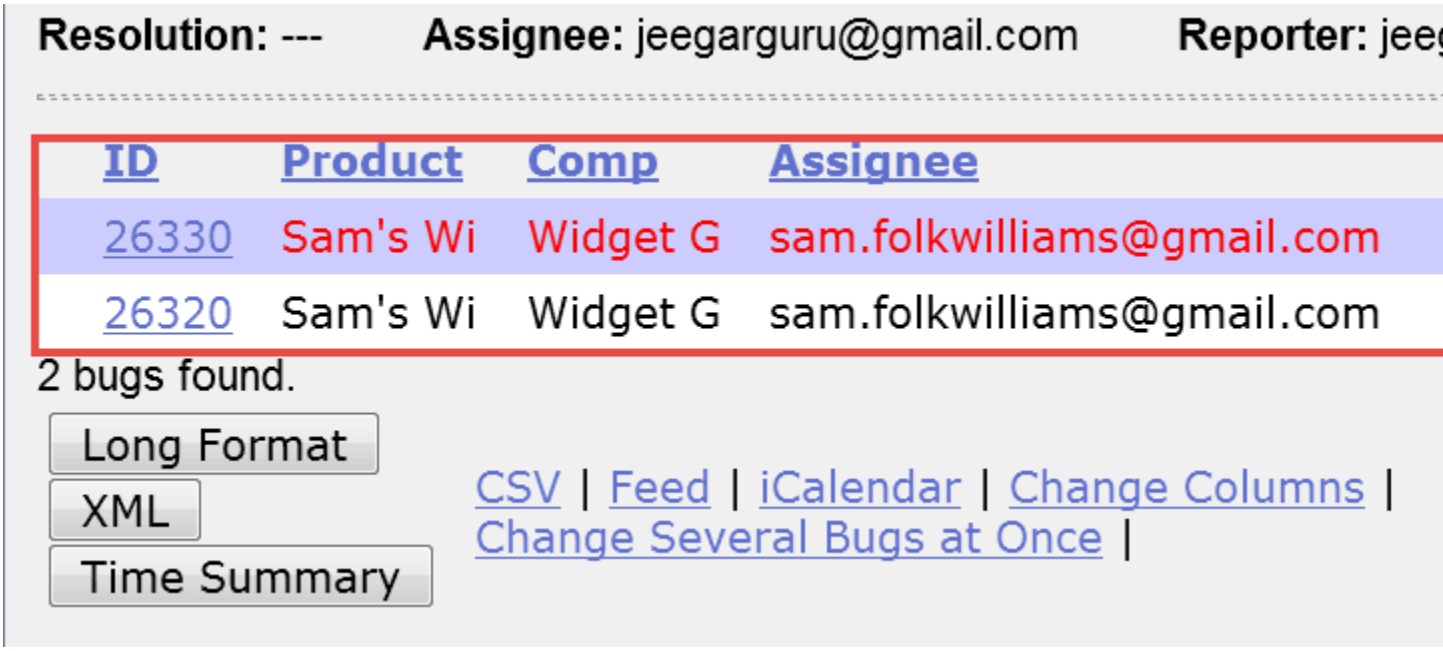


Step 2) We can run our bug from **"Saved Searches".**

- Go to Saved Searches under preference
- Click on the **"Run"** button



As soon as you run your search from Saved Searches it opens your bug as shown below



Step 3) In the same window we can also choose specific users with whom we want to share the search by marking or unmarking the checkbox against the users

Search	Shared By
123	Amrita <osmosys.dici@gmail.com
All Mercury	Ben Schultz (SofTechnics) <ben.schultz@softechnics.com>
ami	Ami <ami_nahmani@walla.com>
Bug1	Aruna <aruna_trimurti@yahoo.co

Home | New | Browse | Search | Search

My Bugs | Widget gears
All Mercury | ami
| 26320 | sams widget

Both these users can edit our bug