"Develop Performance Farm Database and Website" proposal for Google Summer of Code 2018

Hongyuan Ma

mahongyuan1997@gmail.com(CS_MaleicAcid@163.com)

Table of content

---

1. Overview

1.1 Implementation

I plan to implement the PerfFarm project using a separate front-end and back-end development approach and use PostgreSQL as a database for storing test results.The front-end project will be built using React16. Bootstrap will be used as the ui library and Yarn will be the package manager.

The backend project will import Django Rest Framework on top of existing Django. And the Django version will be upgraded to the latest TLS version, which is now 1.11. There will be no html file in the backend project. Front-end and back-end applications will use restful apis to exchange data.

In addition, the backend project will have an admin module. Only administrators of the PG Performance Farm project can log in to the admin module. In the admin module, the administrator has the ability to modify the machine's submission rights, send registration confirmation emails, and so on.

1.2 Goals

The main goal is to provide a website for the PostgreSQL community. Users can register the machine and upload performance test results to this site via client-side scripts. Users can also search and review the test results reported by others on this site.

1.3 Benefits

This project can provide a platform for the PostgreSQL community to collect performance test reports. With this platform, developers and enthusiasts of the PostgreSQL community can clearly understand the performance of PostgreSQL under different machine environments. At the same time, developers and enthusiasts can contribute better to the PostgreSQL community by getting more performance feedback.

2. The front-end application

2.1 Technologies and libraries usage

This front-end application will use React16 as a development framework. Yarn will be its package manager, and Bootstrap will be the main UI library. Due to the use of React16, this project will be written using ES6.

2.2 Pages and functions

2.2.1 The header component

The head component is a common module used by almost all pages. In new front-end applications, the header component will be rewritten based on the existing html file. The head component will contain the site's logo and name as well as a user avatar component for the user to login and manage their machine.

A horizontal navigation bar is provided at the bottom of the head component to guide the user to quickly jump to the page they are interested in.The navigation bar contains at least "Home", "Status", "Machines", and "Help" items.The following will describe the page corresponding to the navigation item.

2.2.2 User's machine management page

The user can login by clicking on the user avatar component.After the user passes the authentication, he can register, browse and manage his machines on this page.

An "Add button" will be provided on this page, which is used to register a new machine. When the user clicks this button, a machine registration form will pop up. The user can fill in the information and submit the form. After checking with the administrator, this machine can be used to upload test results.

User registered machines will be displayed in a list of machines. Each item in the list represents a machine. The machine's activity status, alias, secret key and other information will be displayed. Machine information editing functions are also provided.

2.2.3 Home page

The “Home” item in the navigation bar corresponds to this page.
The homepage will briefly introduce the purpose of the PG Performance Farm website. It will also tell users how to quickly see what they are interested in. At the same time, guidance on how to perform performance tests and submit test results will be provided to users.

2.2.4 Test result list page

The “Status” item in the navigation bar corresponds to this page.
This page will list the latest submitted test results. The test result item will contain the machine alias, machine brief information (OS, compiler, architecture), test item name, test branch, performance (advanced or backward), test date, etc.

The user can filter and sort the result list by changing the branch, performance, test date and other conditions.

2.2.5 Test result details page

Each test result item will have its own details page. The link to the detail page should be easily found in the list item.

The test result details page contains details of the performance test. The information provided on this page includes: machine alias, test date, test branch name, test item name, test item metric score, test result score, system information, etc.

2.2.6 Machine list page

The "Machines" item in the navigation bar corresponds to this page.
This page presents a list of machines where each entry corresponds to a registered and available machine. Machine aliases, owner emails, machine brief information (OS, compiler, architecture), recently submitted reports, etc. will be presented to the user.

2.2.7 Machine details page

When the user clicks on the machine's alias in the machine list item, he will jump to this machine's details page.

The machine details page not only contains the details of the machine but also lists the test results that the machine has submitted. On this page, users can filter the report history of this machine based on the branch version, performance, and other conditions.

2.2.8 Help pages

The “Help” drop-down list item in the navigation bar correspond to these help pages.The help pages includes the following items:

- Contact page
  This page provides the contact information and mailing list of the Performance Farm webmaster. For users to encounter problems when they can ask for help.
- Licence page
  This page is used to inform the user of the license used by the Performance Farm project.
- Privacy Policy page
  This page is used to inform users of the privacy policy used by the Performance Farm project.

3. The Django server

3.1 Technologies and libraries usage

The backend server uses the python language and is developed based on the Django framework. Test report data is stored in the PostgreSQL database.The existing Django version is 1.8, and I plan to upgrade it to the latest TLS version, which is currently 1.11. Django REST Framework will also be imported and restful api will be created based on it.

3.2 Structure design of data tables

The data table will be created using Django's Model class. The following will describe the main data tables that I plan to create.

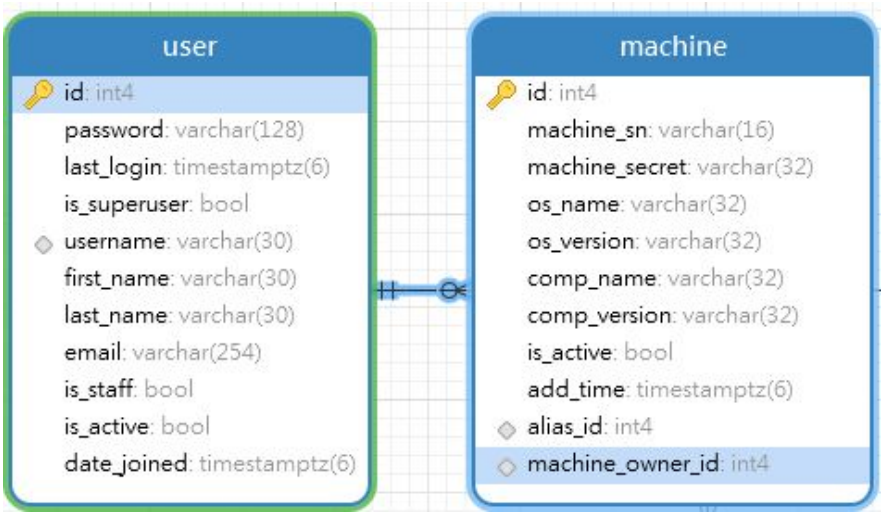
3.2.1 User table

The user table is created using the User class contained in django.contrib.auth.models. Users who are authenticated and passed through the auth module will be stored in this table.

3.2.2 Machine table

The machine table is used to store user registered machines. This includes machine_secret, alias_id, os_name, os_version, comp_name, comp_version, is_active, machine_own_id, and other fields.

The machine_own_id field is used as the foreign key, which corresponds to the primary key of the user table. A user can correspond to multiple machines, as shown in the following figure:



3.2.3 Alias table

The alias table is used to store the machine's alias. After the user registers the machine, if the administrator approves the machine, an alias is assigned to the machine.

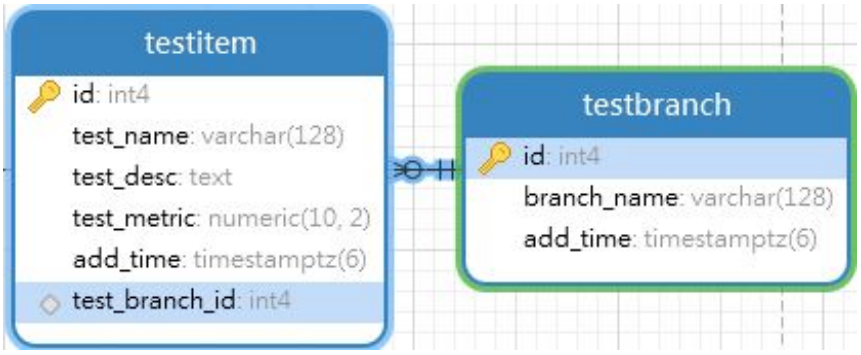
After communicating with the mentors, I learned that Performance Farm will use plant or fish as an alias because the Build Farm project has used the

animal name as an alias. I plan to write crawlers and then crawl the required alias from the wiki site (https://en.wikipedia.org/wiki/List_of_garden_plants).

3.2.4 Test item table and test branch table

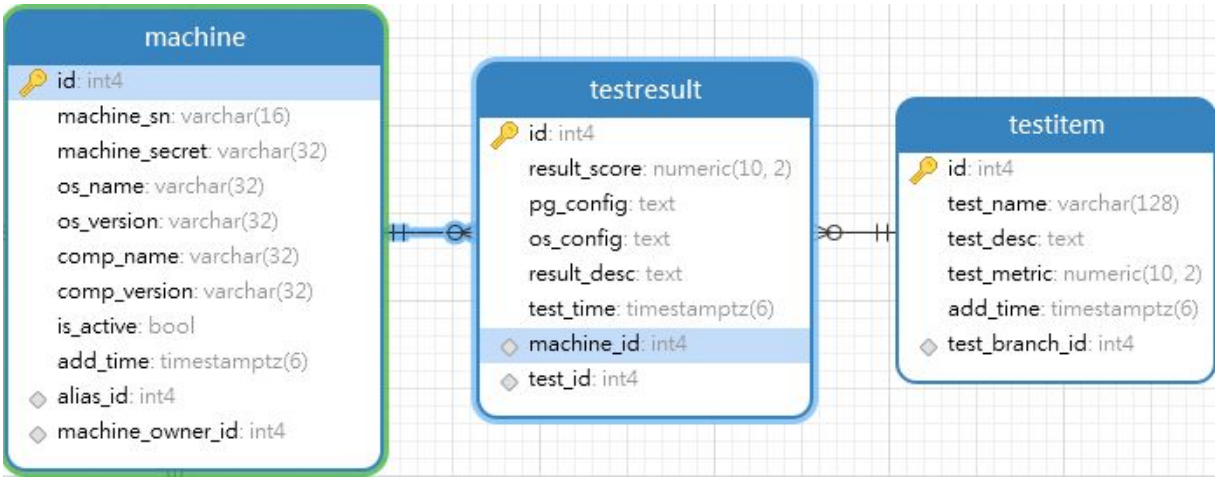
The test branch table is used to record the branch version of PostgreSQL. The test item table is used to record the contents of the test items, including test_name, test_desc, and test_metric fields.

The test_branch_id field is used as a foreign key, which corresponds to the primary key in the test branch table. As shown in the figure, one test branch can correspond to multiple test items.



3.2.5 Test result table

The test result table is used to store the details of the performance test report. Including test_score, pg_config, os_config, result_desc and other fields. The test result table has two foreign key fields: machine_id and test_id. They correspond to the primary key of the machine table and the testitem table, respectively.The machine_id field indicates which machine submitted the test report and test_id indicates the test item.As shown below:



3.3 Web api interfaces provided

3.3.1 User-related interfaces

User-related interfaces are developed based on existing auth modules. The auth module will be slightly modified: The function's return type will be changed to the json type. User-related interfaces are:

- User login interface
- User logout interface
- User search interface

3.3.2 Machine-related interfaces

Machine-related interfaces are used to obtain machine information or to add new machines after the user's identity is authenticated. Machine-related interfaces are:

- Machine list interface (used in machine list page)
- Machine details interface (used in machine details page)
- Machine registration interface (This interface can only be called after the user has been authenticated.)

3.3.3 Test result related interface

These interfaces provide test result information. The interface can sort and filter the data in the database according to the user-specified parameters. These interfaces are:

- Test result list interface (used in test result list page)
- Test result details interface (used in test result detail page)

- Test result upload interface (client can upload reports through this interface)

3.4 The admin module

The admin module will be developed based on Django's own admin module. This module is mainly used by webmasters to approve and manage registered machines.

Initially, the status of the newly registered machine is "pending." When the administrator modifies the status of this machine to "available", the program will modify the corresponding is_active field of the machine and assign the machine an alias and an secret key. The program then automatically emails the machine information to the registrant.

If the administrator finds that an abnormal report has been submitted, he can also modify the machine's activity status and delete the abnormal report.

4. Schedule and milestones

4.1 The first milestone

The goal of this phase is to start the project and do some basic coding work to prepare for the later development. I will initialize the front-end application and rewrite some existing html files. In the Django server, I will upgrade the version of Django and import the needed libraries. I would write a web crawler to get the required alias data, and the model class would also be written.

Task	Duration
Initialize the front-end application. Upgrade the Django version and import the required libraries. Write model classed and generate data tables. Write the crawler to get the required alias from the wiki(3.2.3).	2 weeks
Use React to rewrite existing html files (2.2.1, 2.2.3, 2.2.8). Modify the function return format of the auth module(3.3.1).	2 weeks

4.2 The second milestone

At this stage I will develop frontend pages and restful api. Most of the project's features will be implemented.

Task	Duration
Implement user login function. Realize the function of the user to add machines(2.2.2, 3.3.2).	1.5 weeks
Implement the machine list page, machine details page and related api(2.2.6, 2.2.7, 3.3.2).	1.5 weeks
Implement the test result list page and api.(2.2.4, 3.3.2)	1 week

4.3 The final milestone

At this stage, I will continue to write unimplemented functions and write admin module. Modify the code according to the opinions of the tutor and finally deliver the project.

Task	Duration
Implement the test result details page(2.2.5) and implement the test result upload interface(3.3.3).	1.5 weeks
Implement the admin module(3.4).	1.5 weeks
Modify the code based on comments and eventually deliver the project.	1 week

5. About me

My name is Hongyuan Ma and I am a junior in software engineering major in Shanghai. My time zone is UTC+8. I have used python, java, golang, js and other languages, and I am familiar with common web development frameworks. I participated in the development of web applications from my sophomore year and I have experience coding several small websites. I am passionate about open source and I very much hope to contribute to the open source community.

My github name is MaleicAcid.For some time in the past, I have used "CS_MaleicAcid@163.com" email address to communicate with mentors. I also sent a patch using this email address. However, due to some reasons, I should consider "mahongyuan1997@gmail.com" as my main email address. If you think there is something wrong with my draft, please email me at anytime. I am willing to discuss with you and improve the deficiencies of my proposal.