《数据库系统原理》课程设计

系统实现报告

题目名称： BugFade系统

学号及姓名：\_\_\_\_\_\_\_\_14061081\_吴宁\_\_

\_\_\_\_\_\_\_\_14061060\_刘颖\_\_\_

2016年 11 月 23 日

**一．系统功能需求分析**

1 需求描述

* 1. 系统概述

大中型软件的开发过程中往往会出现很多Bug，记录这些Bug的提交，修复和关闭是一件极其重要的事情。大型软件的开发需要系统的进行Bug的管理。基于此我们设计了BugFade。BugFade是一个缺陷追踪系统，协助用户在软件开发和维护的过程中建立完善的BUG追踪系统。

* 1. 功能描述

BugFade主要功能包括对bug的增加，删除，查询和状态的修改。系统对不同身份的用户提供不同的功能。一个用户可以同时在多个项目中扮演不同的角色。项目组的所有成员，包括管理人员，开发人员和测试人员，都可以看到各自的贡献值。

* + 1. 游客

游客可以看到目前网站上所有的项目。进入该项目界面后可以提出自己在使用过程中发现的问题。游客提交的bug需要经过项目组测试人员的测试。如果该bug经过测试人员的复核被接受，游客可以看到bug处理的进度。

* + 1. 管理猿

1、创建项目

管理人员可以在系统中创建某个项目。

2、接收申请

管理人员可以接受用户的申请，从而向自己创建的项目中添加管理人员，测试人员和开发人员。

3、人员分组

团队人员可以粗略的分为测试人员和开发人员。管理人员还可以对测试人员和开发人员进行更加详细的分组，比如某几个人员负责项目中某个模块的开发。

* + 1. 测试猿

1、复核bug

测试人员可以查看游客提出的问题，并进行复核。复核通过确认项目存在该问题后将该问题记录下来。测试人员还可以复核开发人员修复后的bug并决定该问题是否已成功解决。

2、分配bug

测试人员测试出项目中存在的某个问题后可以在系统中将该问题分配给开发人员中对应的组。

3、bug状态

测试人员可以查看并修改自己提交/复核的所有bug当前的状态

4、分配优先级

测试人员可以给bug设定优先级来让开发人员选择解决bug的先后顺序。

* + 1. 开发猿

1、领取bug

开发人员可以在自己的组内领取bug。领取了之后即相当于开发人员要对此bug负责到底。Bug被领取之后状态就自动从newAdd改变为doing。

2、查看bug

开发人员可以看到项目中所有的bug，但只能对自己领取的bug提出相应的解决意见，被采纳后积分可以加一。

## 二．系统功能结构设计

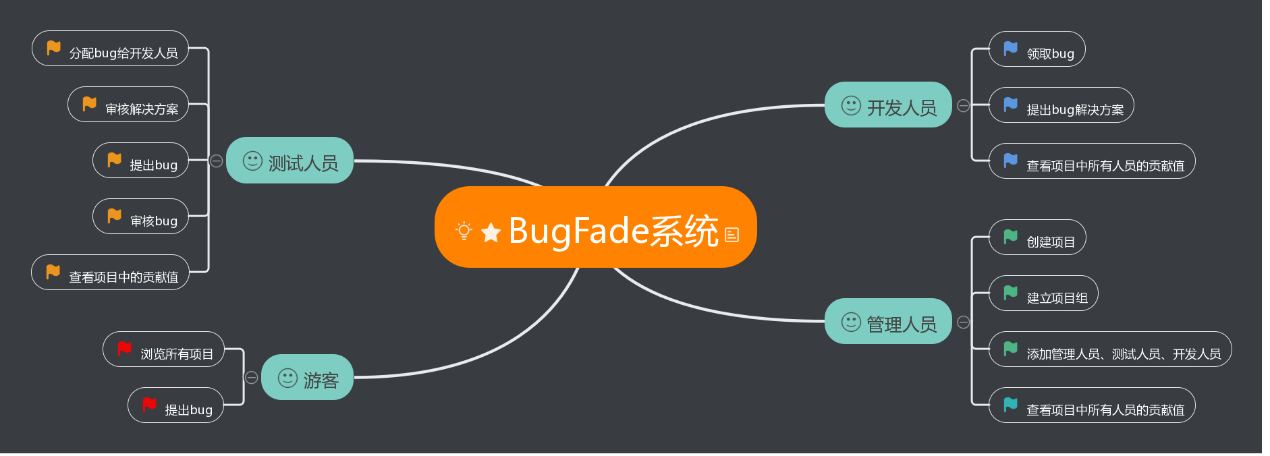


Figure 1 项目功能结构图

**三、数据库基本表的定义**

**1、项目总表**

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 长度 | 描述信息 |
| ID | **Varchar** | **20** | **项目编号** |
| CreaterID | **Varchar** | **20** | **项目创建者编号** |
| createdTime | **Date** |  | **项目创建时间** |
| dueTime | **Date** |  | **项目截至时间** |
| Description | **Varchar** | **1024** | **项目描述** |
| Name | **Varchar** | **20** | **项目名** |

**2、 项目申请表**

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 长度 | 描述信息 |
| aID | **Varchar** | **20** | **申请者ID** |
| Position | **Varchar** | **20** | **申请的职位** |
| Username | **Varchar** | **20** | **申请者名字** |
| Description | **Varchar** | **20** | **申请者的备注信息** |
| Time | **Date** |  | **申请时间** |

**3、 项目bug核实表**

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 长度 | 描述信息 |
| ID | **Varchar** | **20** | **BugID** |
| CheckerID | **Varchar** | **20** | **核实人ID** |
| CreaterID | **varchar** | **20** | **创建者ID** |

**4、 项目bug分组表**

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 长度 | 描述信息 |
| BugID | **Varchar** | **20** | **Bug编号** |
| groupName | **Varchar** | **20** | **Bug分配组别名** |

**5、 bug信息表**

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 长度 | 描述信息 |
| ID | **Varchar** | **20** | **Bug编号** |
| Description | **Varchar** | **15** | **Bug描述信息** |
| State | **Varchar** | **15** | **Bug状态** |
| CreatedTime | **Date** |  | **创建时间** |
| CreatedID | **Varchar** | **20** | **创建者ID** |
| Priority | **Int** | **11** | **Bug优先级** |
| Name | **varchar** | **20** | **Bug名** |
| SolverID | **varchar** | **20** | **解决者ID** |

**6、bug领取表**

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 长度 | 描述信息 |
| BugID | **varchar** | **20** | **Bug编号** |
| resID | **Varchar** | **20** | **Bug领取者编号** |
| getTime | **Date** |  | **领取时间** |

**7、项目组别表**

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 长度 | 描述信息 |
| GroupId | **Varchar** | **20** | **组别编号** |
| Name | **Varchar** | **20** | **组别名** |

**8、项目成员组别表**

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 长度 | 描述信息 |
| GroupName | **varchar** | **20** | **组别名** |
| ID | **Varchar** | **20** | **成员ID** |

**9、项目人员管理信息表**

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 长度 | 描述信息 |
| MemberID | **Varchar** | **20** | **项目成员编号** |
| Position | **Varchar** | **20** | **职位** |

**10、项目解决方案表**

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 长度 | 描述信息 |
| ID | **Varchar** | **20** | **项目解决方案编号** |
| Content | **Varchar** | **1000** | **解决方案内容** |
| BugID | **Varchar** | **20** | **Bug编号** |
| SolverID | **Varchar** | **20** | **解决者编号** |
| Accept | **Int** | **1** | **是否被测试人员接受** |

**11、用户关联项目表**

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 长度 | 描述信息 |
| ProjectID | **Varchar** | **20** | **项目编号** |
| projectName | **Varchar** | **20** | **项目名** |

**12、用户总表**

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 长度 | 描述信息 |
| ID | **Varchar** | **20** | **用户编号** |
| Name | **Varchar** | **20** | **用户名** |
| Password | **Varchar** | **20** | **密码** |
| Email | **Varchar** | **20** | **电子邮箱** |
| Reputation | **Int** | **10** | **声望** |

**四、触发器的定义与实现**

1 删除项目的触发器,将userXproject中的projectID为被删除项目ID的元组删除。

2 测试人员接受Bug的触发器:将该工程内的该Bug状态修改为doing

3删除项目人员之后的触发器，在membergroup中删掉该人员对应的表项

4修改解决方案状态之后的触发器:若解决方案通过，则bug的solverID变成提出方案者的ID，否则变为null，同时增加或减少提出解决方案者的积分。

实现(这里只列举触发器4的实现,后面的实现也都只举一个例子):

CREATE TRIGGER `projectSolutionTrigger` AFTER UPDATE ON `projectsolutions`

FOR EACH ROW BEGIN

if(1=new.accept) then

UPDATE projectbuginfo SET solverID = new.solverID WHERE ID =new.bugID;

if(new.accept!=old.accept) then

UPDATE users SET Reputation = Reputation+1 WHERE ID =new.solverID;

end if;

elseif (0 = new.accept) then

UPDATE project41buginfo SET solverID = 'null' WHERE ID =new.bugID;

if(new.accept!=old.accept) then

UPDATE users SET Reputation = Reputation-1 WHERE ID =new.solverID;

end if;

end if;

END

**五、存储过程的定义与实现**

1由于很多地方都需要统计表的长度，比如说统计网站内的用户数量，项目数量，由于mysql的存储过程不支持动态的表名，因此我不得不自学并使用了一些mysql里面的trick，最后将计算表长度的sql操作封装成了存储过程，只要得到了表名，就可以通过call projectNum('tableName')得到表的长度。

2 将表的更新放进了存储过程，一定程度上实现了代码的复用，传入表名，要修改的行的id，要更新到的值即可，不过只能对特定结构的表进行更新操作。

存储过程的实现(表长度计数):

delimiter $$

drop procedure if exists projectNum$$

create procedure projectNum( table\_id varchar(25) )

begin

set @temp\_query = 'drop temporary table if exists temp\_cursor\_table';

prepare pst from @temp\_query;

execute pst;

drop prepare pst;

set @temp\_table\_query='create temporary table temp\_cursor\_table ';

set @temp\_table\_query=concat( @temp\_table\_query, ' select ID from ' );

set @temp\_table\_query=concat( @temp\_table\_query, table\_id );

prepare pst from @temp\_table\_query;

execute pst;

drop prepare pst;

begin

declare done int default false;

declare stmt1 varchar(1024);

declare stmt2 varchar(1024);

declare count int;

declare id varchar(20);

declare getid cursor for

select ID from temp\_cursor\_table;

declare continue handler for not found set done = 1;

set count=0;

open getid;

fetch getid into id;

repeat

set count=count+1;

fetch getid into id;

until done end repeat;

close getid;

select count;

end;

end;

$$

delimiter ;

**六 视图的定义和实现**

**1**由于在设计的时候将表拆分地比较细，因此在查询数据的时候经常需要对两个表做连接操作，比如说查询某个人的组别和职位，这两个属性存在于不同的表中，因此需要对membermanage和membergroup进行连接操作，所以我们使用了视图来简化代码，将两个表合并为一个视图，这样只要一次查询就可以得到数据。

2 使用了视图将一些表的某些重要的字段，比如id等隐藏起来，比如说查询项目的时候只要找到项目的名字和内容就可以，因此创建者的id，项目的id可以用视图隐藏起来。

视图的实现(表的连接):

create  algorithm=merge  view

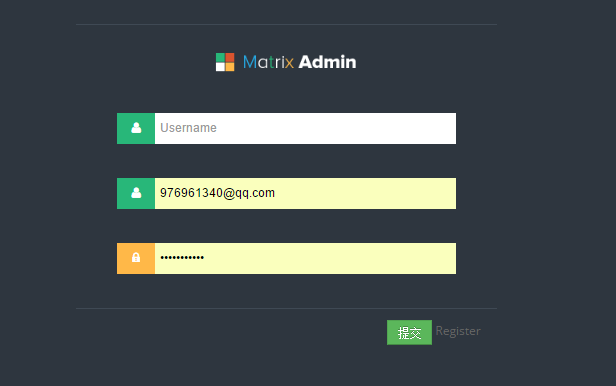
member(id,position,groupName) as [select](http://127.0.0.1/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html)

projectmembermanage.memberID,projectmembermanage.position,projectmembergroup.groupID fromprojectmembergroup,projectmembermanage

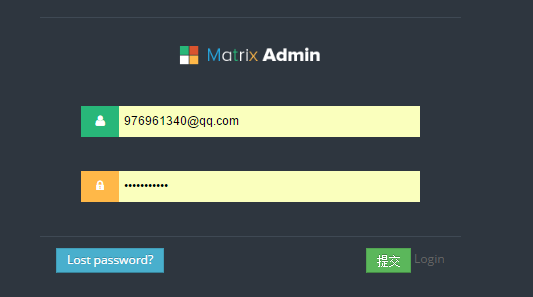
where projectmembergroup.name = projectmembermanage.memberID with local check option

**六、系统实现结果**

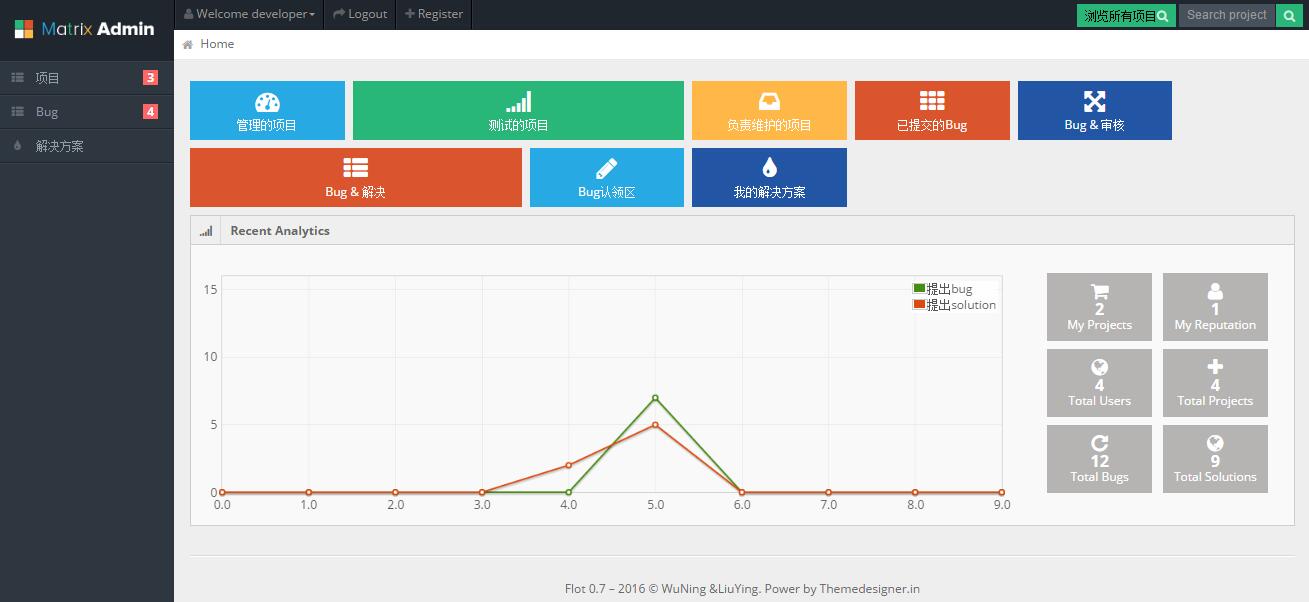
注册界面

****

登陆界面:

****

首页:

****

下方的折线图为最近十天提交的bug数量折线图和最近十天提交的解决方案数量的折线图，反映了用户最近的活跃程度。

折线图的右边是一些统计，包括当前用户参与的项目数量，当前系统内项目的数量，当前系统的用户总数，当前用户的声望，当前已提交的bug总数，当前系统内已有的解决方案总数。

点击管理的项目，测试的项目，负责维护的项目即可进入到当前用户作为管理人员参与的项目，作为测试人员参与的项目，作为开发人员参与的项目。大致内容如下:

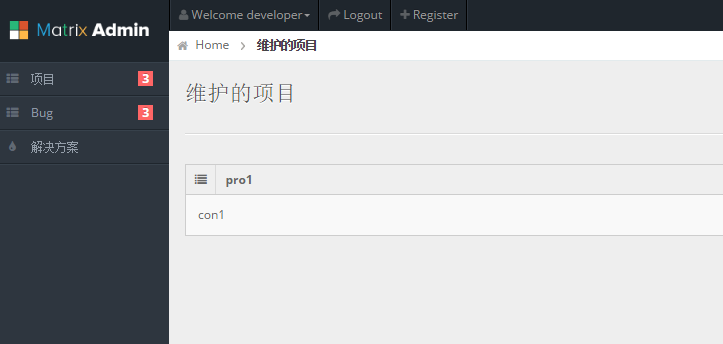
点击管理的项目，进入如下页面:



可点击创建新项目来新建一个由自己作为管理员的的项目，

也可以点击下方项目进入项目的详细界面

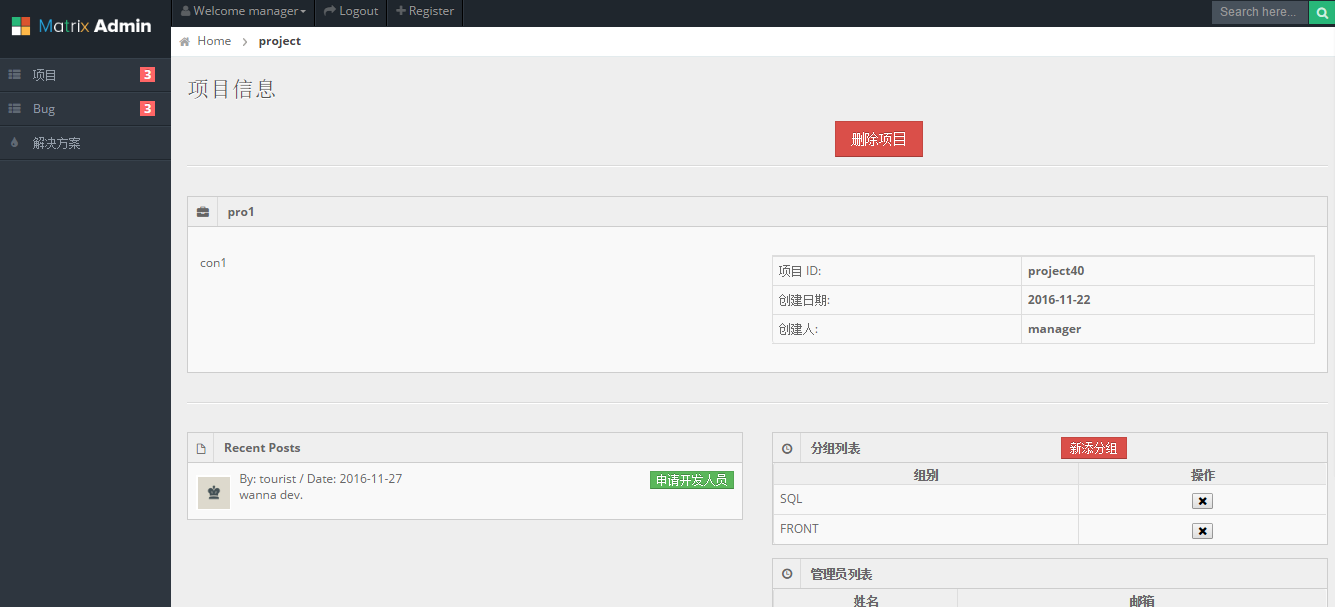
点击 “负责维护的项目”可进入如下页面(点击“测试的项目”进入的界面类似):



同理，可以点击下方项目进入项目的详细界面。

一个项目的详细界面对于不同身份的用户显示出来的东西也不一样。

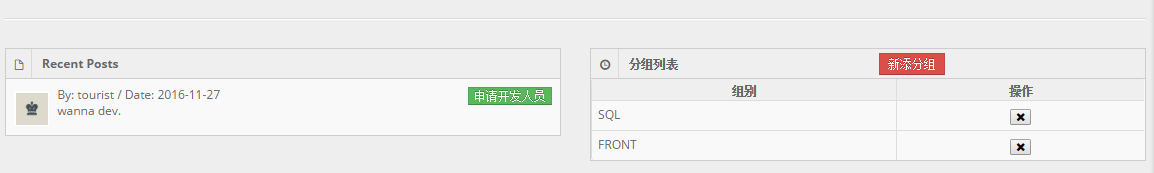
对于管理员，项目的详细界面如下:





点击删除项目就可以将这个项目删除

删除项目按钮下方为项目的详细信息，包括项目名，简介，项目ID,项目创建时间，项目创建人



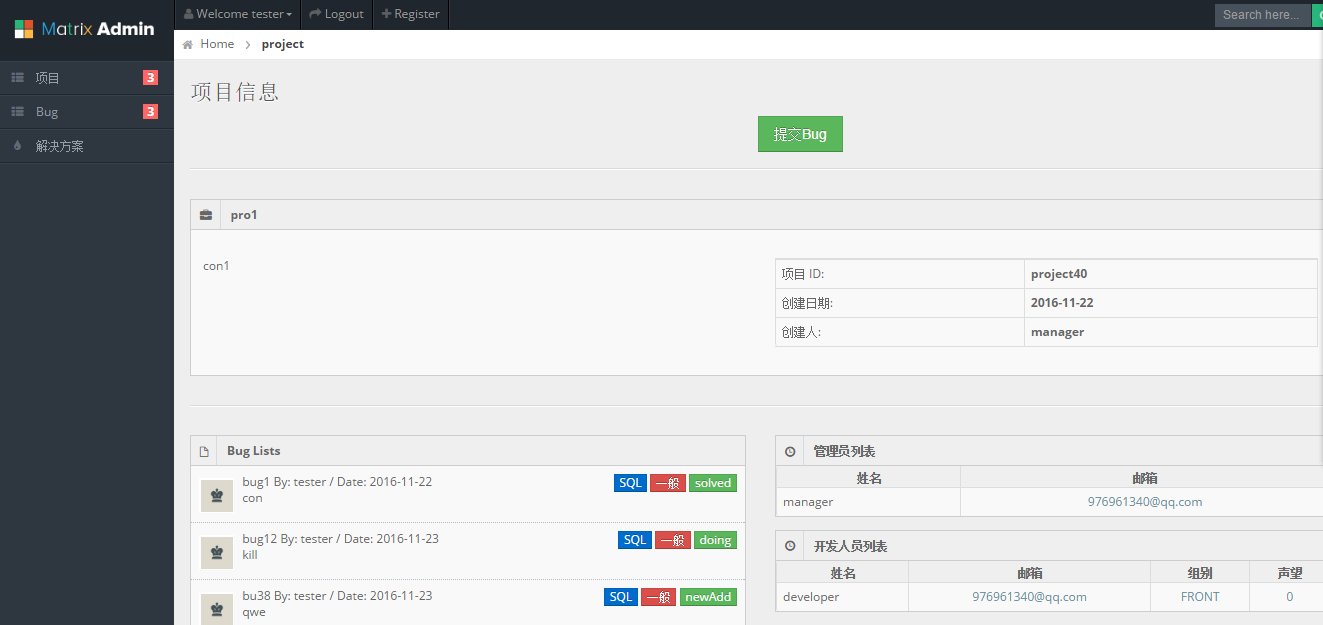
左边显示申请信息，不属于这个项目的用户提出申请之后会在管理员查看的项目界面显示出来，管理员可以点击绿色的按钮同意申请。

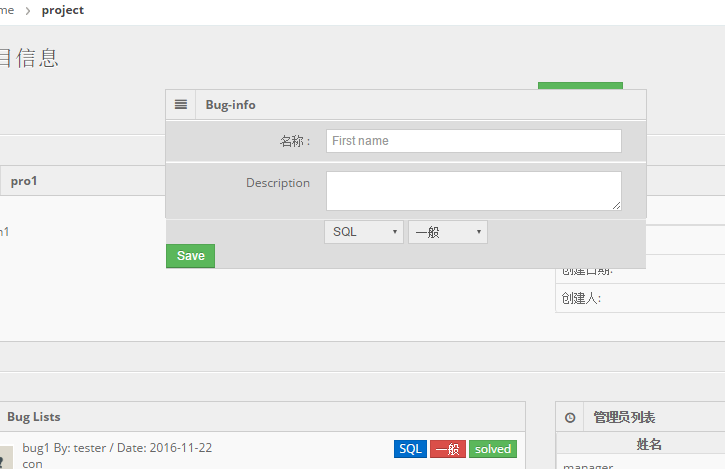
右边为项目分组信息，点击红色的新添分组即可输入新分组的名字并添加。下方列表显示了当前的所有分组，可点击叉按钮删除该分组。



最下方项列表显示项目内所有的成员信息，可点击修改按钮修改这个成员的组别，但是组别只能在现有的分组中选择，点击叉按钮可将该成员移出项目。

对于测试人员，项目的详细界面如下:



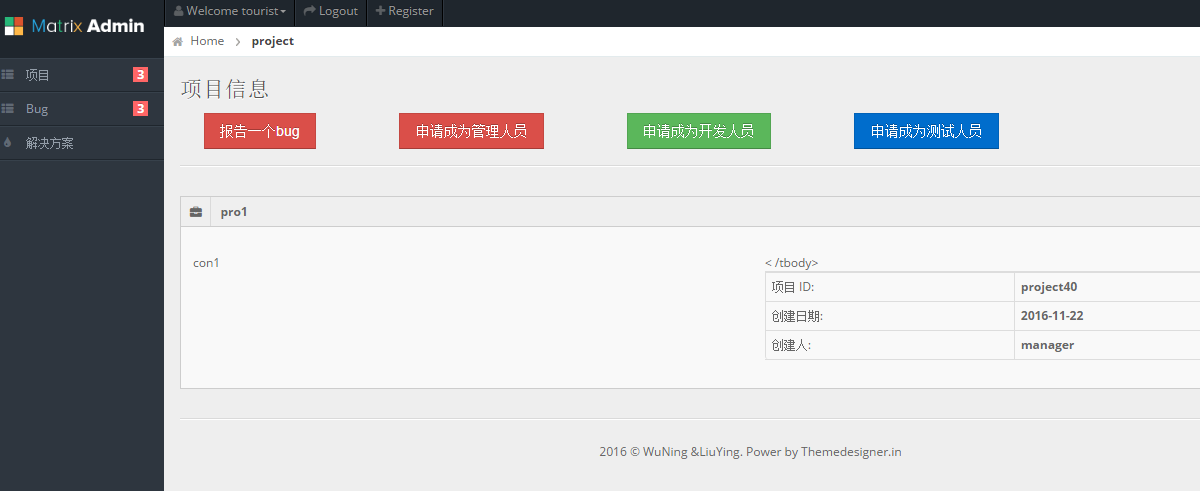


大致布局和管理人员的项目界面类似，但是测试人员没有删除项目的权限，不过测试人员可以点击绿色的提交bug按钮，填写bug的信息，选择bug的分组和优先级，然后提交该bug。

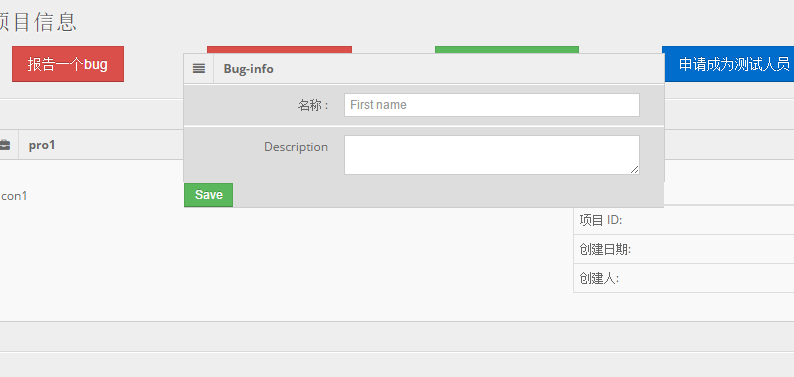
除此之外，测试人员不能审核加入该项目的申请，也不能看到任何的申请信息。在下方左侧会显示出该项目的所有bug，点击bug的名字即可进入到bug的详细界面。

开发人员看到的项目详细界面和测试人员基本一样，不过开发人员不能够提交bug。

非项目内成员可以点击右上角的浏览所有项目来进入和自己无关的项目:



点击报告一个bug即可以项目外成员的身份提交一个bug，该bug需要被测试人员审核之后才能被对应分组的开发人员看到。

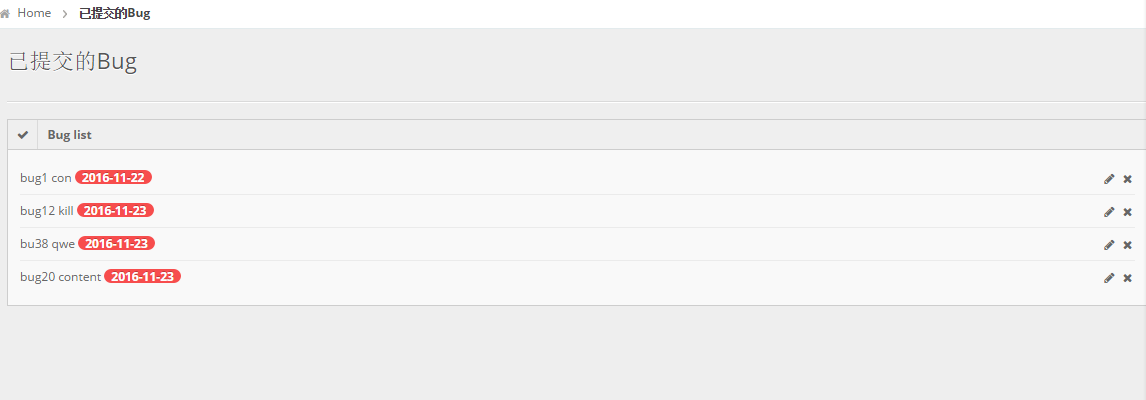


点击剩下的三个按钮，然后在弹出框填写备注信息即可申请该项目的管理人员/测试人员/开发人员。

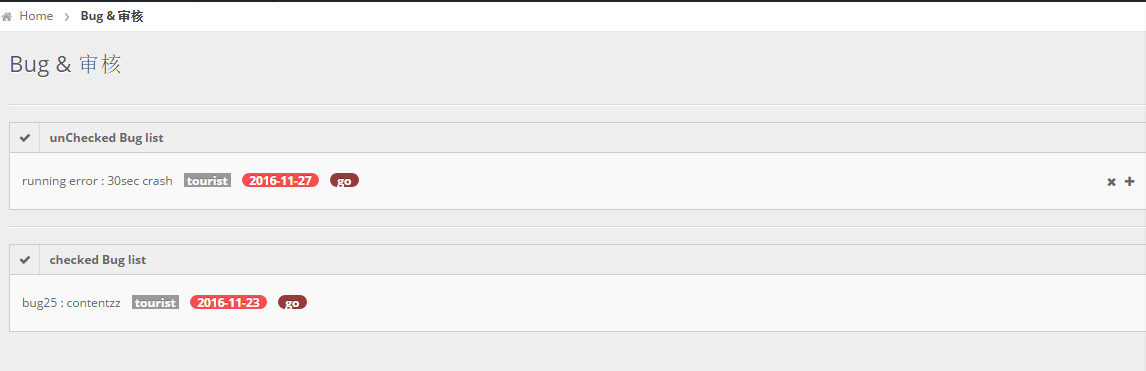
以上为项目管理部分,主要介绍项目的人员管理和信息查看过程。

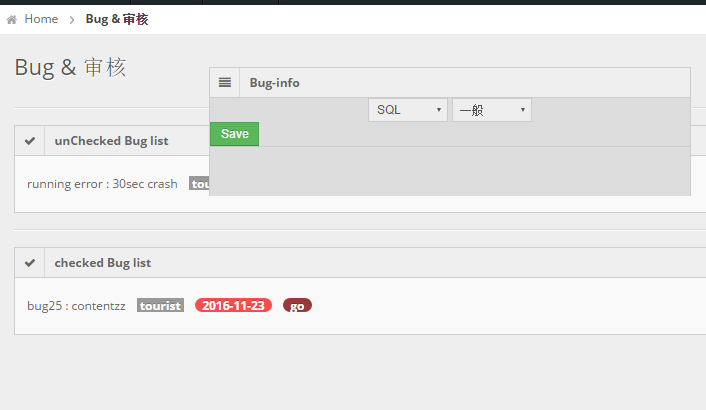
接下来是bug和解决方案部分:

点击首页的已提交的Bug方块，即可见到自己已提交的所有bug，再点击右边的按钮即可进入bug的详细界面

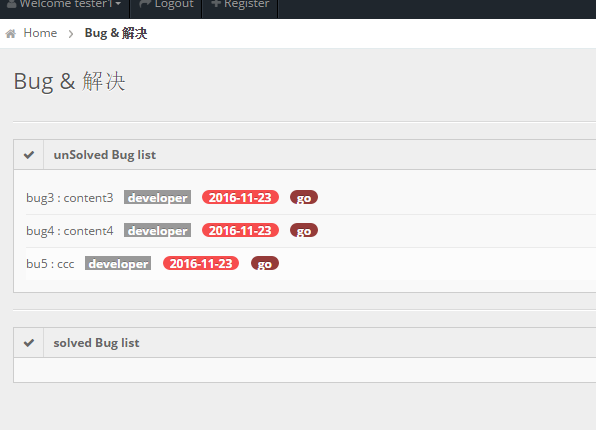


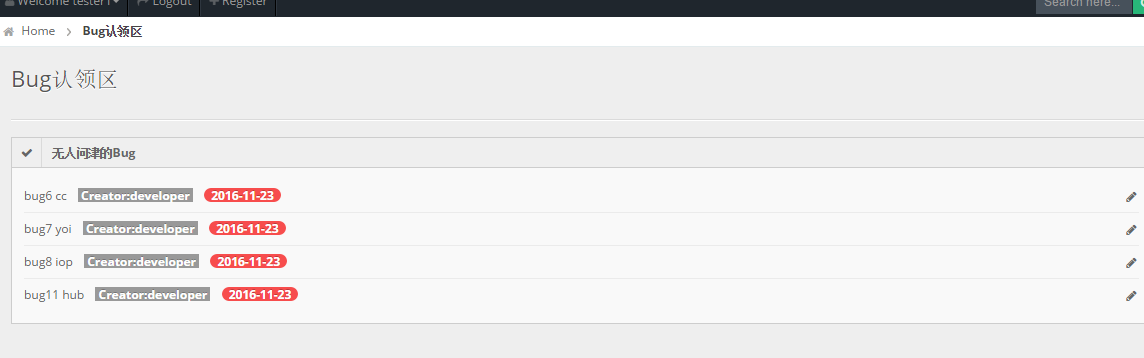
点击首页的Bug&审核方块，即可见到等待自己审核的bug，和自己审核已通过的bug。点击叉按钮就可以删除这个bug，即不通过该bug，点击加号就会弹框要求选择该bug的分组和优先级，点击save即可将该bug状态从waitingexam改为newadd。



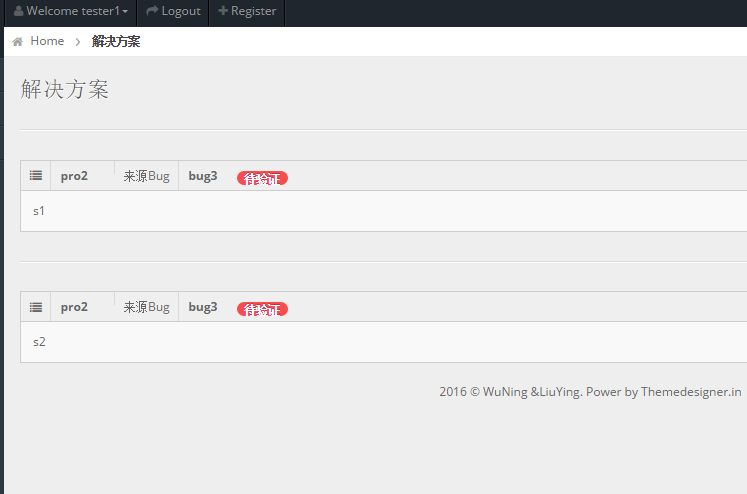


点击首页的Bug&解决方块，即可见到由自己负责的bug，开发人员领取的bug会在unSolved Bug list中显示，点击go，即可前往bug详细界面，下方的Solved Bug list显示的是该成员已解决的bug。



点击首页的Bug认领区方块即可见到和自己分组相同但是还没有人领取的bug。点击右边的按钮即可领取该bug，点击bug的名字即可进入bug详细界面。

点击我的解决方案即可查看当前用户提交的所有bug。点击项目名称即可进入到项目的详情界面，点击bug名字即可进入bug的详情界面。



对于该bug的负责者，即领取了该bug的开发人员，看到的bug详细界面如下，可以点击绿色的按钮提交自己的解决方案:



对于该bug的提出者(需要是项目内的人员)，或者该bug的审核者，即通过了该bug的测试人员，看到的bug详细界面如下:



点击修改bug状态可将bug的状态修改为solved，或者waitSolves等等

点击解决方案的修改状态按钮即可将解决方案的状态修改为通过或者从通过修改为审核中

对于和该bug无关的人员(非该bug的提出者也非该bug的负责者)，看到的界面和bug的负责人员类似，但是没有提出解决方案的按钮。

**七、总结**

刘颖：

通过这次的数据库课设我感受到理论和实践的差距。首先是选题方面经过了仔细的斟酌，最后选择了bug开发系统这一相对有趣的题目。自己开发一个数据库系统要考虑前端和后端的设计以及前端和后端如何连接起来。经过跟结对小伙伴的商量我们选择使用PHP来实现。小伙伴有一定的PHP开发基础，并且也有JavaScript的开发经验，在完成课设的时候相对比较容易。但我在入门阶段遇到了较多的困难，PHP的使用以及JS代码的编写都没有经验。特别感谢结对小伙伴，他在众多deadline压身的情况下还指点我如何实现我这部分的功能，而没有催促或者抱怨我。

助教学长在初期展示的时候提醒我们要多在数据库设计方面下功夫。我们意识到，前期我们在前端界面上投入的精力太多了，注重显示效果如何更好，如何给用户提供更多更方便的功能，忽略了后端的设计部分。于是我们又设计了视图、存储过程等。另外我们在基本表的设计上有点粗糙，之前设计基本表时我们参考了一个大型系统的设计方法：基于查询方便和高可扩展性的原则，我们在每新建一个工程时都会新建很多表来存储工程的相关信息。经过学长的提醒我们意识到这样可能会带来一定的数据冗余。如果要让我给下一届学弟学妹建议的话，我希望他们能在一开始的时候就把握好重点，在数据库的设计方面投入更多的精力。

经过这次结对编程，我也从小伙伴的身上学习到了很多。之前自己比较注重课本上的知识，对项目开发不是很了解。经过与小伙伴一起完成数据库课设，我发现通过编写代码能更加透彻的理解课本上的知识。正如我们从小就学到的那句诗：纸上得来终觉浅，绝知此事要躬行。

吴宁:

在这次的数据库课设上我们从无到有设计并实现一个数据库系统，一路上学到了很多东西也犯了很多错误。比较大的收获就是理解并实现了存储过程，触发器和视图，感受到了它们在数据库开发中给人带来的便利。比较重大的错误就是把表设计地太多了，导致了代码实现时的困难，并且因此出了很多bug,花费了大量的时间用来调试。

很多时候不动手的话是看不出自己的设计到底有什么问题的，往往在实现的时候才会发现某些字段是冗余的，有的表缺少了重要的字段，需要再设计一张表，有的表是不必要的等等。通过这次的实践，我们不仅在数据库的设计上面积累了非常宝贵的经验，在具体实践上也是收获匪浅:Get方法和post方法，javascript的ajax技术，mysql的一些tricks等等，让我们对前后端的交互，sql语言都有了更为深入的理解,sql语言可远远不止insert,select,update,delete,create这寥寥几个操作。

使用原生的sql语句进行数据库系统的开发固然痛苦，但却是深入理解sql语言的必经之路，一个只会调包用框架的程序员注定编写不出优秀的数据库管理系统。