CMPE-273
LAB 1

HANDSHAKE PROTOTYPE

- Kapil Mulchandani
(014501396)

HANDSHAKE JOB SEARCH PORTAL

Google Drive Link :
https://drive.google.com/file/d/1L95DO1UOfwkWG2m23_-QXUjAc_DVMhkC/view?usp=sharing

AWS Link :
http://handshake-kapil.s3-website-us-east-1.amazonaws.com/

Github Link:
https://github.com/kapilmulchandani/Handshake_Lab1

System Goals for Handshake Job Portal Application:
The Handshake project is aimed at developing a job search portal for potential students to find potential companies that provide jobs, along with the option to register for events.
The students can apply to the jobs by uploading their resume, RSVP for events, see their applied jobs and events in a separate section.
The companies can post a job, post an event, get the list of students who applied for the job along with students who registered for a specific event.

Purpose of the System:
The system consists of 2 major parts:
1) Student's Part.
2) Company's Part.

The 2 parts of system serve different purposes:
- The Student's part of the system allows students to search for jobs and events and let them filter the job searches based on the full time/

internship/ part-time or on campus. The Search Jobs functionality can be
accessed after student is logged in to the system. Along with this Search Jobs functionality, events can also be searched and registered after the student is logged in.The new students can sign-up to the system using sign-up form. Also, students can view and update their profile in the system.

- The Company's part of the system allows the companies to post jobs and events to the system. The companies can also view the list of students who applied for their job with the option of previewing the resume and have an option of changing the status of the application. Companies can view the profiles of students who have applied for their job. Also, companies can view and update their profile in the system.

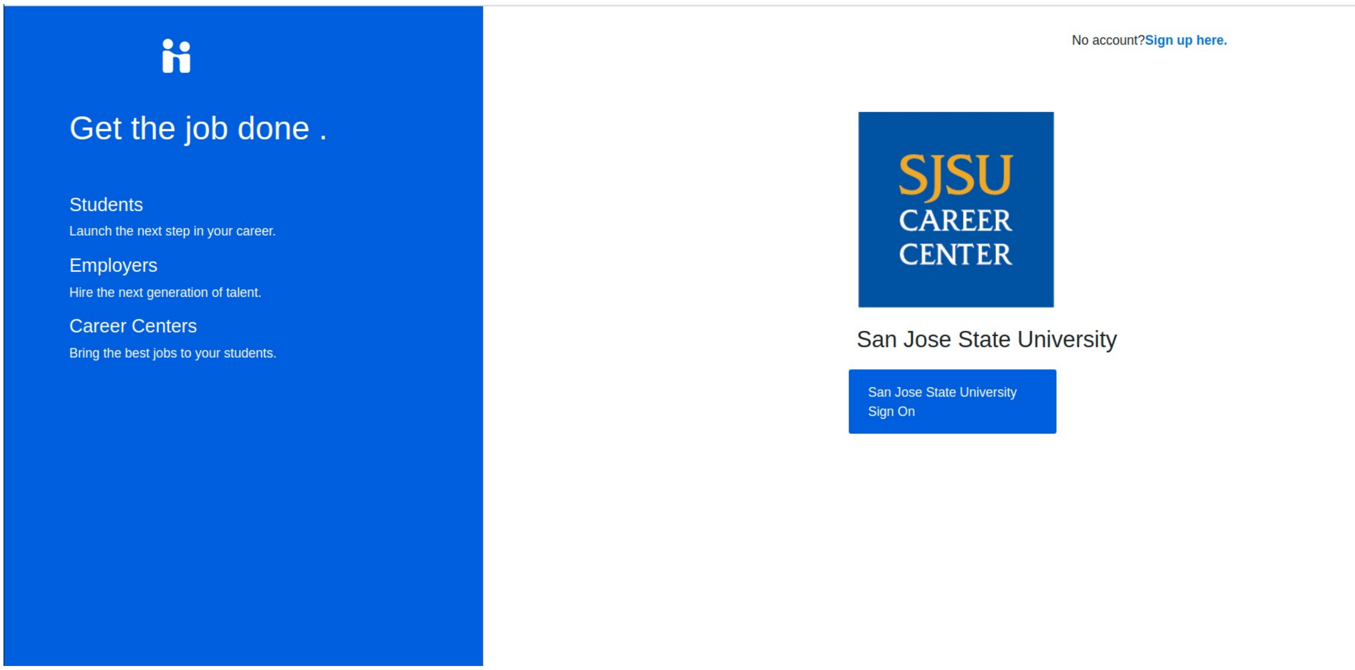
System Design:
The system here uses a 3-tier architecture where client side uses ReactJS, HTML5 to manage User Interactions. Backend uses NodeJS for interacting with database. MySQL is the database for this system. The client and servers interaction happens via RESTful API's.
----diagram-----

- The website should make use of ReactJS, HTML5 for interactive frontend performance
and PrimeReact/React-Bootstrap for better User Interface.

- The backend server is implemented in NodeJS using express framework as middleware which takes the Rest API calls from the frontend part.
- The MySQL Database also provides inbuilt connection pooling mechanism to serve the optimization in database query calls.
- Authentication is achieved by using session and cookies in the system. Also, the passwords are encrypted for owners and buyers using bcrypt library for safety measures.

Screenshots the application:

1. Starting page



Get the job done .

Students
Launch the next step in your career.

Employers
Hire the next generation of talent.

Career Centers
Bring the best jobs to your students.

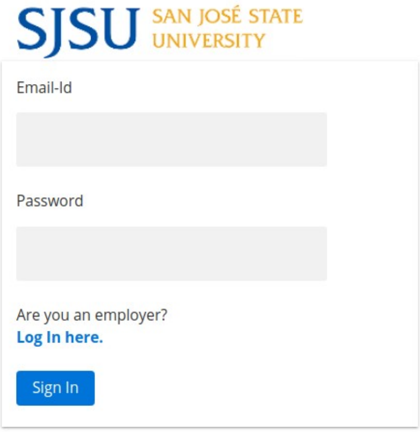
No account?**Sign up here.**

SJSU CAREER CENTER

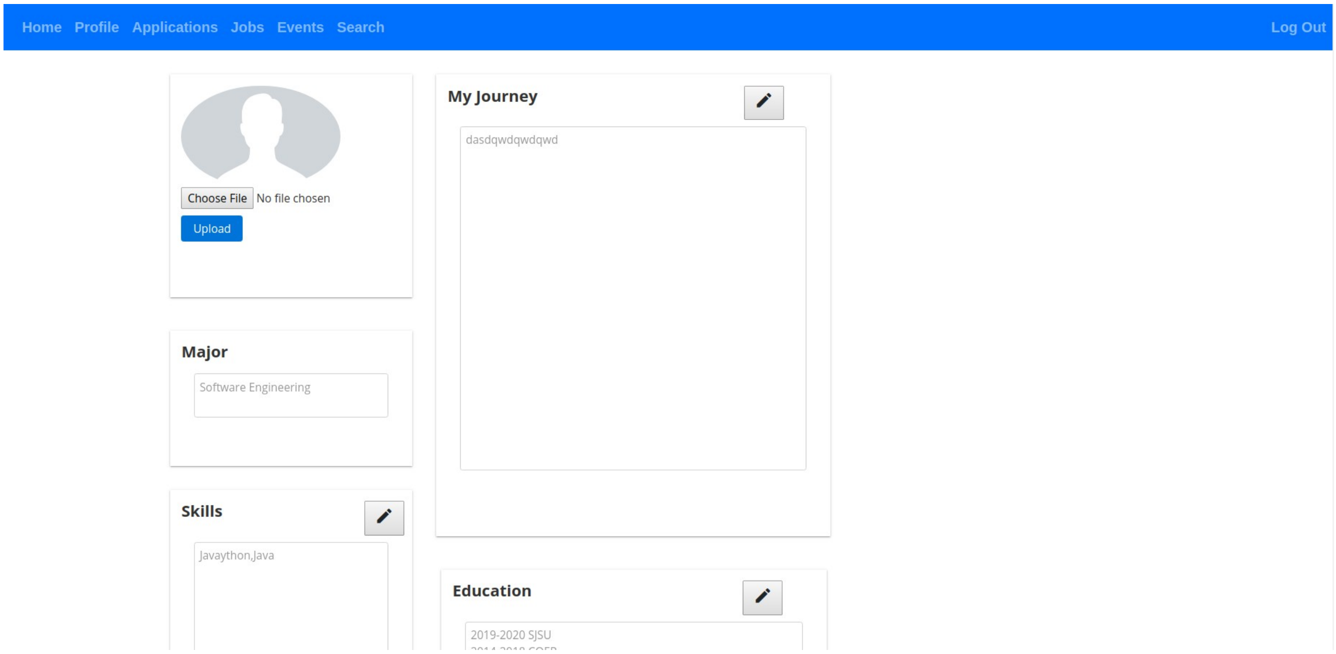
San Jose State University

San Jose State University Sign On

2. Login Page



3. Student profile



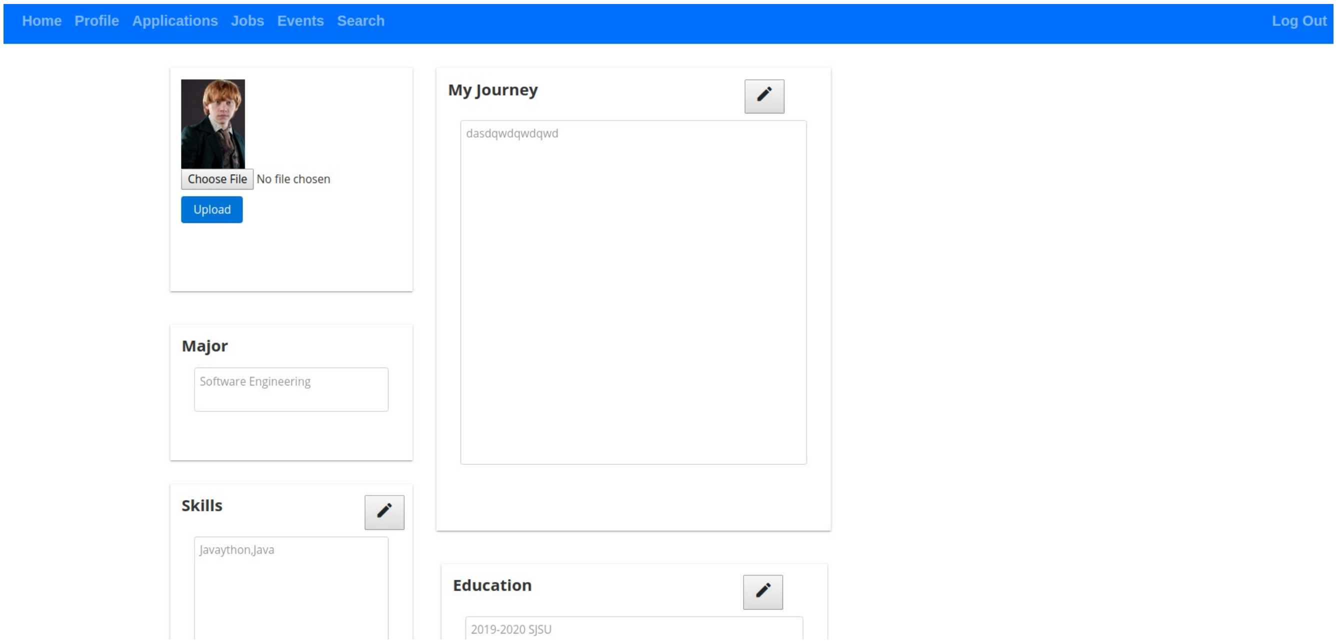
My Journey: dasdqwdqwdqwd

Major: Software Engineering

Skills: Javaython,Java

Education: 2019-2020 SJSU

4. Student upload profile picture



My Journey

dasdqwdqwdqwd

Major

Software Engineering

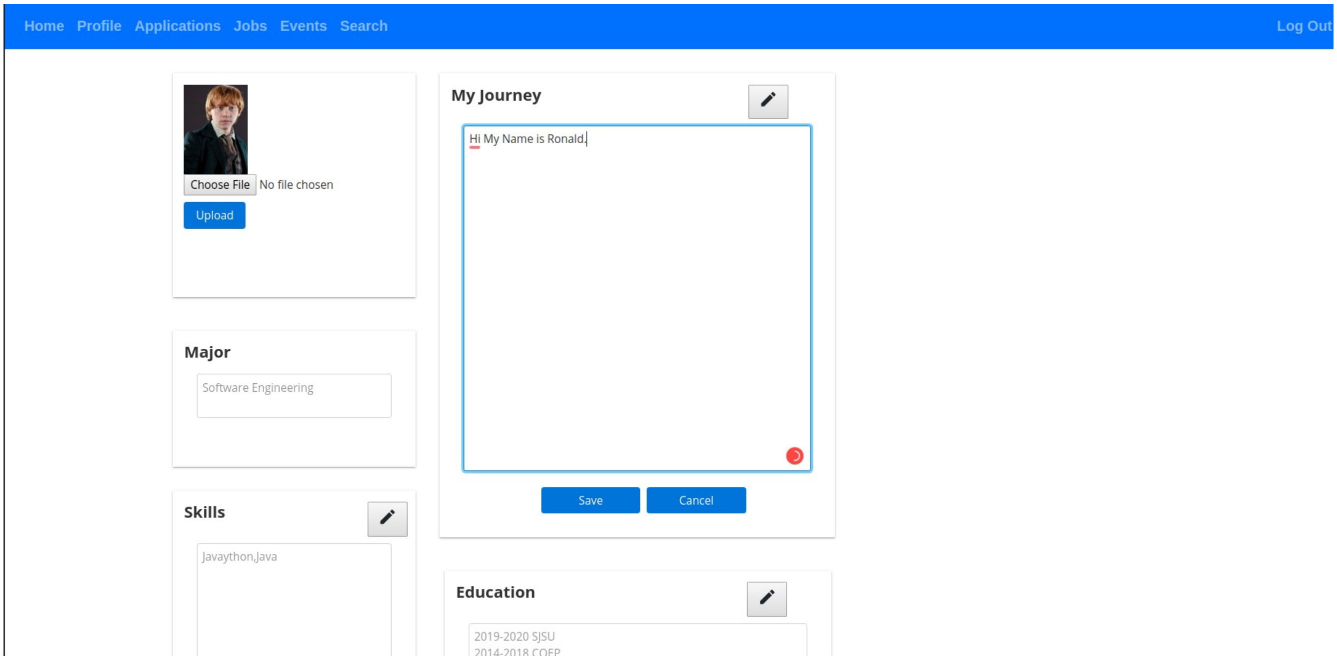
Skills

Javaython,Java

Education

2019-2020 SJSU

5. Edit profile



My Journey

Hi My Name is Ronald.

Major

Software Engineering

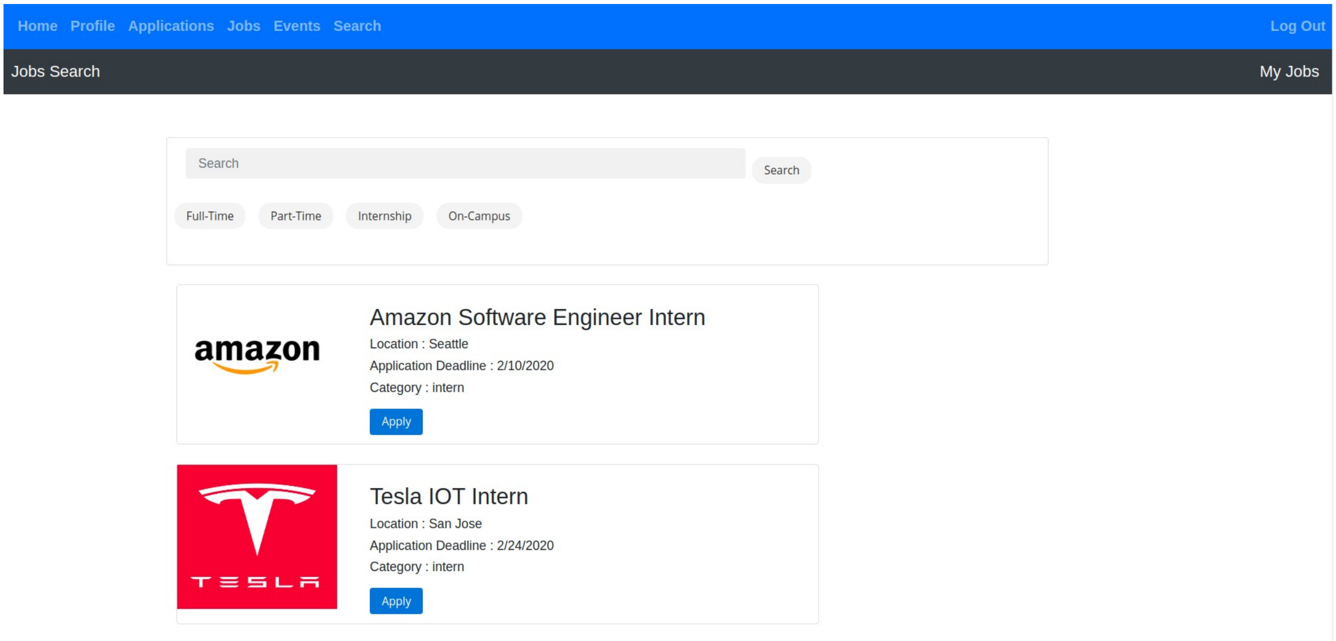
Skills

Javaython,Java

Education

2019-2020 SJSU
2014-2018 COEP

6. Job Search



Amazon Software Engineer Intern

Location : Seattle

Application Deadline : 2/10/2020

Category : intern

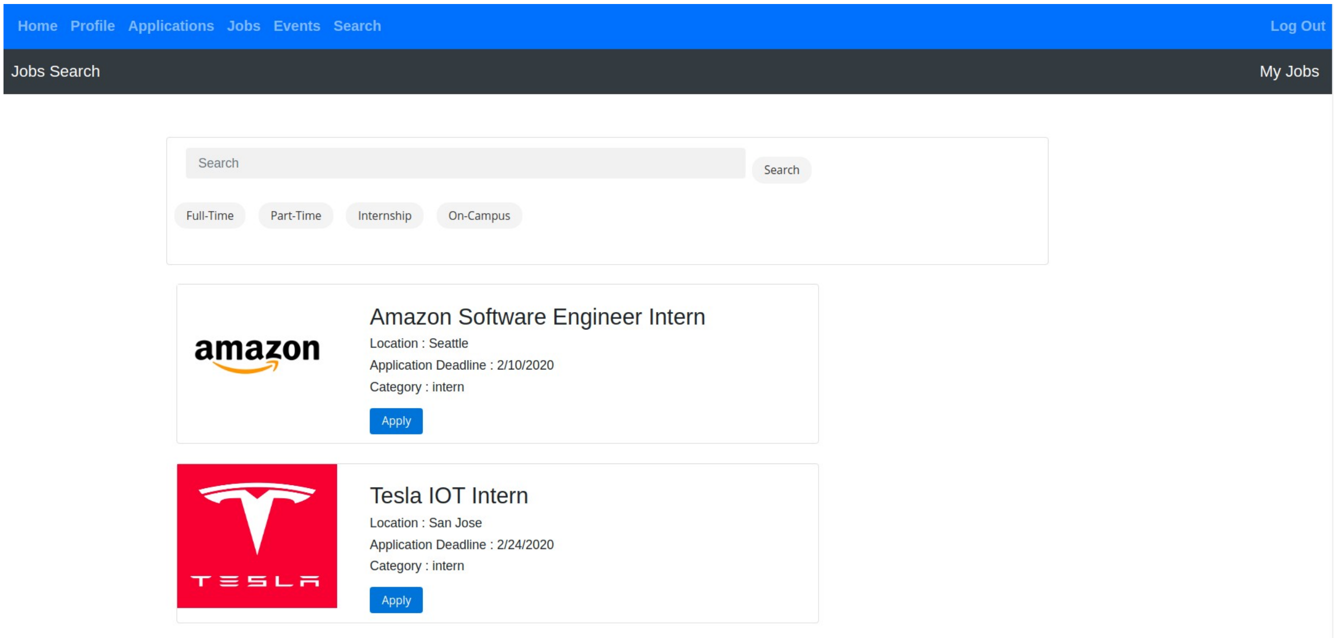
Tesla IOT Intern

Location : San Jose

Application Deadline : 2/24/2020

Category : intern

7. Events Registered by students



Amazon Software Engineer Intern

Location : Seattle

Application Deadline : 2/10/2020

Category : intern

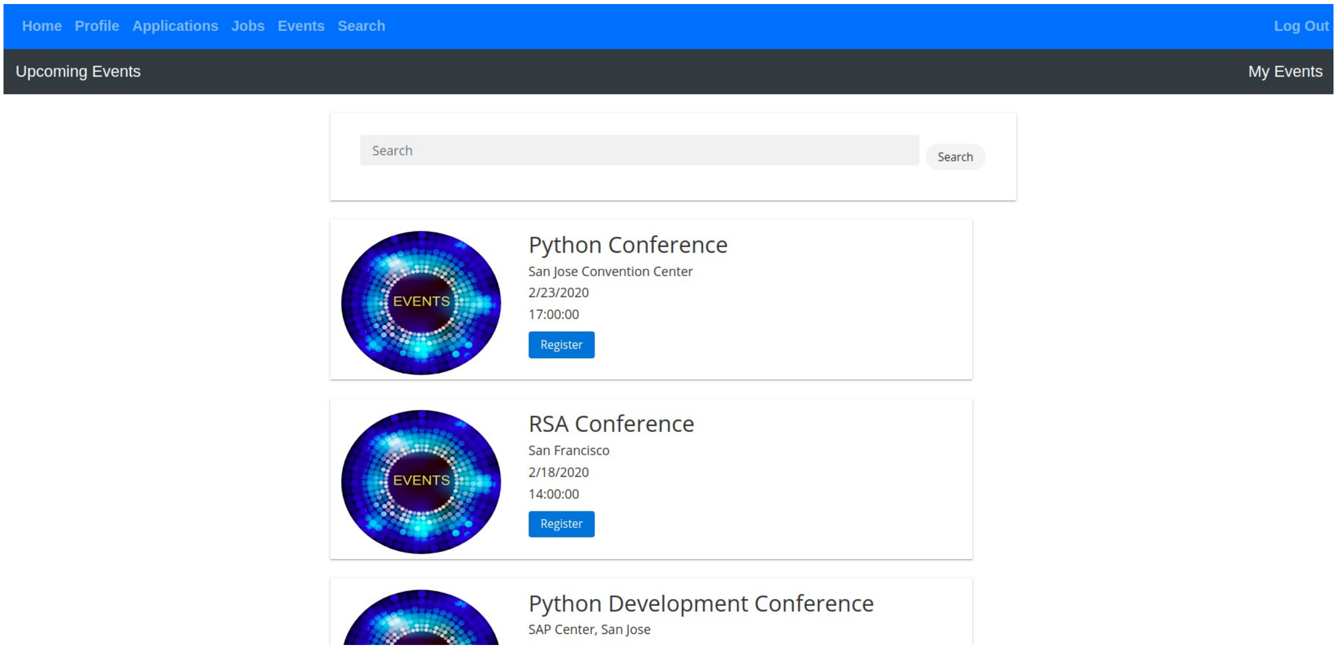
Tesla IOT Intern

Location : San Jose

Application Deadline : 2/24/2020

Category : intern

8. UpComing Events



Python Conference

San Jose Convention Center

2/23/2020

17:00:00

RSA Conference

San Francisco

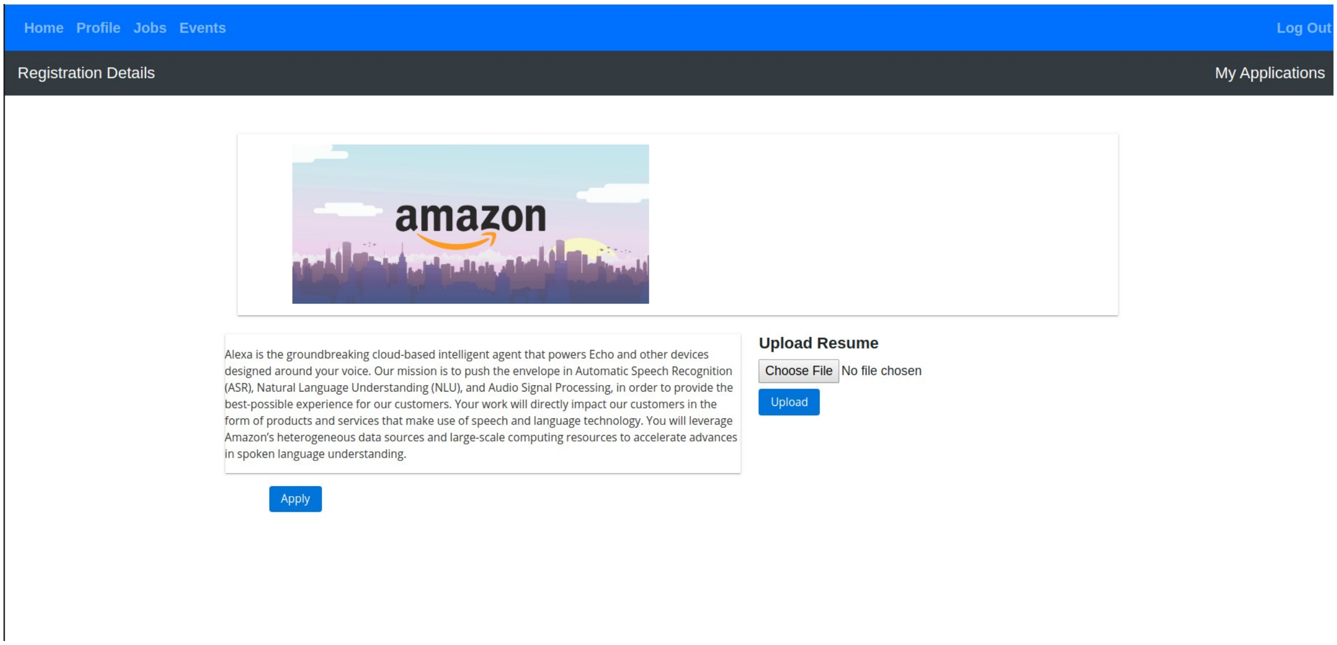
2/18/2020

14:00:00

Python Development Conference

SAP Center, San Jose

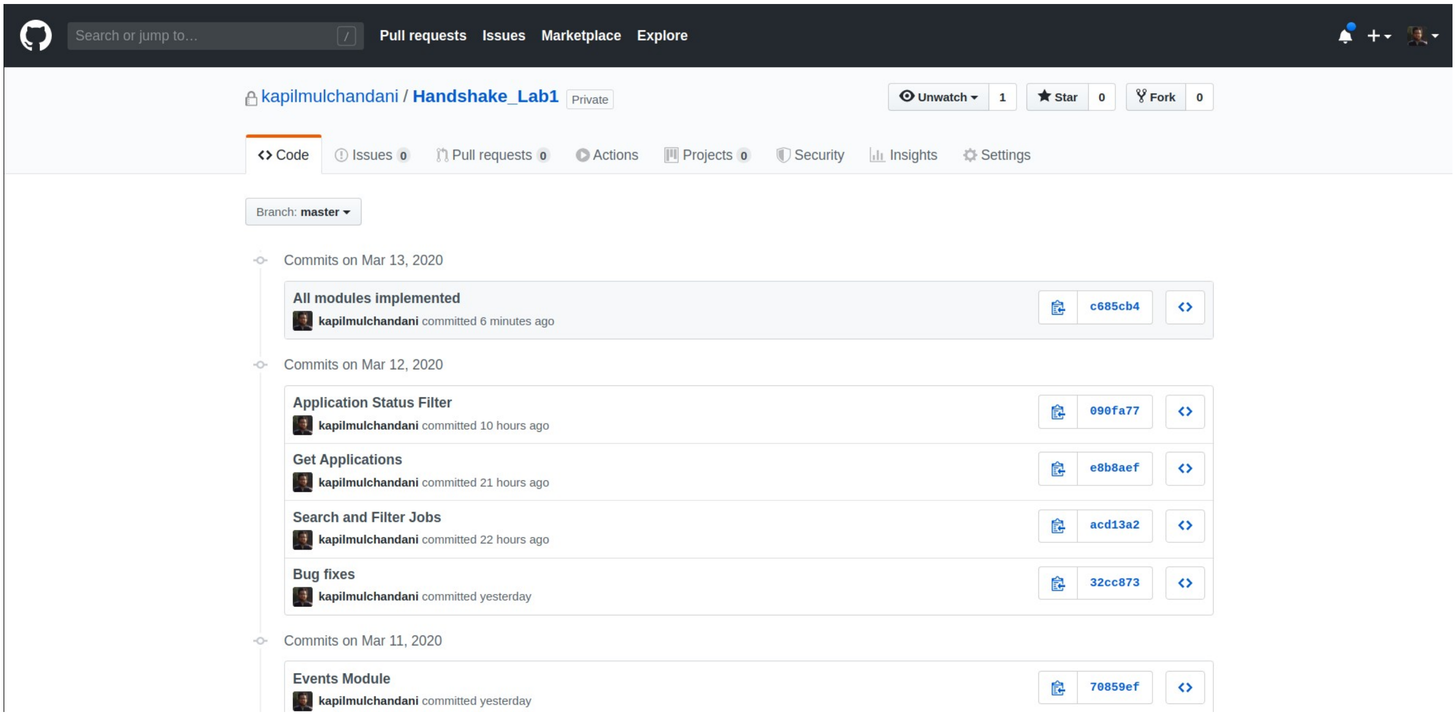
9. Apply for Job



Alexa is the groundbreaking cloud-based intelligent agent that powers Echo and other devices designed around your voice. Our mission is to push the envelope in Automatic Speech Recognition (ASR), Natural Language Understanding (NLU), and Audio Signal Processing, in order to provide the best-possible experience for our customers. Your work will directly impact our customers in the form of products and services that make use of speech and language technology. You will leverage Amazon's heterogeneous data sources and large-scale computing resources to accelerate advances in spoken language understanding.

]

GitHub Commit History:



Commits on Mar 13, 2020

All modules implemented
kapilmulchandani committed 6 minutes ago — c685cb4

Commits on Mar 12, 2020

Application Status Filter
kapilmulchandani committed 10 hours ago — 090fa77

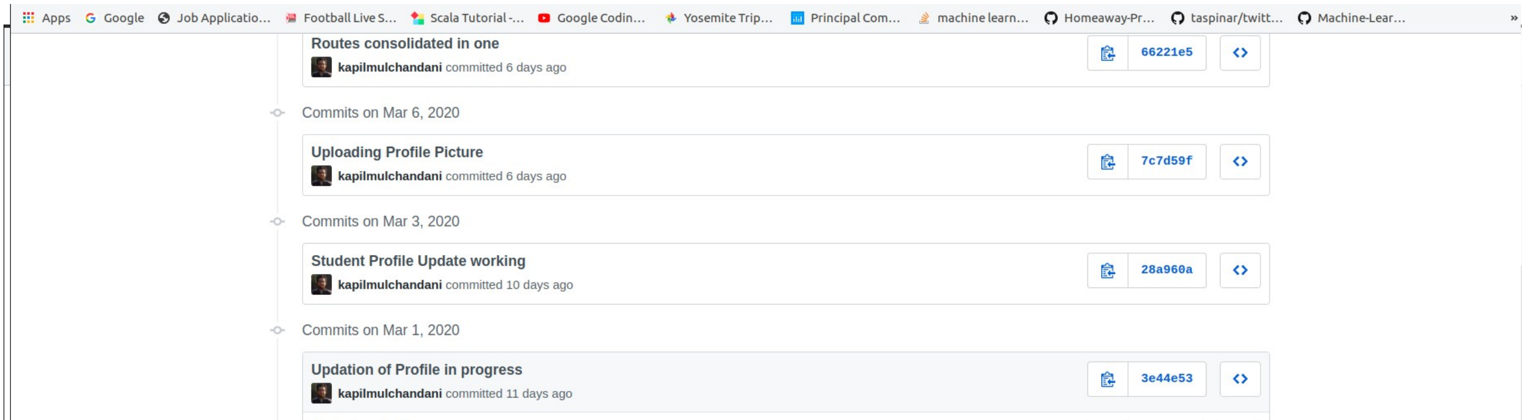
Get Applications
kapilmulchandani committed 21 hours ago — e8b8aef

Search and Filter Jobs
kapilmulchandani committed 22 hours ago — acd13a2

Bug fixes
kapilmulchandani committed yesterday — 32cc873

Commits on Mar 11, 2020

Events Module
kapilmulchandani committed yesterday — 70859ef



Routes consolidated in one
kapilmulchandani committed 6 days ago — 66221e5

Commits on Mar 6, 2020

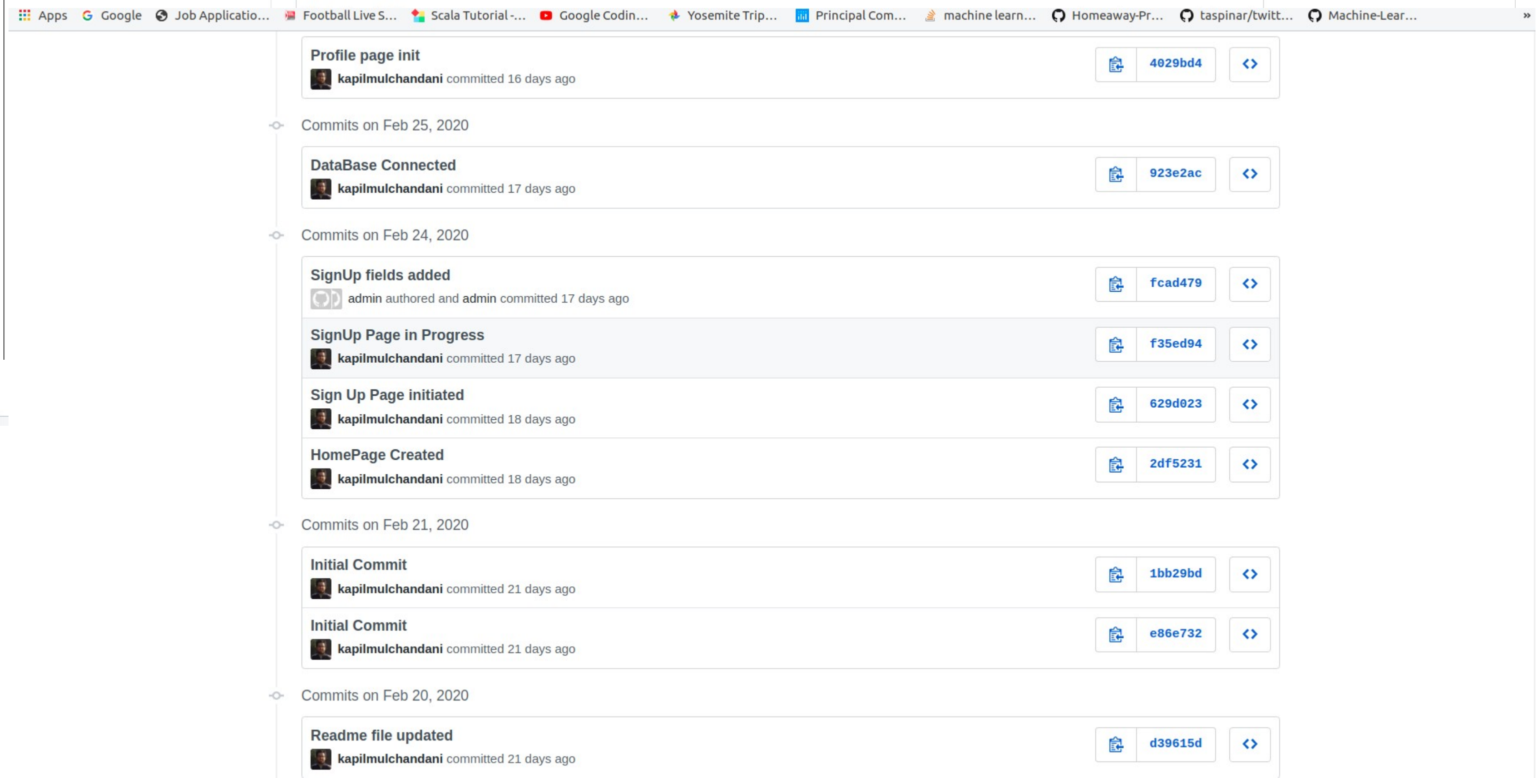
Uploading Profile Picture
kapilmulchandani committed 6 days ago — 7c7d59f

Commits on Mar 3, 2020

Student Profile Update working
kapilmulchandani committed 10 days ago — 28a960a

Commits on Mar 1, 2020

Updation of Profile in progress
kapilmulchandani committed 11 days ago — 3e44e53



Profile page init
kapilmulchandani committed 16 days ago — 4029bd4

Commits on Feb 25, 2020

DataBase Connected
kapilmulchandani committed 17 days ago — 923e2ac

Commits on Feb 24, 2020

SignUp fields added
admin authored and admin committed 17 days ago — fcad479

SignUp Page in Progress
kapilmulchandani committed 17 days ago — f35ed94

Sign Up Page initiated
kapilmulchandani committed 18 days ago — 629d023

HomePage Created
kapilmulchandani committed 18 days ago — 2df5231

Commits on Feb 21, 2020

Initial Commit
kapilmulchandani committed 21 days ago — 1bb29bd

Initial Commit
kapilmulchandani committed 21 days ago — e86e732

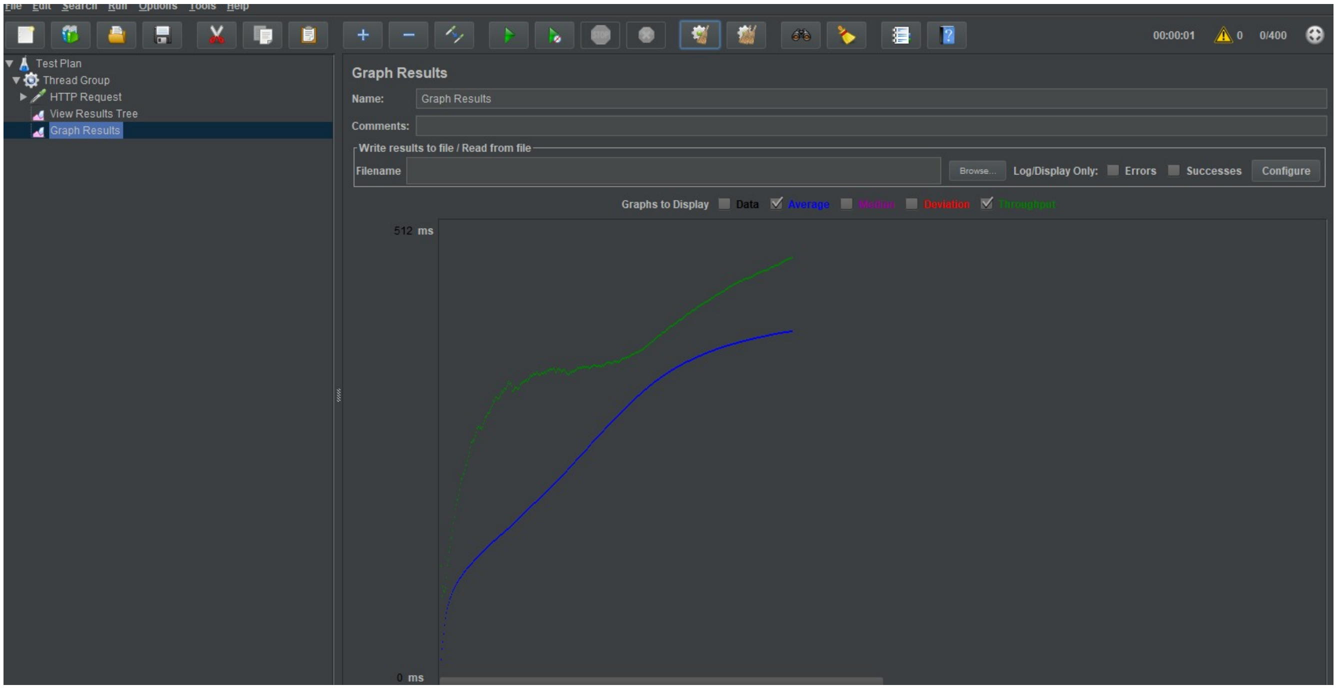
Commits on Feb 20, 2020

Readme file updated
kapilmulchandani committed 21 days ago — d39615d

Questions :

Q.1) Result of connection-pooling and algorithm of connection-pooling:

100 Without Pool :



Graph Results

Name: Graph Results

Comments:

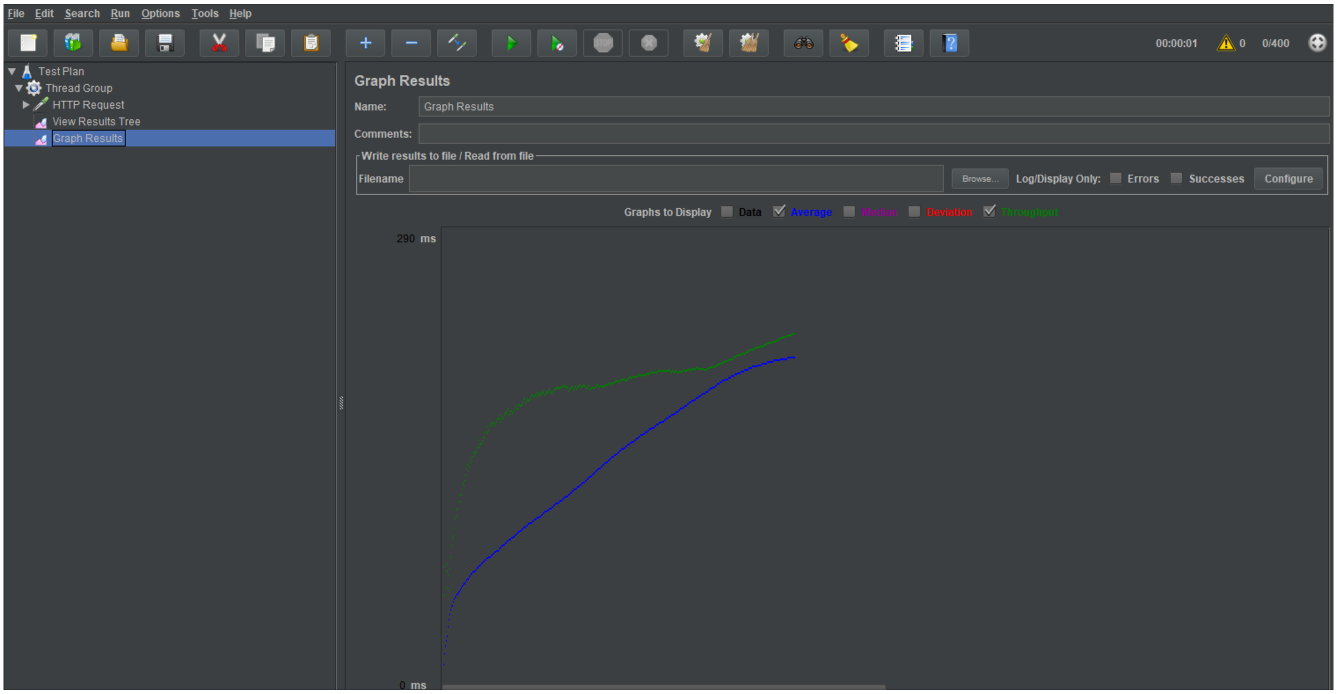
Write results to file / Read from file

Filename

Log/Display Only: □ Errors □ Successes

Graphs to Display □ Data ☑ Average □ Median □ Deviation ☑ Throughput

100 With Pool :



Graph Results

Name: Graph Results

Comments:

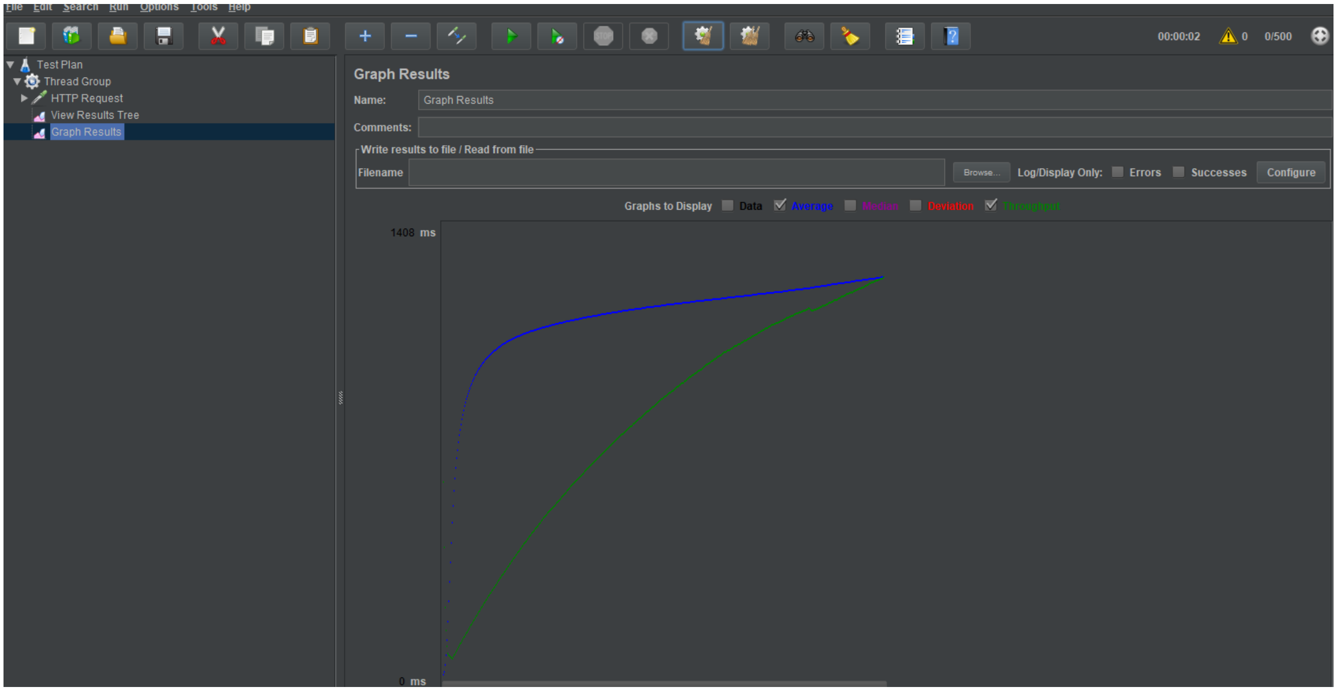
Write results to file / Read from file

Filename

Log/Display Only: □ Errors □ Successes

Graphs to Display □ Data ☑ Average □ Median □ Deviation ☑ Throughput

200 Without Pool :



Graph Results

Name: Graph Results

Comments:

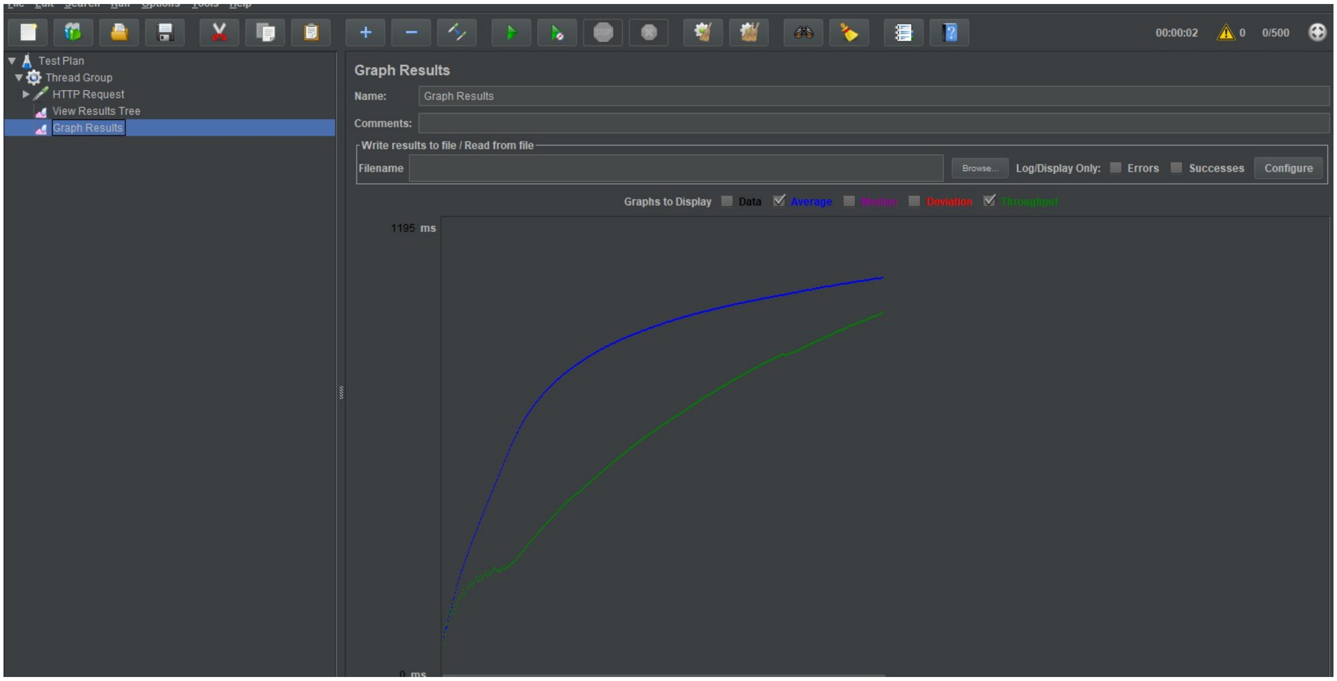
Write results to file / Read from file

Filename

Log/Display Only: □ Errors □ Successes

Graphs to Display □ Data ☑ Average □ Median □ Deviation ☑ Throughput

200 With Pool :



Graph Results

Name: Graph Results

Comments:

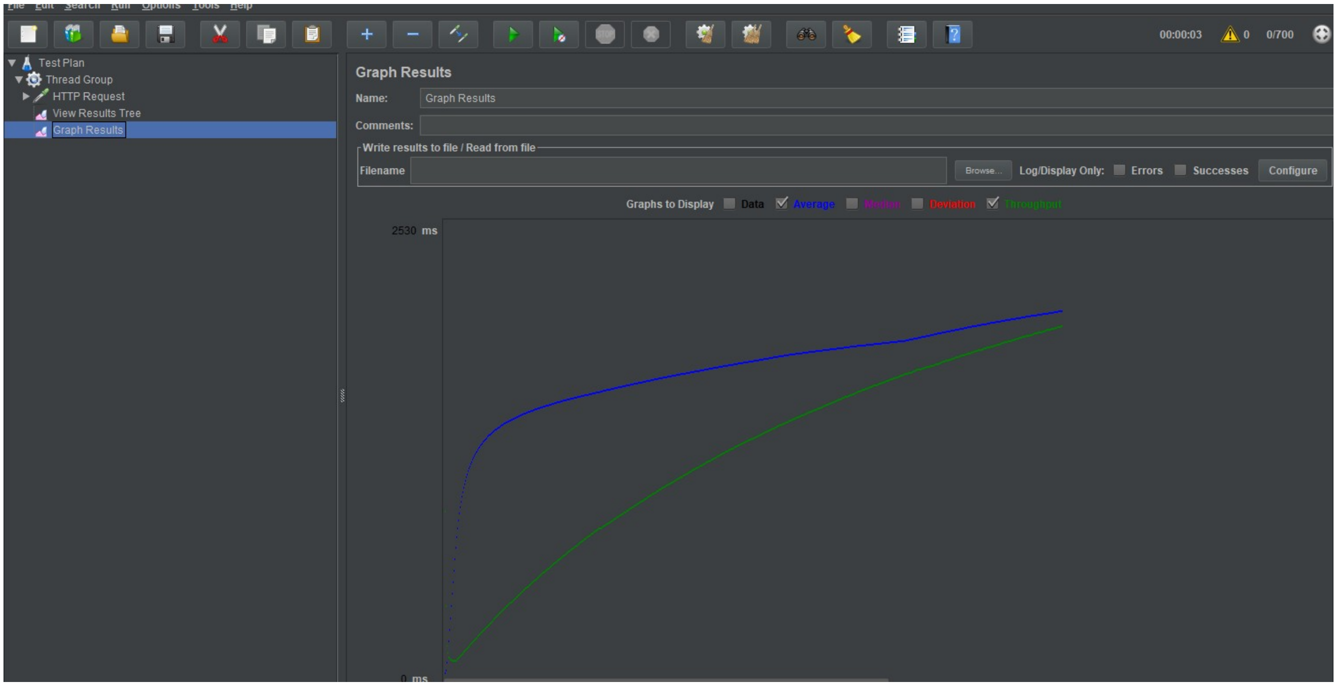
Write results to file / Read from file

Filename

Log/Display Only: □ Errors □ Successes

Graphs to Display □ Data ☑ Average □ Median □ Deviation ☑ Throughput

300 Without Pool :



Graph Results

Name: Graph Results

Comments:

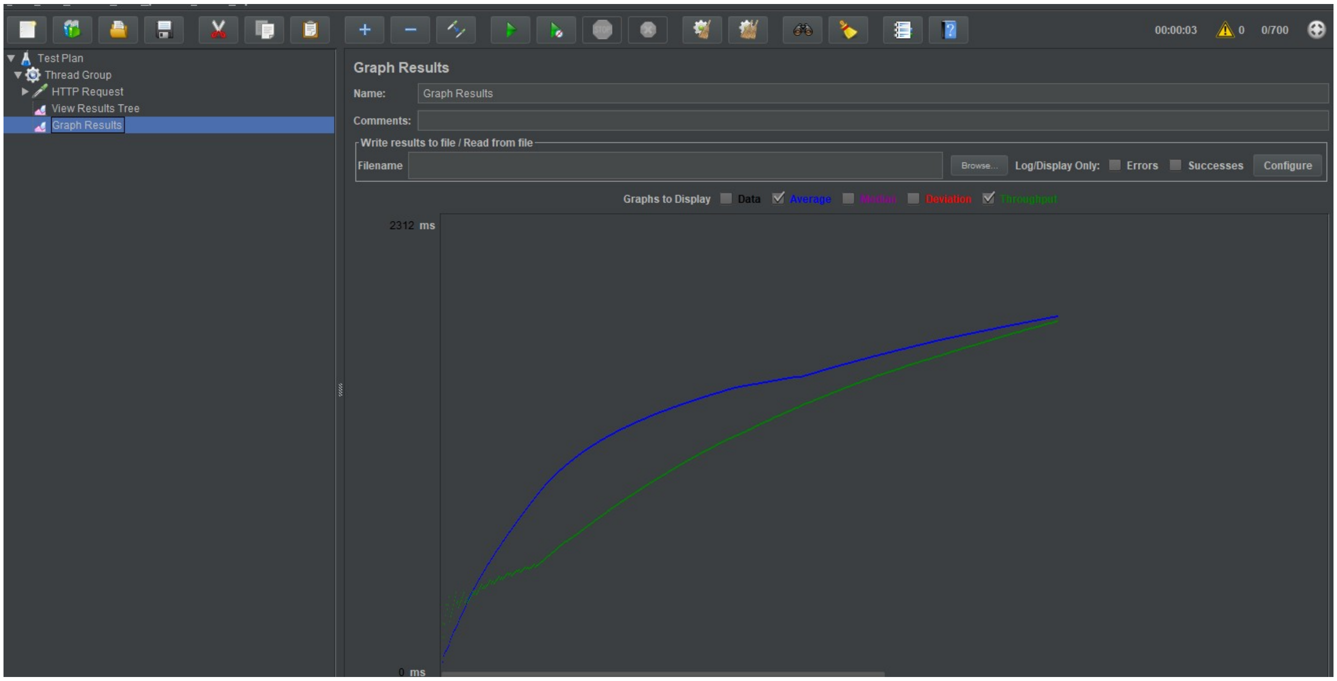
Write results to file / Read from file

Filename

Log/Display Only: □ Errors □ Successes

Graphs to Display □ Data ☑ Average □ Median □ Deviation ☑ Throughput

300 With Pool:



Graph Results

Name: Graph Results

Comments:

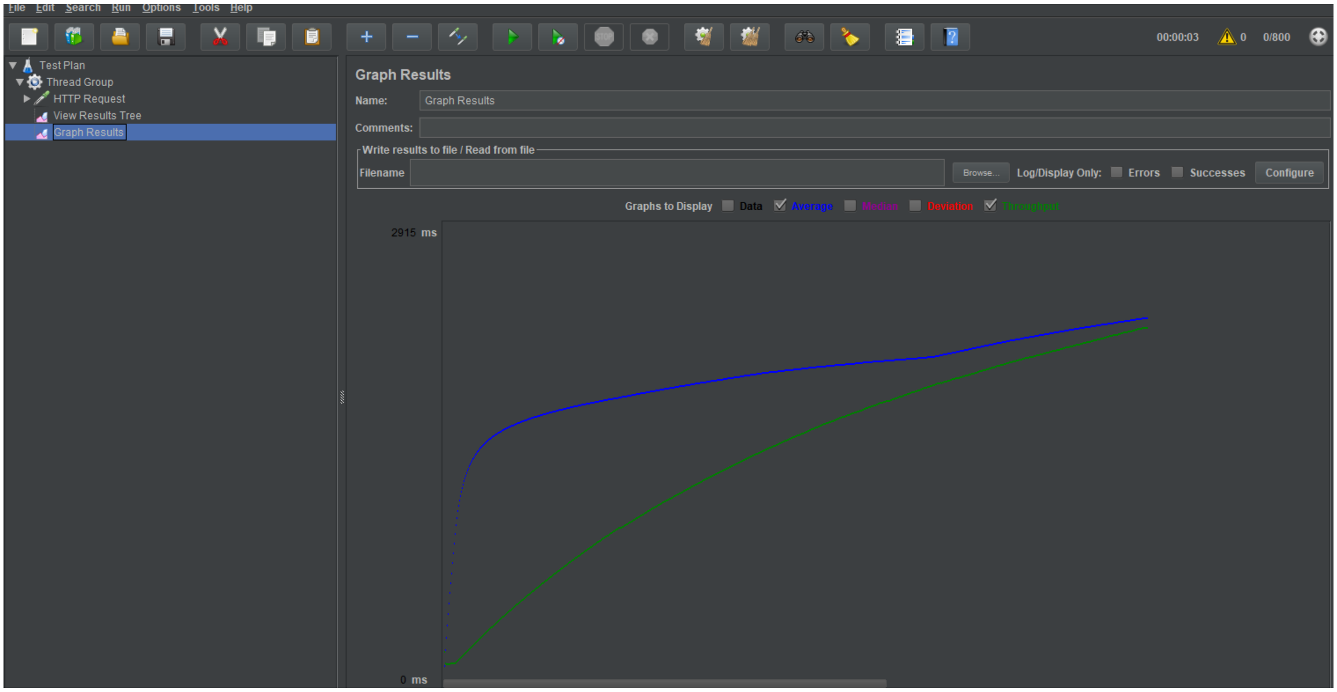
Write results to file / Read from file

Filename

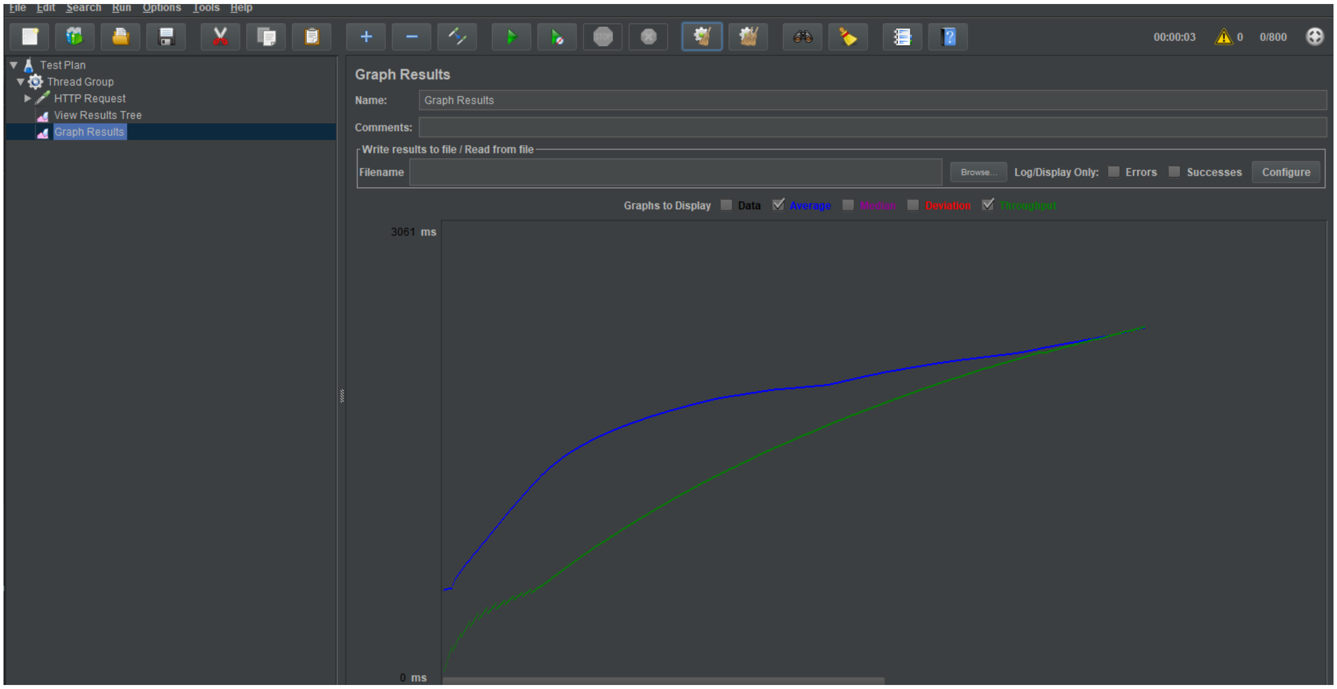
Log/Display Only: □ Errors □ Successes

Graphs to Display □ Data ☑ Average □ Median □ Deviation ☑ Throughput

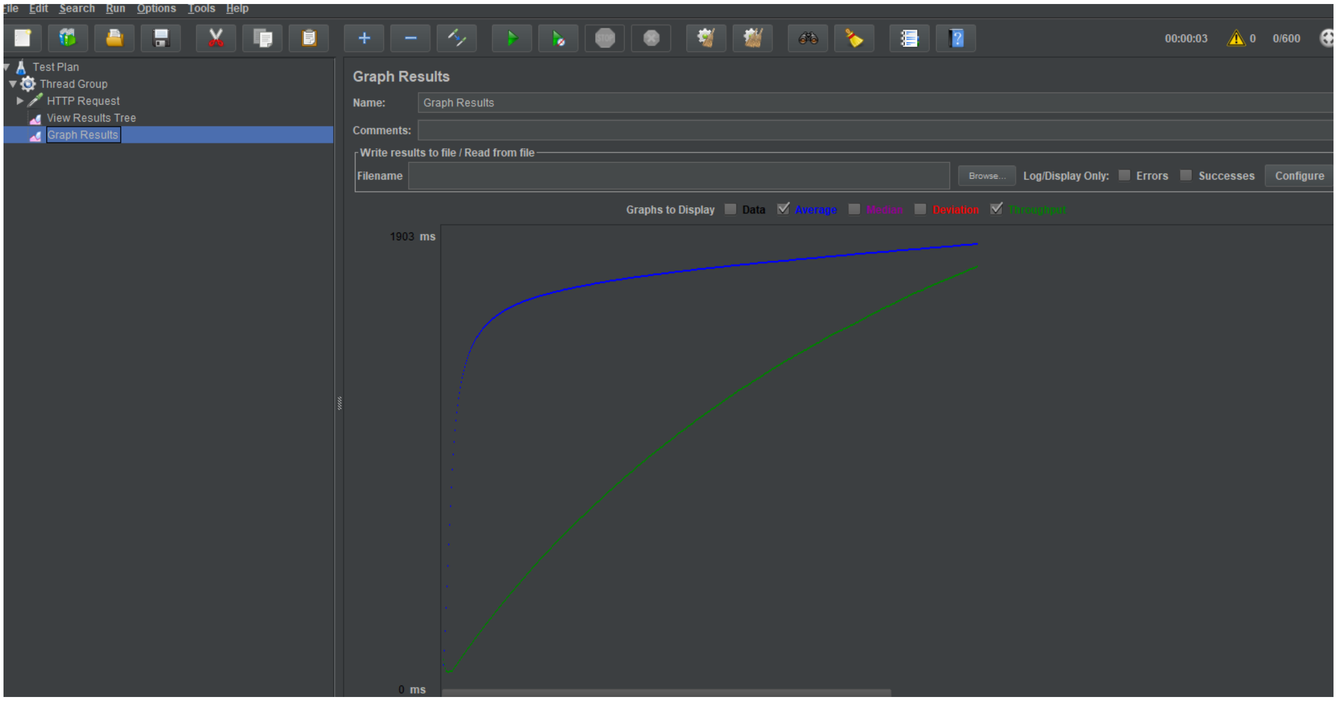
400 Without Pool:



400 With Pool :



500 Without Pool :



Graph Results

Name: Graph Results

Comments:

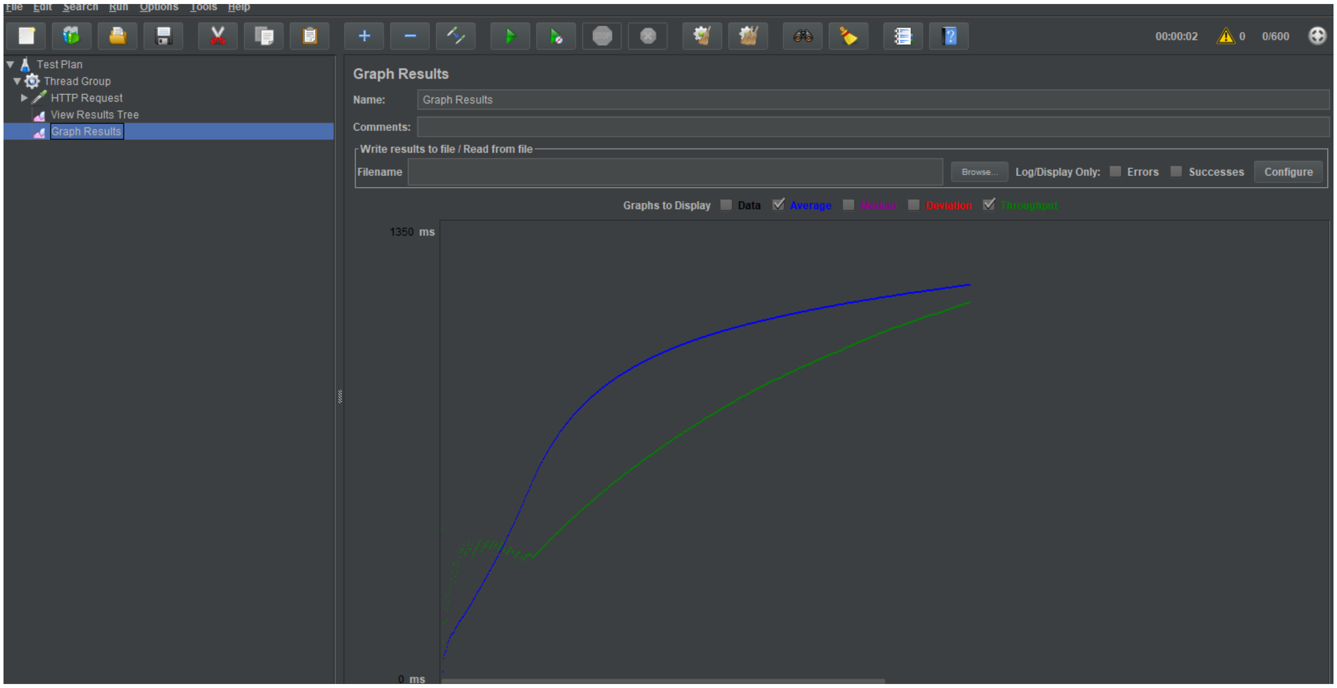
Write results to file / Read from file

Filename

Log/Display Only: □ Errors □ Successes

Graphs to Display □ Data ☑ Average □ Median □ Deviation ☑ Throughput

500 With Pool:



Graph Results

Name: Graph Results

Comments:

Write results to file / Read from file

Filename

Log/Display Only: □ Errors □ Successes

Graphs to Display □ Data ☑ Average □ Median □ Deviation ☑ Throughput

In Connection pooling, the client has and maintains a pool of connections, so that each thread that needs to use a connection will checkout once and return it to pool when done using.
If not using connection pooling, everytime a new connection is requested a check is made to see if an ‘identical’ connection is already open.
Connection pooling is by default reusing the connection that was already created, instead of making a new connection everytime . This reduces the number of times a new connection has to be opened.

If connection-pool is enabled:
request → database → connection estabilished.
When Connection is disposed, connection goes to the pool and is alive.

Another request → database → New connection is not created but the connection that was created previously is used.

If 2 requests come at a time and the connection pool limit is 1 then 1 connection is reused by one request and for other request, new connection is created.

Q.2) Ways to improve SQL performance :

I) Caching
Query caching can provide a boost to the performance of SQL for a high read environment and can be included without any application overhead. This technique may affect the performance of environments with heavy write to read ratios.

II) Avoid using Select *
One should stop using select * in queries for getting data and instead start using the field names that are needed for the results. Select * retrieves unnecessary data as well as increases the network traffic used for our queries.

III) Locate Problem Queries

There are a lot of queries in any application and about 5-10 queries of these queries are problem queries and these result in upto 80-90% of poor SQL performance. So if performance of SQL is down, be sure to check problem queries in your application.

Q.3) Session Strategy : horizontal scaling.

No, in my case horizontal scaling won't work but it can work if, we opt for this technique where each instance is responsible for only some part of the application's data. This partitioning requires a step that needs to be performed before each operation to let it know which instance of the application to use. Another way of horizontal scaling could be to spawn one process of each core of one's machine.