# GSoC 2021 PostgreSQL Project Proposal

Develop Performance Farm Benchmarks and Website

## 1. Basic Information

• Name: YoungHwan Joo

• Website (CV & Portfolio & Links): https://rulyox.com

• Email: rulyox@gmail.com

• Location: Seoul, South Korea (UTC+09:00)

## 2. About me

I am YoungHwan Joo, a 20-year-old student majoring Computer Science in Hanyang University, Seoul, South Korea. I finished 4 semesters and I took a semester off. So, I am not attending university classes in 2021. I am mostly interested in full-stack web development and databases.

I am currently working at Bio-Medical Informatics Lab, Seoul National University Hospital as an application developer and a data scientist. I worked at SNUH from December 2019. I worked full-time in vacations and part-time during the semester. One of the biggest projects that I developed is called Drug Dashboard. This is a web application that visualizes statistics about drugs that are used. I used the OMOP CDM Schema on PostgreSQL and AWS Redshift. I developed this full-stack including databases, back-end, front-end, and deployment.

Also, I am participating at a project called LobbyView which is from Massachusetts Institute of Technology. I worked on data querying using SQL, front-end development, and API development. I am working at LobbyView since July 2020 and this project is not a full-time job but more of a sub project.

I am familiar with technological stacks such as Front-end using React, Vue, Back-end using Node.js Express, Python Flask, Django, Java Spring, Relational Databases, and Infrastructure Systems such as Docker.

Furthermore, I love learning new technology and solving problems. I can quickly learn and combine software skills to solve a complex problem.

## 3. Why I am interested in PostgreSQL

I have always been passionate about open-source and I use a lot of open-source software, frameworks, and libraries. However, I didn't have a chance to contribute to an actual open-source project. This is why I am applying to GSoC 2021.

Because my biggest interests are web development and databases, and because I am using PostgreSQL a lot, I thought that I would be most passionate if I could contribute to PostgreSQL's projects.

While I was working in the organizations above and doing my personal projects, I learned some complex SQL skills such as Window Functions, CUBE, and ways to optimize Common Table Expressions. I am also a big fan of JSONB.

## 4. Project Abstract

PostgreSQL is a big and active open-source project. This means that the code changes frequently and there are a lot of branches and commits. When big changes are made, performance has to be tested.

Performance Farm is a program that can be used to collect and visualize the benchmark results while the source code changes.

The actual tests can be run on Linux and maxOS machines while the users can easily see the results using a website.

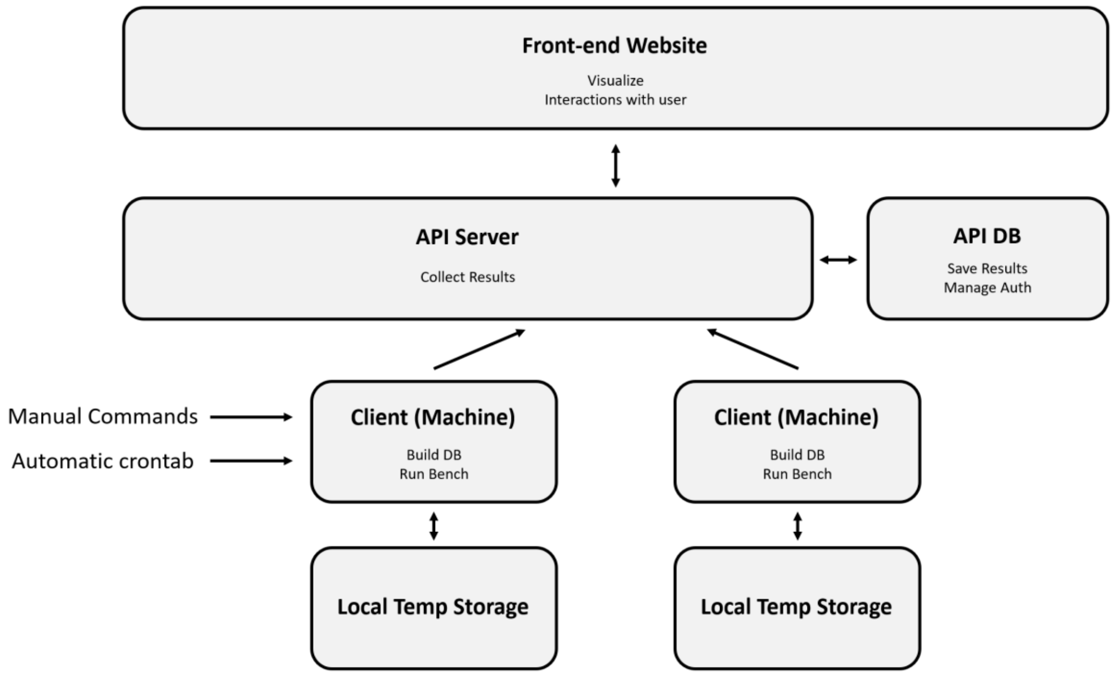
## 5. Current System

The Performance Farm consists of 3 parts.

• Client: Clones source code from a specific repository. Create a temporary PostgreSQL cluster. Run *pgbench*. Collect results. Upload results to the API.

• API: Uses Django. Receive benchmark results from the Client. Provides REST API. Uses DB for storing past results and user information.

• Front-end Website: Built with Vue. Visualizes benchmark results.



Current System Diagram

## 6. Features to be implemented

Below are the features that were in the GSoC ideas list and I added some details.

• Collect system metadata

Integration of *collectd* and *pg_stat_statement* data.

When doing a benchmark, collecting information about the system is important.

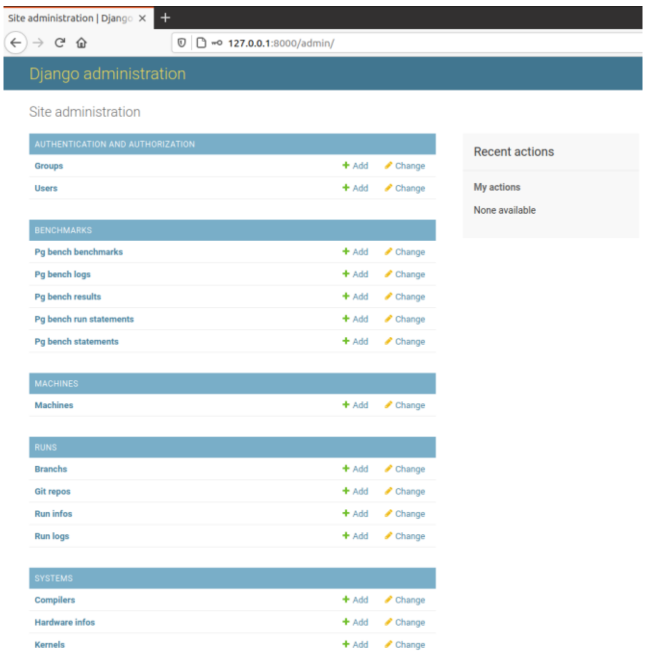
In the current version, only the operating system and kernel data is collected.

• Admin panel inside website

Currently, the only way to change the API's configurations is through the Django Administration. However, this should not be open to the public.

Adding an admin panel to the Front-end Website would be the appropriate way.

In the admin panel, there should be menus to manage the connected machines.



Django Administration Page

• GUI improvements

User Interface and User Experience is always an important point to consider.

The users of Performance Farm should be able to easily see the benchmark results, find especially fast or slow results, and compare results between specific time points or git commits.

• Dependency removal

The Client's and API's dependencies are managed by requirements.txt and I think that not much packages are being used. However, I believe that packages such as *requests* or *simplejson* could be removed.

The Front-end Website uses a lot of npm packages. Vue itself is also a dependency. Conversion to vanilla JavaScript will also be an improvement. However, if Vuetify is removed, we should think about alternative design systems or ways to deliver a clean and seamless UI.

## 7. Potential ideas

These are the ideas that I thought would be a nice feature for Performance Farm. However, these ideas should be discussed more and they have lower priorities.

• Benchmark execution through API or Website

Currently to manually start a benchmark, a user has to access the Client directly.

I think permitting the administrator or privileged users to start a benchmark at a specific machine would be a useful feature.

• Watch for git actions

Currently, tools such as *crontab* are used for automatic executions. However, checking the actual git actions would be more efficient and accurate than simply running according to time.

I think that developing a small program that watches for commits, pull requests, merges and executes Performance Farm Client would be nice.

• Support custom forks

There are organizations and companies that build their own fork version of PostgreSQL. I thought that these organizations might want to run Performance Farm specifically for their forks to test their version's performance.

The current version of Performance Farm only displays results of official repositories. I hope to add options that enables to choose certain repositories to be displayed.

• Containerizing using Docker

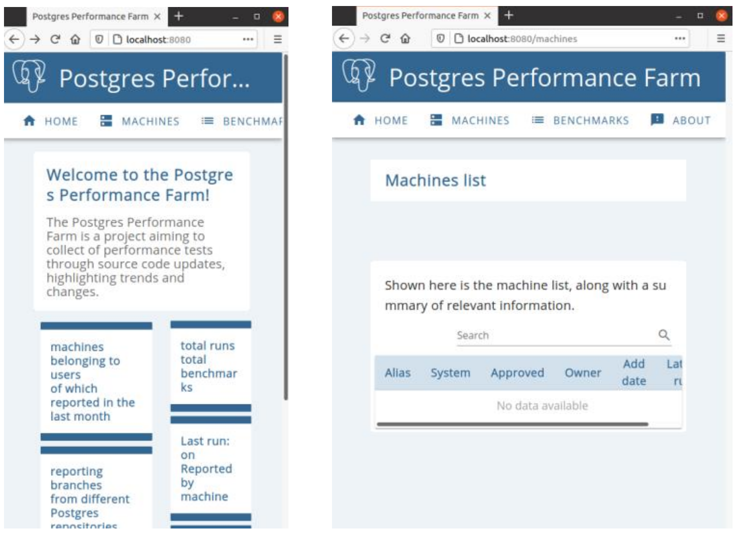
I have cloned the Performance Farm source code and tested it. However, I had some dependency issues and some processes were uncomfortable. Because Performance Farm builds the whole DBMS, there were a lot of requirements needed.

If Performance Farm gets containerized, not only setting up everything on a new machine but also processes such as updating Performance Farm to a new version will be much more easier.

• Responsive web page

I guess that most people who use Performance Farm would use their computers than mobile devices. Still, I think it would be better if various browser sizes display the website properly.

Below are examples of broken pages in small screen sizes.



Welcome to the Postgre s Performance Farm!

The Postgres Performance Farm is a project aiming to collect of performance tests through source code updates, highlighting trends and changes.

machines belonging to users of which reported in the last month

total runs total benchmar ks

reporting branches from different Postgres repositories

Last run: on Reported by machine

Shown here is the machine list, along with a su mmary of relevant information.

Examples of small sized browsers

• Code conventions and testing

Many people can contribute to open-source projects. This means that having a rule to enforce a consistent code style can be helpful. Also, unit testing could be a great way to find bugs and fix them.

I checked that PEP8 and ESLint are being used in the current code. When using JavaScript, TypeScript could also be used.

For testing, Python unittest and JavaScipt libraries such as Jest can be used.

## 8. Schedule

• Application Period ~ April 21 (3 weeks)

Before GSoC announcement.

Test Performance Farm using different environments and settings.

Use and understand unfamiliar PostgreSQL tools such as *pgbench* and *pg_ctl*.

Analyze existing code.

• April 21 ~ May 18 (4 weeks)

Design concrete improvement methods.

Make mockups for new web elements and the admin page.

• May 18

GSoC announcement.

• May 18 ~ June 2 (2 weeks)

Get to know the team better.

Community bonding is an important step because many people participate in open source projects.

Start discussing about potential ideas.

• June 2 ~ June 23 (3 weeks)

Coding officially starts.

Upgrade the Client and the API.

Add metadata collection to the Client

Add authentication features to the API.

• June 23 ~ July 13 (3 weeks)

Upgrade the front-end website.

Add the admin panel.

Improve GUI.

• July 13 ~ July 17

First evaluation between mentors and students.

- July 13 ~ July 28 (2 weeks)

Try to get rid of dependencies.

Migrate to vanilla JavaScript.

- July 28 ~ August 17 (3 weeks)

Test and fix bugs.

Implement other potential ideas after discussing with mentors and the team.

- August 17 ~ August 24

Submit code and final evaluation.

- The future

I hope to contribute to Performance Farm and other software in the PostgreSQL ecosystem even after GSoC 2021.