Google Summer of Code 2016 Proposal

LabLua

Name: Nikhil. R
Email: rnikhil96@outlook.com , rnikhil2751996@gmail.com

BASICS

1) What is your preferred e-mail address?

E-mail 1: rnikhil96@outlook.com
E-mail 2: rnikhil2751996@gmail.com

2) What is your web page / blog / GitHub?

Blog: http://rnikhil275.github.io/
Resume: http://rnikhil275.github.io/assets/Nikhil.%20R-%20Resume.pdf
GitHub: https://github.com/rnikhil275/
StackOverflow: http://stackoverflow.com/users/4379159/rnikhil275
The blog is still under construction. Will start posting once GSoc starts.

3) What is your academic background?

Third year undergraduate at Birla Institute of Technology and Science Pilani, Pilani Campus, India studying Dual degree in M.Sc (Hons.) Economics and B.E Manufacturing Engineering

4) What other time commitments, such as school work, another job (GSoC is a full-time activity!), or planned vacations will you have during the period of GSoC?

I'm completely free from May 23 to August 23. I have no other commitments/jobs. I may take one or two day break which will be informed ahead in time. I will work every day and hence can easily target about 50 hours a week.

Experience

1) What programming languages are you fluent in? Which tools do you normally use for development?

I am quite proficient in PHP, Python, Lua, JavaScript, and C. I can also program in Ruby and Erlang but would require some googling.
I use Ubuntu (15.10) with LXDE on top as my main OS but I also boot into windows for domain specific work. I work in terminal most of the time and my preferred choice of text editor is Sublime Text with minor edits done using nano. I use Git for version control.

2) Are you familiar with the Lua programming language? Have you developed any projects using Lua?

Yes, I am familiar with Lua programming language. I can use most of the features of the language like tables, metatables effectively.
I am new to the community and I have started work on the following projects:

Sailor-admin center: https://github.com/rnikhil275/sailor-admin
Luagoogle: https://github.com/rnikhil275/luagoogle
Sailor: https://github.com/rnikhil275/sailor

I am also trying to contribute to the Sailor project by solving issues.

Apart from this, I have also written a few scripts for my campus DC++ client.

3) Have you developed software in a team environment before? Any projects with actual users?

Yes, I have lot of experience developing software in teams before.
I was part of an organization called SSMS (Student Society for Mess Services) in my second year and we were responsible for managing the mess accounts of 4500 students. I was also part of a startup (as the backend developer). I also interned at Vedanta Resources last summer where we worked on automating a bunch of data collection and logging system. These projects have actual users.

4) 1) What kinds of projects/software have you worked on previously? (Anything larger than a class project: academic research, internships, freelance, hobby projects, etc.)

SSMS Portal: (https://github.com/SSMS-Pilani)
It's a government registered organization responsible for development and maintenance of Billing and Inventory Management System used in all the BITS Pilani messes and all night canteen. I co-wrote the backend and the web interface for the website and also helped in the development of the software run on messes itself. All billing through the ID card system are managed on this portal and every month end this data is organized and sent to the institute for processing. The website is live on our intranet.
Tech Used: PHP, MySQL

Doctorsnow: (https://github.com/rnikhil275/doctorsnow)

It the backend for a telemedicine startup which I built. It's a doctor patient management system with features like appointment booking, video call etc. The backend is a REST API which interacts with the Angularjs frontend using JSON. I used SlimPHP for the routing, MySQL for the database, Redbeans for the ORM layer and Apache for the server. It also has a signaling server for the WebRTC module which is written in Erlang.

Protocol Development (CEERI Pilani): (Code not on the internet)

I worked under Mr. Solomon Raju, a scientist in Central Electronics Engineering Research Institute to develop a function for the MAC layer of the protocol stack they are developing. I wrote a CSMS/CA (Carrier Sense multiple access with collision avoidance) function to send data from various nodes to the main node. It checks whether the main node is active (not in "rest" mode), not busy transferring data from other nodes and then starts sending data to the node. It also makes sure there is no packet loss. I wrote code on a simulator which then converts it to the architecture/assembly set of the underlying bread board and microprocessor.

StarMash: (https://github.com/rnikhil275/StarMash)

It's similar to FaceMash (From "The Social Network" Movie) It was a fun project which scraps a bunch of photos from IMDB and pairs them up randomly for to be rated by the user. It uses Elo rating system and I used this algorithm (http://stackoverflow.com/questions/3848004/facemash-algorithm). The site hosted from my laptop got around 1000 unique IP hits in 24 hours. The entire site is built in PHP hosted on an nginx server.

Vedanta Resources: (Internship)

Vedanta Resouces (https://en.wikipedia.org/wiki/Vedanta_Resources) is an aluminum/ power manufacturing company and it's one of the biggest in the world. I interned with them during last summer as part of my institute's Practice school program. I analyzed and automated a lot of internal review, data reporting and logging systems. Saved around 80 hours of manual labour monthly for the organization by setting up various cron jobs to collect data and assemble them in proper format. The entire code was written in Python with Flask and Python-excel packages like openpyxl, xlutils, etc.

I have done a bunch of other personal projects also which can found on my GitHub profile (https://github.com/rnikhil275/)

2) In particular, are you (or have you been) involved with any open source development project? If so, briefly describe the project and the scope of your involvement.

Some of the projects above are open source. I haven't contributed to any big open source organization but I do contribute to some small ones as seen on my GitHub profile.

Project

1) Did you select a project from our list? If yes, which project did you select? Why did you choose this project? If you are proposing a project, give a description of your proposal, including the expected results.

Yes & No. I did not select Luagoogle and the admin center from the list but the integration of Sailor and Elasticsearch was selected from the list.

The reasons for selecting the projects are below.

I decided to work on the Admin Center because it was the most beginner friendly project I found on the Sailor To Do list. I thought it would give me a decent understanding of working of the internals of Sailor framework.

I decided to work on the elasticsearch integration with sailor is because I found this to be a very realistic project which I found myself using in the future. The concepts of storing relational data in a search index effectively for faster searching can be taken further for the web applications I would be writing in the future. I felt I can learn the most by doing this and found it to be fun.

Luagoogle would be my first package in luarocks. Though it does a niche job, maintaining it with all the API changes, publishing it, would give me a experience which can be taken ahead for more complex projects in the future. I would learn about luarocks packaging, writing rockspecs, dependency management etc.
I found the Sailor community a good match for me and decided to work with them. They were quite helpful from the beginning and I would love to work with them on the project.
Sailor Project: https://github.com/sailorproject/sailor
http://sailorproject.org/

I have decided to do a three things and I will be explaining each one in detail

A) Admin Center for the Sailor MVC framework.

I shall be referring to this as the base URL (base): https://github.com/sailorproject/sailor/tree/master/src

Right now, there are a bunch of auto generator functions inside base/sailor/autogen.lua. They can be used after changing a setting in the conf file. At present this autogen function (after enabling it), given a table name generates the model and controller code for CRUD operations. They can later be accessed under /?r= (controller name) route.

I want to build an entire admin app which encompasses these autogen functions inside it and also another feature for changing stuff in the configuration file itself.

First it will be another controller named admin in base/sailor/blank-app. There will also be corresponding models and views. This entire module would be disabled by default and the user must enter the admin username and password to enable it.

The access module will be used for the authentication purposes for the admin center. The session state would be maintained using the same module.(A future edit maybe be necessary after successful solving of https://github.com/sailorproject/sailor/issues/58). A user has to manually enable the admin module and set a password too. The access module's hashing function would be used and the password would be authorized and verified. There would be all kind of standard checks during authorization. The username and password should not be same (like user: admin pass: admin), there would be checking for weak password. All this would be done before hashing and storing the password.

After authentication it will redirect to the dashboard where there would be options for making models and CRUD code. There would also be an option for editing the configuration file though it would require a restart of the server for changes to take effect. A logout option will also be available. The Web interface would be bootstrap using the signin class. It will look

something like this https://getbootstrap.com/examples/signin/ . The data would be sent as a post request to the backend. The session state is maintained using the access module. Once logged in there would be HTML form for inputting the table name so that the corresponding model can be generated. The table has to be made beforehand. Once the model has been generated the CRUD functions can also be generated for the corresponding model. The code for all this is already available inside base/sailor/autogen.lua.
All these functions will be edited to check for auth so that they cannot be used by a non-admin user. The web interface for all this is also inside the autogen.lua file. Everything would be documented while the code is written itself.

Now coming on the conf file editing feature, it will basically be a huge HTML form will all the fields entered before (We know the conf file the user has). The present values will also be set as placeholders in the input boxes. The user after making changes should submit on which the changed values (can be found using js) would be replaced on. There will also be a small check to check the integrity of the values. The module used for this would be https://github.com/sailorproject/valua .

The logout option destroys the session and the user has to signin again to access the dashboard. The user will be redirected the signin page again. The signing page also will have an option to redirect to index.lua.

B) Integrate Elasticsearch-lua client into Sailor:

Right now, the elasticsearch-lua client works fine independently. The aim of this project is to accomplish exporting the models used in the particular application to elasticsearch indexes so that they can be used inside the application namespace for searching. It's would include elasticsearch-lua client library as a dependency.

The way the models are represented in the elasticsearch index depends entirely on the search use case scenario and this choice should be left to the user. This project would be focusing on implementing the necessary functions to achieve the above from inside sailor.

Stage 1: Making connection to the elasticsearch index and exporting models to its indexes as documents.

The first month would be spent discussing with the mentor on various design cases and how the models should be indexed.
The above choice in implementing which way to search the data is left to the user.
The overall plugin (search model) would consist of a few modules. I will explain each one in detail here.

Connection Module: This module will basically deal with the elasticsearch-lua client by requiring it and then making connection to a node in the cluster. All the http methods will also be in our namespace for us to use. Now we can use the connection instance to store and modify data.

Index Module: This module will be used to index/store data into the elasticsearch index.
Client:index() of elasticsearch would be used. Again here, the use case would be dependent on the user as discussed above.
The models based on the present database structure will be stored as necessary in the elasticsearch index. The _bulk endpoint can be used to store a lot of data in one go.

Indexing relational data in elasticsearch can be done in the following ways:
1) Indexing each row as a document
Each row in the table would be document with id corresponding the ‘id’ column name.

2) Indexing multiple rows as one document
This would first require a join of the tables and then storing the new table into the index. This is an example of a particular use case.
3) Indexing a set of columns as nested object
4) Indexing multiple nested objects per document

All four scenarios are explained in this page I found:
http://voormedia.com/blog/2014/06/four-ways-to-index-relational-data-in-elasticsearch

Appropriate methods would be made for all these four cases.

Stage 2: Building the search and the form module.

Search Module: This module would be used for searching the indexes based on the query parameter. Various filter options can also be used to filter the data. The search endpoints in the elasticsearch-lua client shall be used. The result data would also be stored in a variable to be rendered using the page object.

Form Module: This is basically just another method in the already existing form module. Form.search(modelname, attributes, html_options). The modelname to be searched for, the column names (attributes) to be searched under can be specified in the parameters. This will be used in the search module for applying filters.

C) LuaGoogle:

This would be lua module for getting google search results programmatically. I already helped write a module named pygoogle for

PyPi and the code for the same can be found here(https://github.com/rnikhil275/pygoogle) which does a similar job. It would be made in such a way that it can be imported into the application and the results can be programmatically obtained. It will use the google Ajax API. There would be few methods bundled with the module like

1) search() which returns a table with the results as key/value pairs where title is the key and the url is the value.
2) get_urls() which returns a list of all the urls. It would just be an array
3) get_result_count() which returns the number of results
4) print_resutls() Prints all the results -> for command line interpreter usage.

The number of pages to be retrieved, safe mode, number of results to be retrieved can also be set while making the request. Everything would also be logged. It will use lua sockets for making the http requests and a json parser (probably https://github.com/grafi-tt/lunajson) for obtaining the results as a object. This object can be used for all kinds of looping and searching through it. This object will also be used for formatted printing in command line usage and also for returning to the end user in a module usage.

The module would be used in places where google search results are necessary in the backend of the application and people want to loop through them. This would also be my first publication to luarocks.

I am new to the lua community and I was really amazed by the work of the people here. I have done simple things in lua before like writing a simple script to download a particular series whenever the magnet gets posted on main chat. I never got a chance to get involved with a community at this level before and I plan on staying and contributing to the Sailor/lua community in whatever way I can. If time permits, I would also want to work on

https://github.com/sailorproject/sailor/issues/90 and then make it easy for other developers to develop modules for the Sailor project (like user-login, etc.). It would enable to create an ecosystem like http://www.yiiframework.com/extensions/ .

The priority would be finish the admin center along with the elasticsearch module first, then luagoogle, and then work with Etiene on the above. I am a beginner who wants to get into open source so I feel writing documentation is also a good way to increase my knowledge of the framework. I would also like to make changes to docs of the Sailorproject. I found a few functions to be undocumented/some are outdated and I would also like to write a tutorial explaining lua and Sailor to be published on http://lua.space/ . This tutorial would be beginner friendly tutorial to make a basic blog system in lua using Sailor.

2) Please provide a schedule with dates and important milestones/deliverables (preferably in two week increments).

The following below is a tentative timeline for developing admin center, the elasticsearch plugin and luagoogle.

April 22 – May 22	Discussing and improving the design of admin center and the integration of elasticsearch. I want to have a proper idea before I start coding of the expected outcome. All functions and routes would be finalized. Luagoogle's design would also be discussed. Discussing search case scenarios with the mentor and discussing use of plugin to store relational data in the elasticsearch index.

May 22 – June 5	Finalizing the design of the admin center and setting up basic infrastructure and architecture of the application. All the modules would be finalized and skeleton code shall be written. All the methods in Luagoogle would also be ready by now. Form module for elasticsearchlua integration would be done. All filter_options would be finalized. Skeleton code shall be written to transport data from the view to the controller and then back. Connection module to connect to the elasticsearch index would also be ready.
June 6 – June 12	Keeping secure practices in mind, the simple access module (temporarily) would be used for creating the login page. The logout function and the session features should also be working by now. The Index module would be created. It can be used in a controller. This will be done with documentation done side by side for the code.
June 13 – June 20	Continuing with the login/logout system, the dashboard would also be constructed. All autogen functions would be included and necessary checks/validation would also be performed. The necessary views would also be written. The Form module would be built. Form.search() method would be ready.

June 21 – June 28	Midterm evaluation. Will be discussing with the mentor and tweaking the existing codebase.
June 29 - July 12	The configuration file editor would be developed. Luagoogle user related documentation and testing would be done. Error reporting would also be done by now. The Search Module would be done by now. All the search endpoints would be used from elasticsearch-lua client. There would be support for filtered results also.
July 13 – July 30	Documentation and unit testing will be done side by side for the entire project. I will also write a tutorial on making a blog system using Lua and Sailor. Luagoogle should be published by now.
July 31 – August 15	Fixing any bugs, adding any minor features left out during initial phases, integration testing. Discussion with the mentor. I expect the entire admin application to be working by now. Elasticsearch-lua client should be also usable inside the sailor framework for storing models and searching through them.
August 16 - 24	Polishing of the code with suggestions with the mentor. The admin center can now be bundled with the blank-app. Elasticsearch-lua client module would be also be available as an extension in the Sailor Project.

3) What will be showable two months into the project?

After the end of two months, the admin center would be ready and functional with tests and documentation. Elasticsearch will also be fully integrated with the Sailor framework and most kind of models could be indexed. User documentation for this would also be written. Luagoogle would also be published in luarocks.

GSOC

1) Have you participated to GSoC before? If so, how many times, which year, which project?

No, I haven't participated in GSoc before.

2) Have you applied but were not selected? When?

Yes. I applied in 2015 to develop a simple Nodejs application which takes geolocation data through an endpoint and then plots it on a map using the google maps API. The project wasn't selected but the application was finished in the first month itself.

3) Did you apply this year to any other organizations?

No, I did not apply to any other organization. I really liked the community and decided to stick with them.