**Software Definition**: Dropbox

**Goal**:

To provide end user, a user friendly application to store their data online, use and share it effectively.

**Purpose**

The purpose of this application is providing a system which allows user to put their file online and access it from anywhere.

It also allows user to download the file when they needed. System also provide the facility to share the files. Use can also delete the previously uploaded files.

**System Design:**

We are using **partial MERN Stack** where MySQL took place of mongo DB.

**Node JS** is used for the server implementation. The CORS technique is used to separate the client and server and still maintains integrity.

We have app.js as controller of the server which actually catches all the comming request and redirect them to the particular handler.

Also, we are using client session technique to track that the authenticity of the request calls.

Modularity is implemented in the system by putting proper code in the proper directory. This also improves the readability of the system.

Each functions are made as small small API calls for reusing purpose to reduce the code redundancy.

On Client Side, we are using **React JS.** The purpose of using react js is to take advantage of unidirectional data flow in UI and partial rendering technique. It allows use to render a small part of DOM where the change is needed. Also the changes made are really quick as they are event driven.

The CORS technique allows the React client to send data to the Node Server. In reach Js we have a request sender named API.js which is taking care of all the request sending and response gathering from the node server.

Bootstrape and material UI is used to make the system more interactive.

**MySql** is used as database which interact with node server. We are using mysql dependency in node js to connect which the mysql database.

The database contains tables for each entity and each functionality. 3NF normalization is applied to the tables.

For making system more efficient, **connection pooling** technique is used with 100 connection available in pool.

For more secure transaction system is using **prepared statement** to avoid sql injections.

Also generalize API

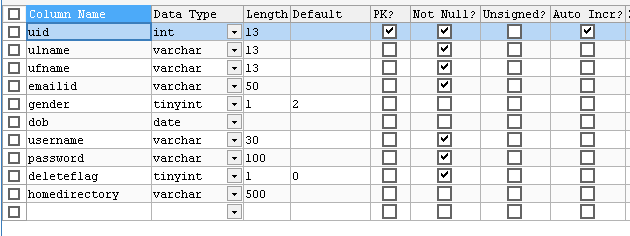
For **Authenticity**, The system is using one server interceptor to authenticate each and every request. If requests is authenticated, it will be redirected to handler else the unauthorized access status will be send to the client.

While storing passwords into the database, **bCrypt** encryption is applied to make passwords unreadable.

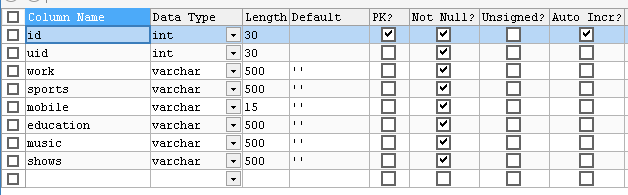
**NodeMailer** is used to send sharing notification to the users.

**Database Designs:**

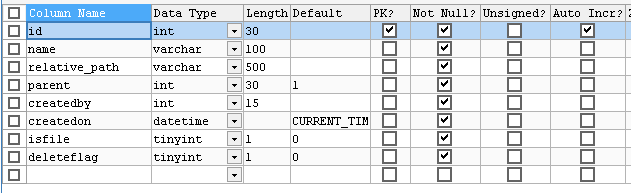
**Users:** To store user signup information



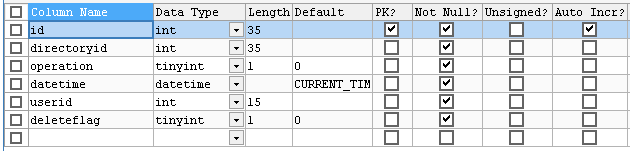
**UserProfile**: Used to store user profile related information



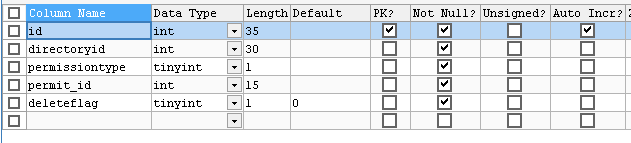
**Directories**: The tables if used to keep track of directories in the system.



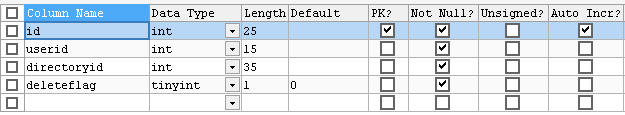
**Directory\_Logging**: this table is used to store the operations performed on the directory by particular user



**Directory\_permission**: This table is used to store directory sharing information



**Staerddir**: This table is used to store starring information of directory by user



**System Flow:**

The system starts with the login/Signup pages.

**Sign Up**:  This is the functionality which a user can use to register themselves with the system to use its functionality.

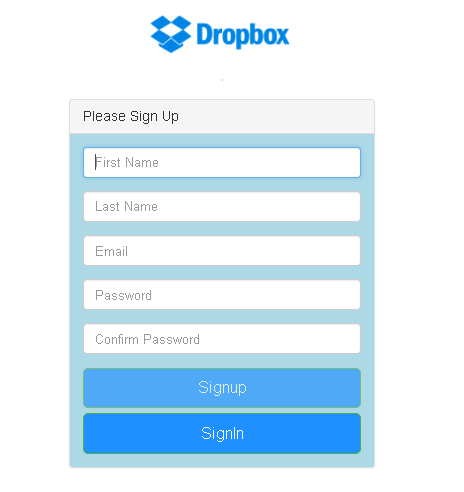
The users will have a screen where they have to insert some basic information such as first name, last name, username, email address, password.

After successfully inserting data, user clicks on the Signup button and the system is doing basic validation on the values entered by the user.

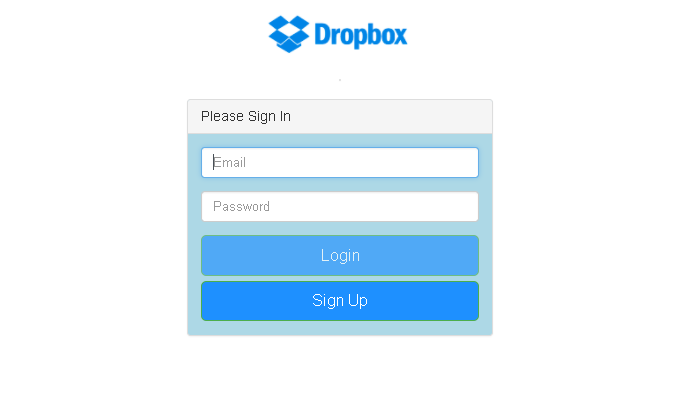
If all data are correct, the system is generating a new account for the user and redirect them on the signin page to use the system.

As the password is the most crucial thing,It's encrypted before storing in the database.

Tables Used to store information: **users**



**Sign In:**



This is the entry page to the system. User must have to login to access the application.

On this page, user have to insert username and password for validation.

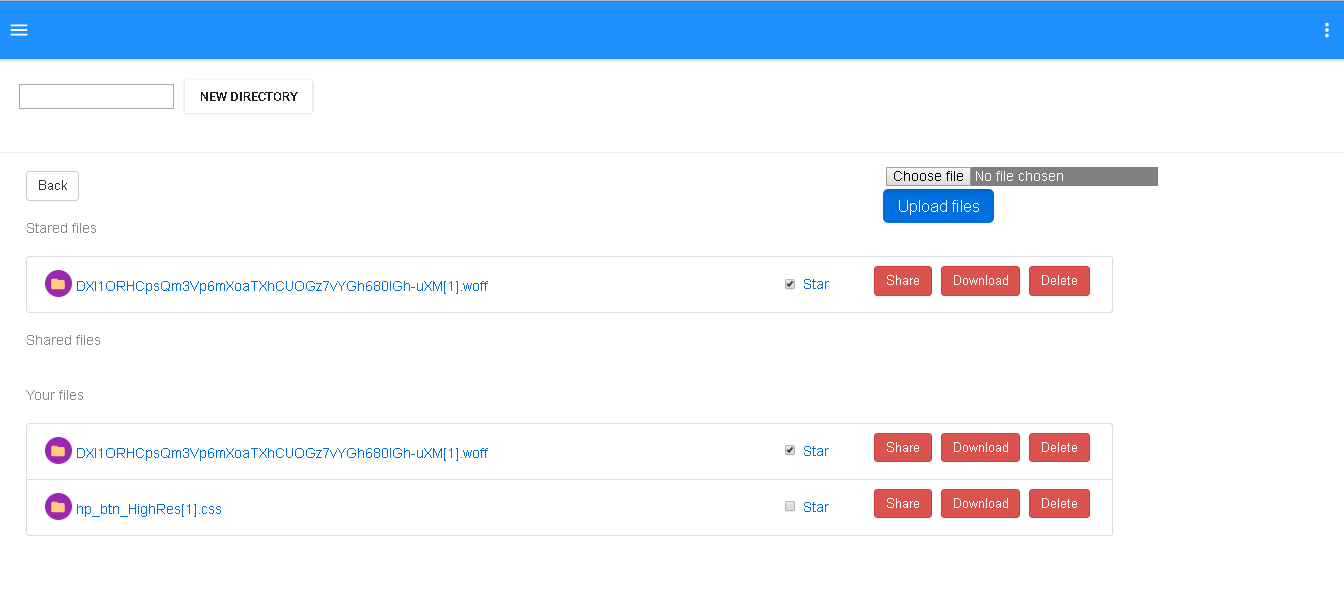
After successfully inserting the information, user clicks on sign in button and the server validates the inputs entered by the user.

Then the system matches the data with the available records in the database. If it matches , it allows the user to get into the system and use the feature.

The system then redirects the user to the welcome/home page.

Tables Used to get information: **users**

**Home Page:**

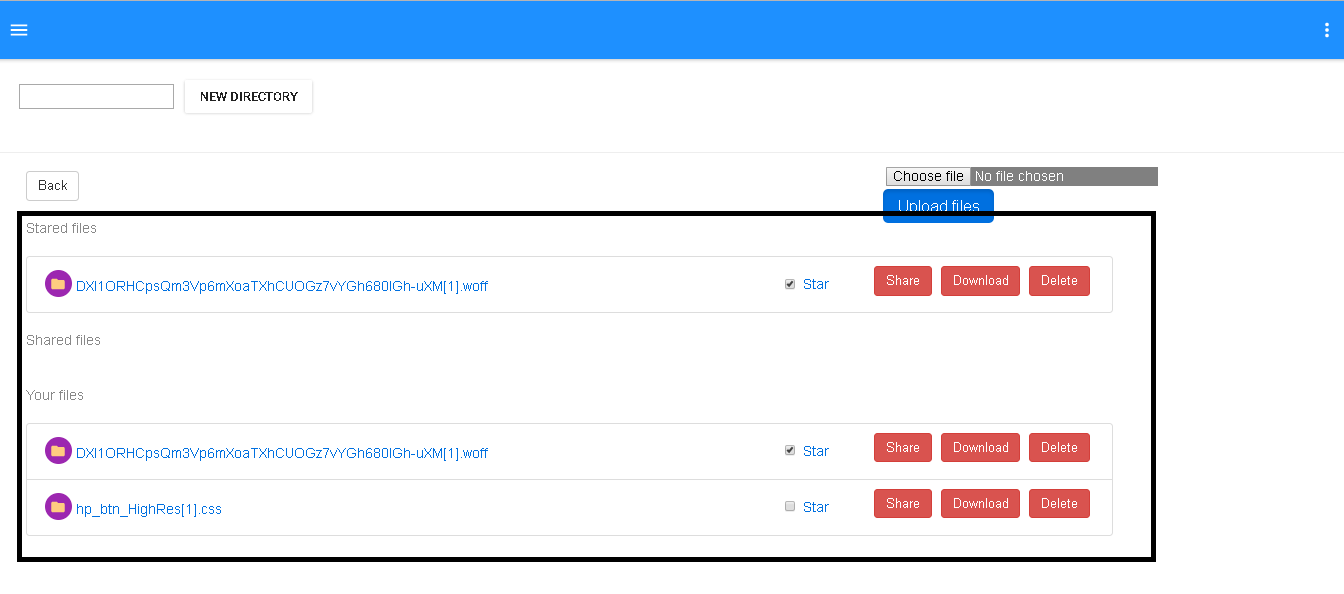


This is the page where most of the functionality of the application resides.

It contains File listing, file sharing, file download, file deletion.

It also includes directory creation and links to the user profile and user activity logs.

**File Listing:**

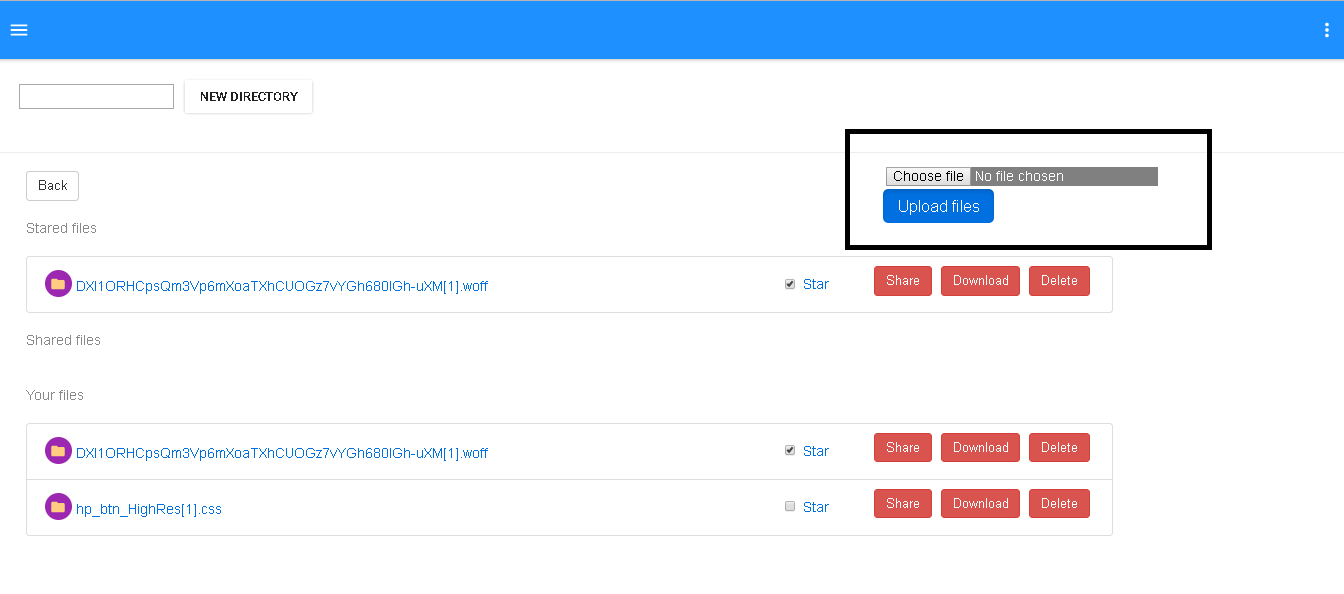
****

Each user has their own set of directories which they can access. On home page, the user by default have the list of already uploaded or shared files.

User can also Star the file and it will be shown in the starred tab of the screen.

Tables Used to get information: **Directories**

**Upload File**:

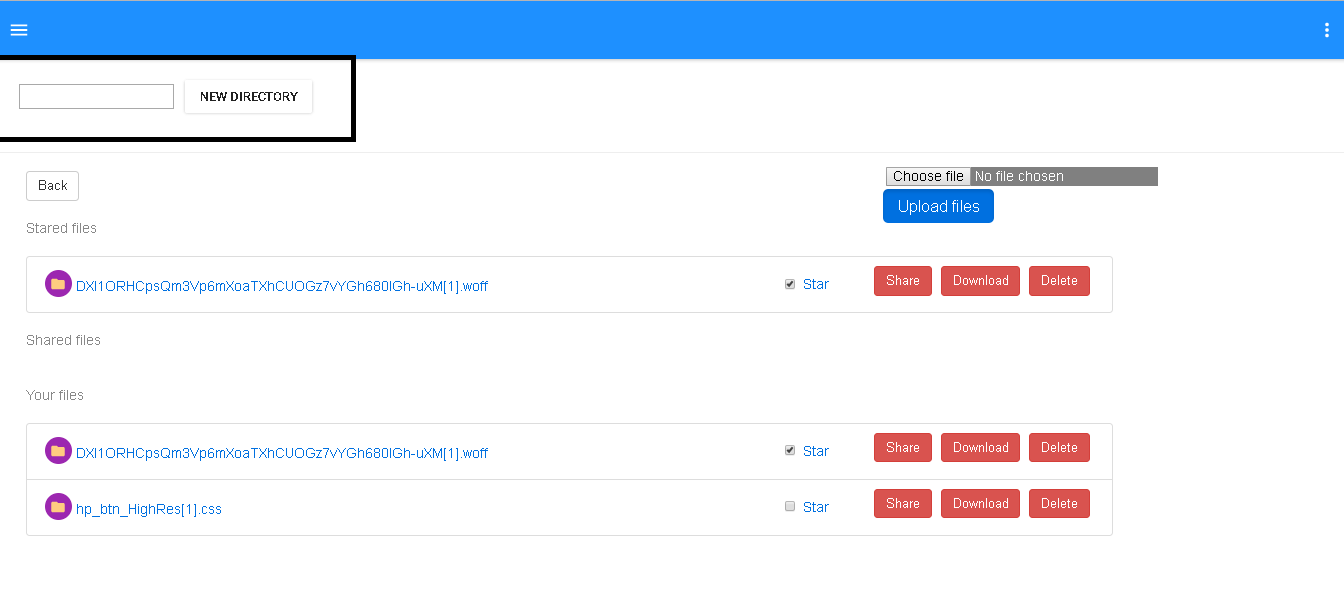


This functionality allows user to upload their own file into the system.

User can select a file from file chooser available in the screen and click on upload. After that, the system will check the current directory to upload the file. The file will be uploaded and the same will be noted down in the tables.

Tables Used to store information: **Directories, directory\_logging**

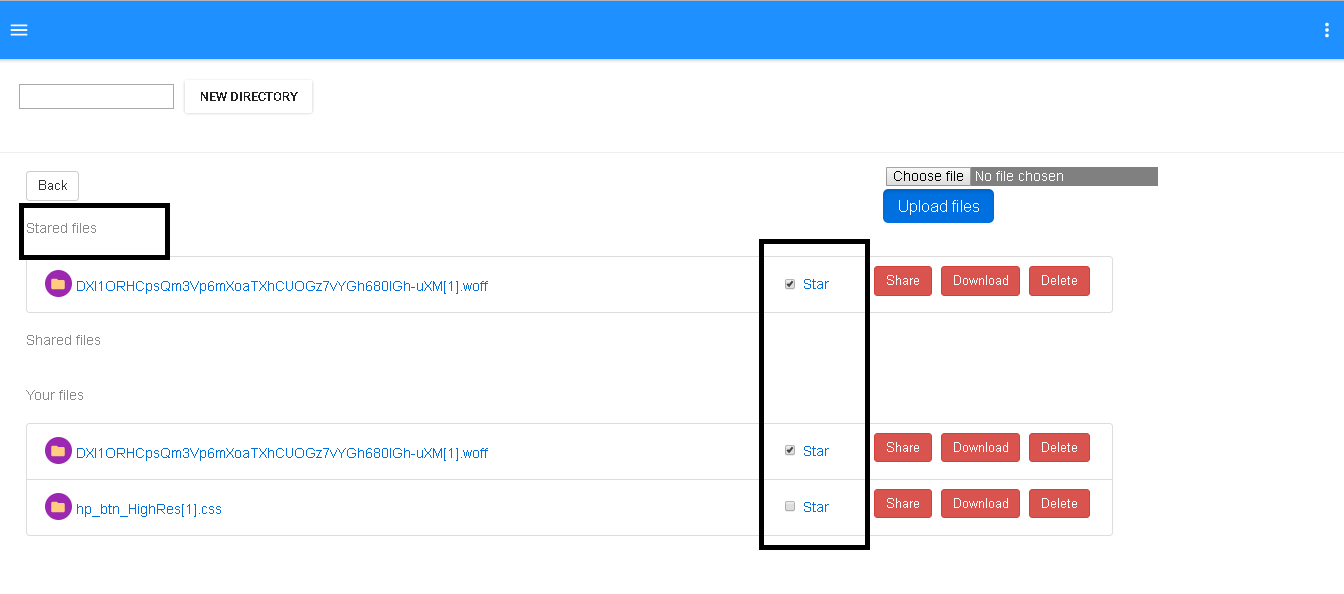
**Create Directory**:



This feature allows user to create new directory. The user can input name in the text box and clicks on the new directory button. In response, the system creates new directory and also logs that in particular tables.

Tables Used to store information: **Directories, directory\_logging**

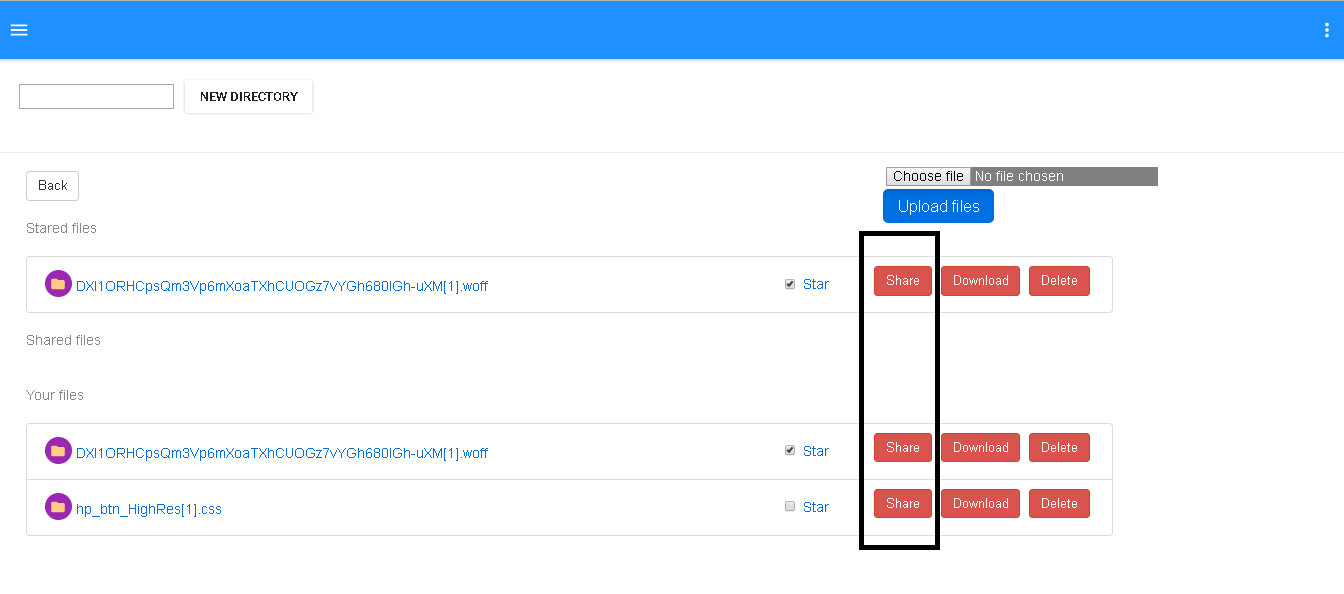
**Star Directory:**



User can start a directory or file to show them always up in the stack. The user will click on the star checkbox and the system will toggle the star status of the directory/file.

Tables Used to store information: **stardir**

**Share Directory/File:**

****

The user can select the share button and a text box will pop up which allows user to enter comma separated email address. By clicking the share button, the user submits the request.

The system then decides that given email address are contains all existing user or any user outside the system.

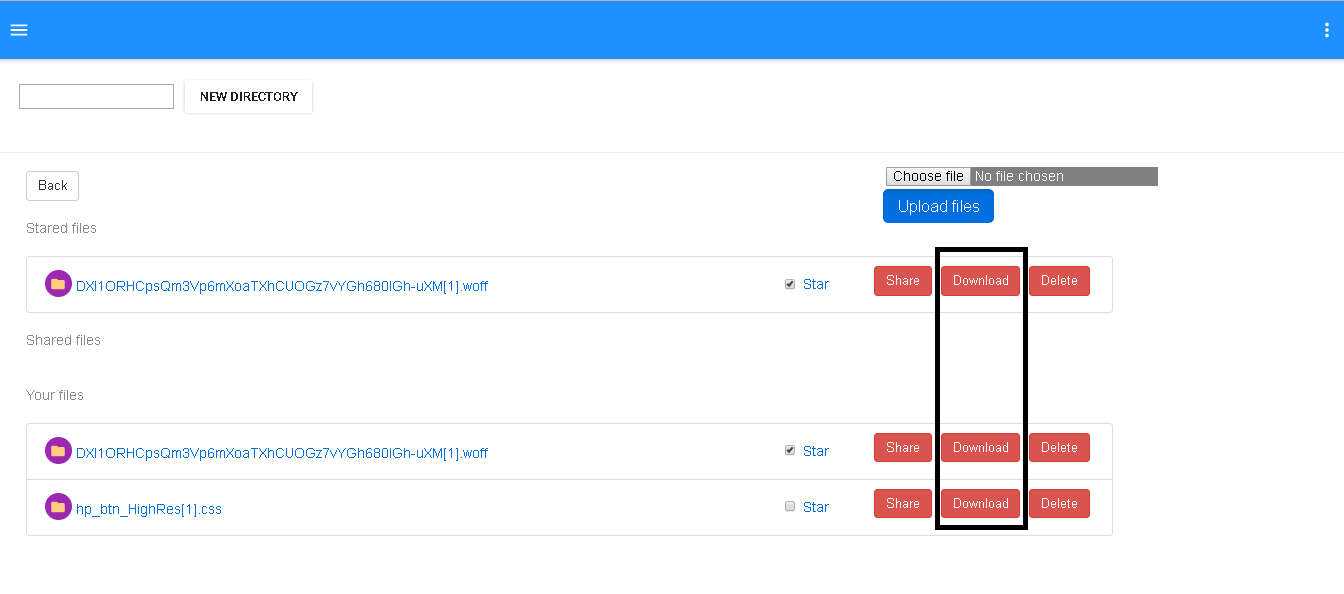
If any user is outside the system, system asks user that the sharing is going to happen by link and anyone having the link can accept it.

After having user’s consent, system shares file as link and send a message to notified users.

If all the emails are of existing customers only, te system shares file as user permission and shared users can see the file in their file list.

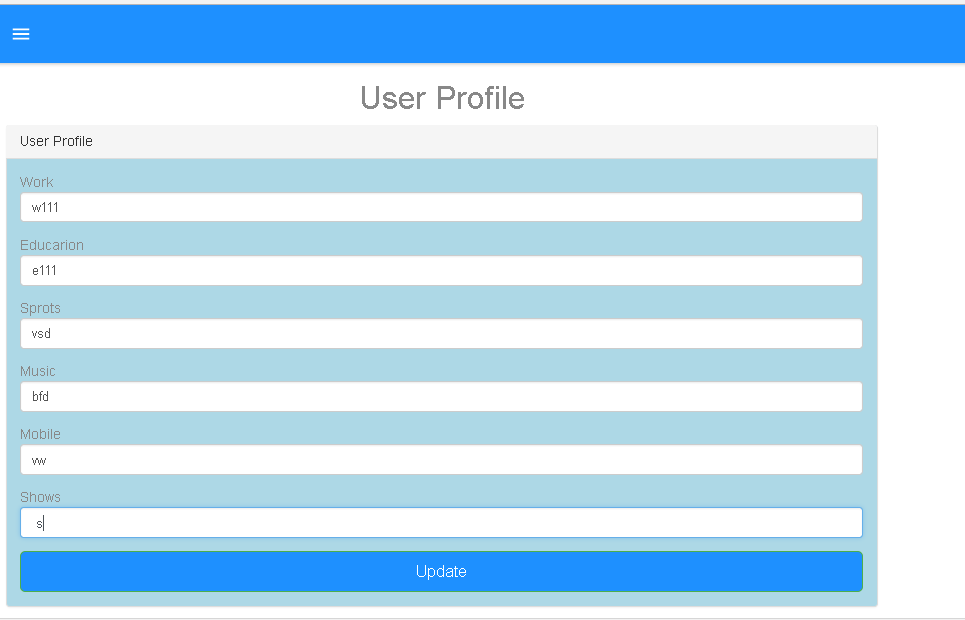
Tables Used to store information: **directory\_permissions**

**Download File :**

****

This functionality allows user to download particular file . User will just click on the download button and the system will download the file on the browser.

**User Profile:**



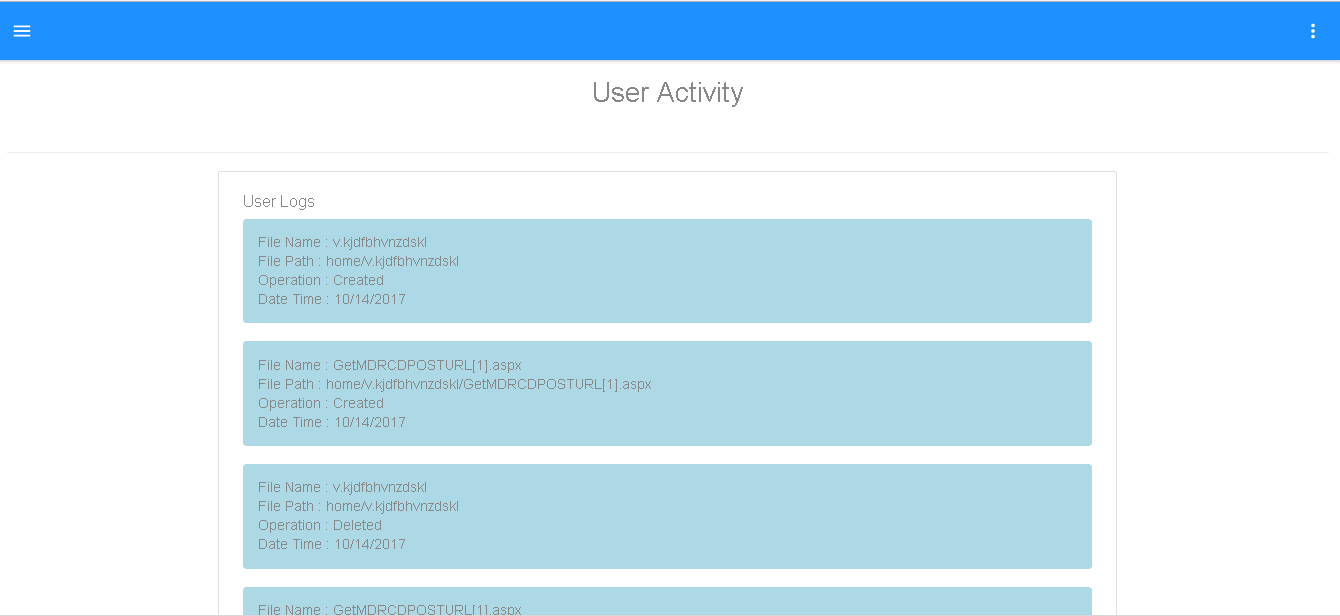
This feature allows users to add/update their user profile which contains user’s extra information.

Users will enter the details in the text boxes and clicks on the update button.

The system then stores data into the system and repopulate it on the screen.

Tables Used to store information: **userprofile**

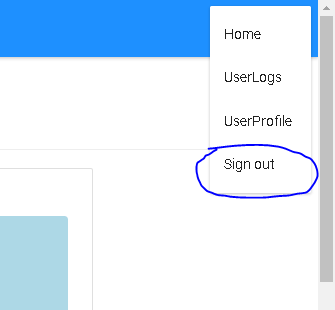
**User Activity Logs:**



This functionality is used to show user’s their activity log on the files. The user can see their file activities in a table.

Tables Used to store information: **directory\_logging**

**Sign Out:**



This functionality allows users to securely letting user to logout from the system.

**Performance**

After running the system, performance being measured is as follow.

1. The system is responding every request within 10 seconds.
2. The System is handling all the expected errors
3. System is changing routes for the functionalities accordingly
4. System allows users to insert data when necessary
5. System is changing necessary part of DOM instead of refreshing whole page
6. Every operation is well performed and executed properly on the server
7. Whenever needed, proper loggings are performed to the operations
8. Crucial data are safe and not shown on the UI
9. UI is always free for the user to perform any task as all the activities are asynchronized
10. Code is calling proper API for proper operations
11. System is not allowing any unauthorized calls to get data

Why This Performance?

React JS is used to **prevent unnecessary refreshing of pages**. Instead of that it only renders the necessary part of the DOM.

The **UI never freeze** and allows user to interact with other functionalities due to **unidirectional** flow of React JS.

**API calls** are written to minimize the server calls and make code more readable.

As node JS in available in backend, the **callback functionality** is used to make the calls **asynchronous** when needed.

**Small methods** are implemented to perform small tasks to **remove code redundancy** **and increase code reusability**.

**Connection Pooling** is used to decrease time of connection creation and closing. Which improves the response time of the system.

**Prepared Statements** are used to make the database inserts more reliable and secure by preventing possible SQL injection attacks.

**Q1)** **Explain the encryption algorithm used in your application. Mention different encryption algorithms available and the reason for your selection of the algorithm used.**

**Encryption Algorithm** :

The algorithm used in the system is bCrypt with salt and hash technique.  bCrypt is hashing based algorithm which was designed first in 1999 by Niels Provos and David MAzieres.It is based on the Blowfish cipher, and presented at USENIX in 1999.

In today's’ time where security is the most concern for IT industry, there are a lot encryption algorithms available to claim a proper security of data. Some of them are as below.

1. bCrypt
2. RSA
3. SHA-256
4. SHA-512
5. AES
6. Triple DES
7. Twofish

These all are mainly categories in Sympatric Encryption, Asymmetric Encryption and Hashing Algorithm.

**WHY bCrypt?**

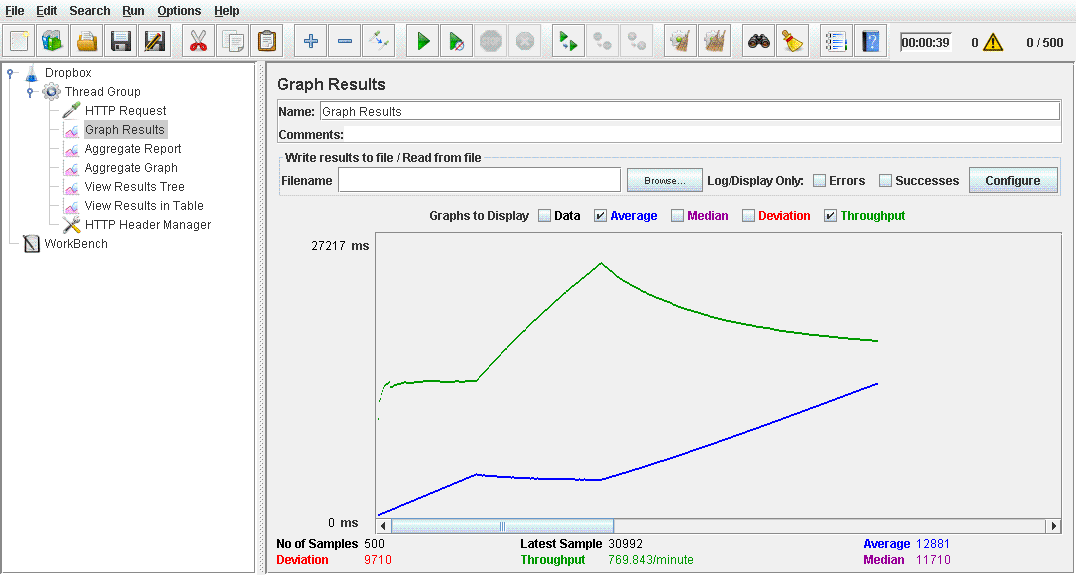
bCrypt is a hashing encryption algorithm which claims of being slow. Which allows less scope for the attackers to run the bulky script fast. This automatically decreases the risk of brute force attacks.

To add more, most of the industrial attackers prefer to avoid PC over GPU as they are allowing more use of cycles. As bCrypt relies on the hashing table throughout, this method works fast in individual PC then the GPU systems. So attackers are not getting boost which they used to have while cracking other encryption algorithms.

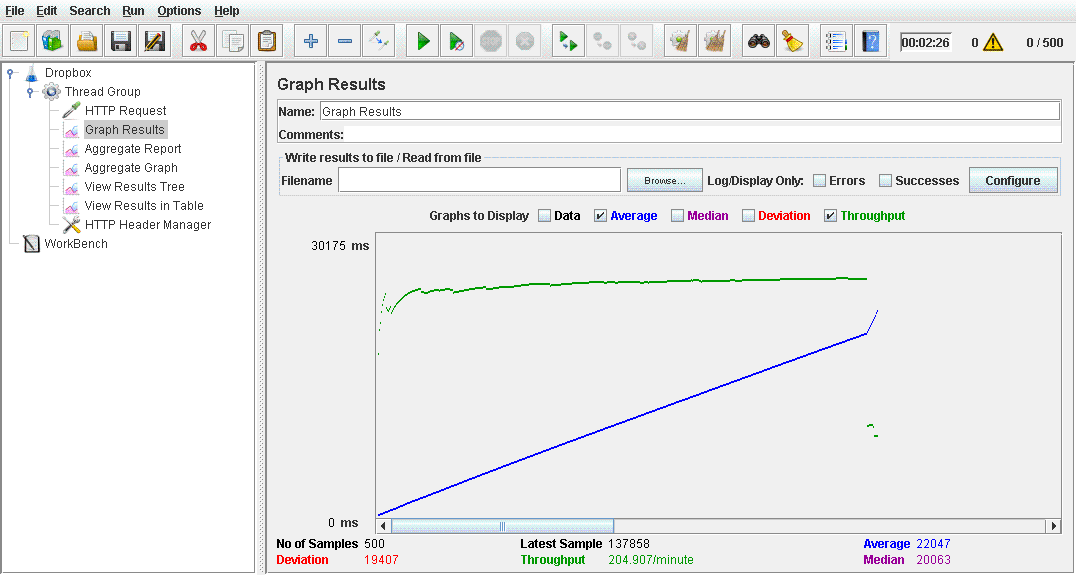
Also this 11 + year old algorithm is also recommended by a lot industry experts.

**Q2) Compare the results of graphs with and without connection pooling of database. Explain the result in detail and describe the connection pooling algorithm used in your code.**

**With Connection Pooling**



**Without Connection Pulling**



1. While comparing the graphs which of Jmeter tests i).Before connection pooling, ii). With connection pooling clearing shows that connection pooling is much better.
2. We can see the average time low when we are using connection pooling over not using it.
3. Also the throughput is high with connection pooling which allows multiple simultaneous calls from the users.
4. When number of concurrent user increase, the connection pulled db will not affect more as we don’t need to constantly open and close the connection. We are already having a thread of pool which is configurable according to the application use. If we are not using pooling, then we need to constantly create and close the connection which is very high weight process and needs much time.

In the code, the connection pooling algorithm used is the default mysql connection pooling algorithm. This algorithm actually creates the configurable number of connection and create a pool of it. When we need a connection, it returns us a connection available in the pool which is not being used by any other activity. When the work done, instead of closing the connection, it actually put it back in the pool and make it available to use.

**Q3) What is SQL caching? What all types of SQL caching is available and which suits your code the most. You don’t need to implement the caching, write pseudo code or explain in detail.**

SQL caching is the technique to store the sql data in physical memory to reduce the DB io calls.

SQL caching can be of many type. Some of them are as follow.

1. Adhoc query caching
2. Autoparameterization
3. Prepared queries
4. Stroed Procedures or other compiled objects

Adhoc Query caching are type where the database caches some adhoc queries and stores them as a cache. As this query are adhoc, we cannot actually rely on them as the tables can be altered after the caching.

Auto parameterization: This kind of caching we can refer to the prepared statements. When we are running same query again and again multiple times with same or different parameters, the database changes the parameter part as a generic part where different parameter can be put and executes.

As we can see this is useful in redundant query scenarios, we have to do the forceful auto parameterization to store all such queries and which will increase the number of unnecessary queries stored.

Prepared queries: we can create such kind of batches in the code and execute them in batch to reduce the server’s query plan generating process. Also, the DB has its own prepared queries storages.

Stored Procedures: This are known type of SQL caching where we can actually store the queries in the Database and use them when needed.

Also, now a days, server side caches are available such as memcache. We can store the queries which their desired output by assuming that the content of the table is not going to be changed.

They are storing data as key (query), value (table data) pair from where we can easily access it.

Ref: <https://msdn.microsoft.com/en-us/library/cc293623.aspx>

**Q4) Is your session strategy horizontally scalable? If YES, explain your session handling strategy. If NO, then explain how can you achieve it.**

Yes, as the system manages session on server side, rather than client side it is horizontally scalable.

Also, it is using rest less API and passes the unique identity of client via session.

Here we are achieving a stateless protocol and managing the sessions on server side. The Sessions are not actually stored in the server but they are recognized by the server as it was created on server itself.

In my application, server has the sole responsibility to manage the sessions.  Sessions are generated, altered and passed in any response which server is sending to the client.

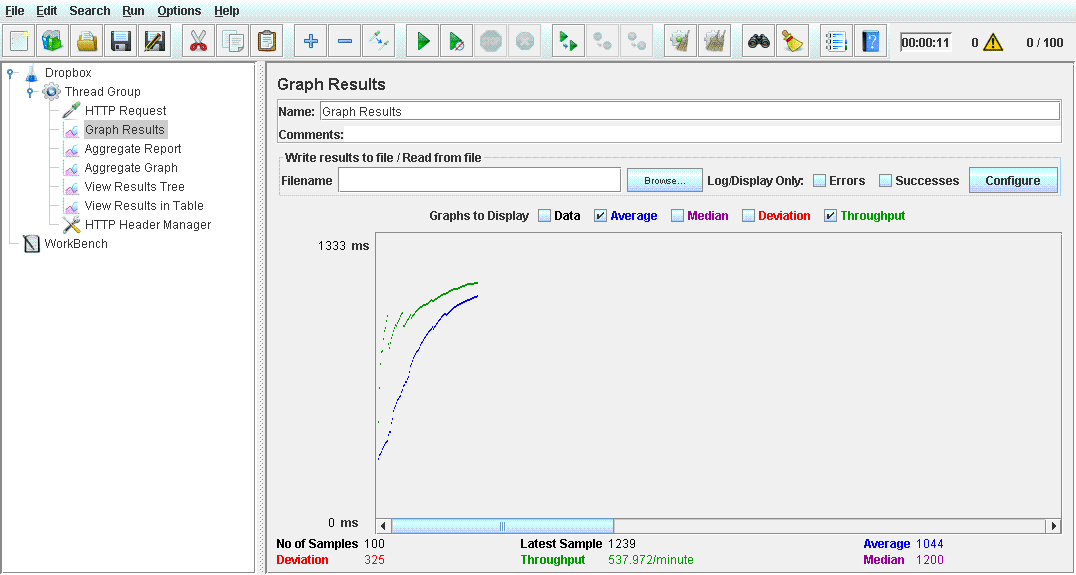
The client does not know anything about the session, it just gets it in response and pass it again in every response and the rest is all up to the server.

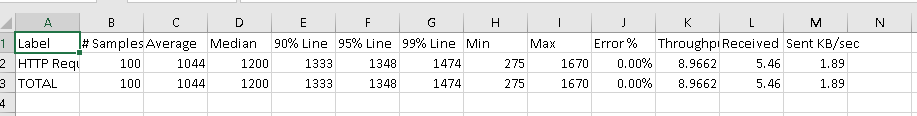
In the system, sessions are also used to authenticate the logged in user request and fulfill the stateless protocol. The system has one interceptor which is tracing all the request from the client and checking about the authenticity, if it founds that the request has no session or invalid session, it stops the request and responses with the 501-unauthorized access status.

**Jmeter Testing for DropBox (Restfull API)**

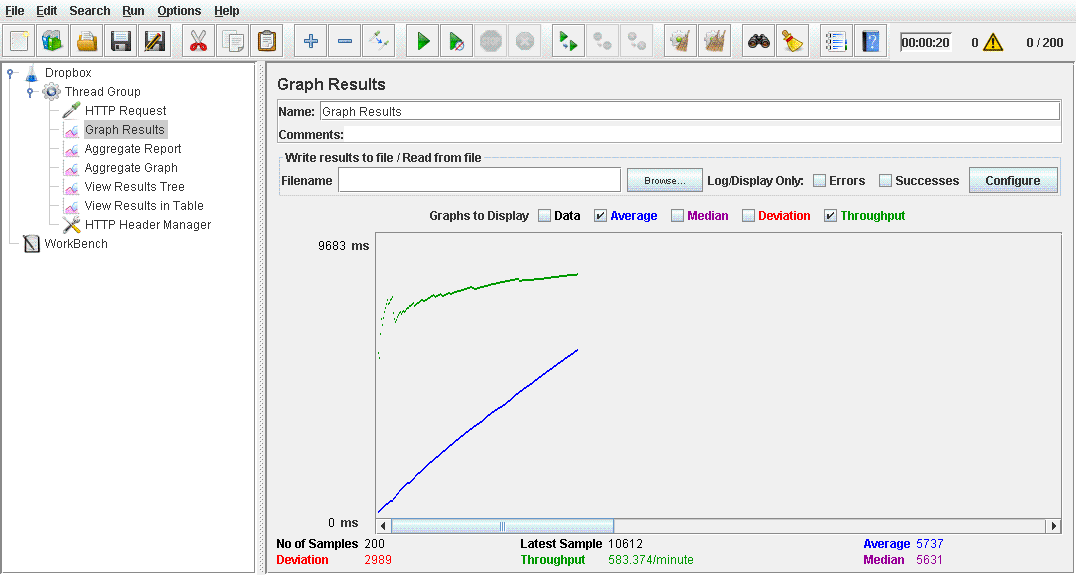
1. **Without Connection Pooling**

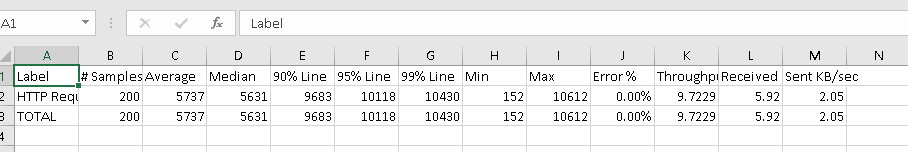
**100 users**



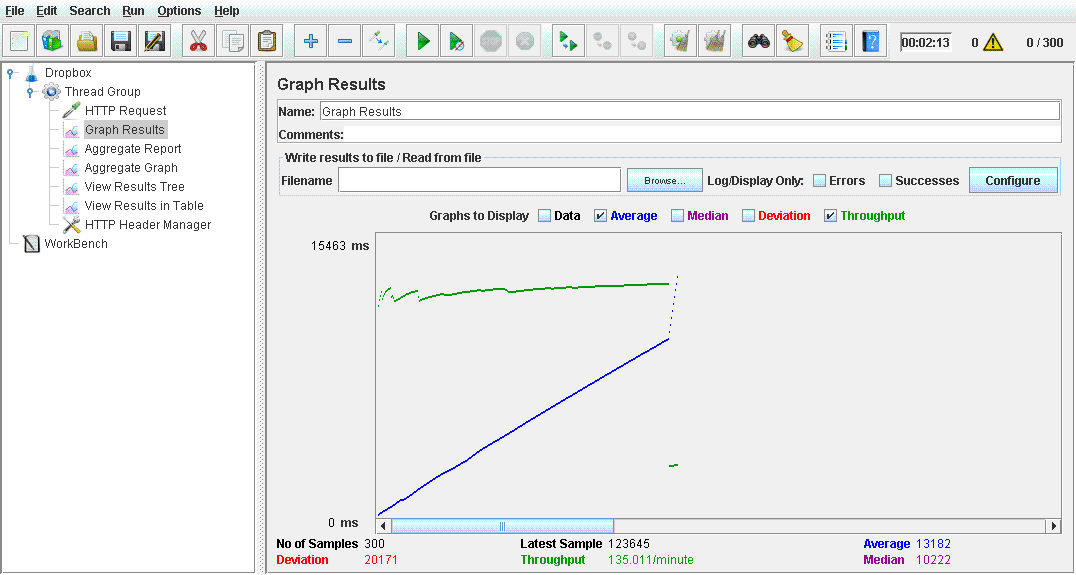


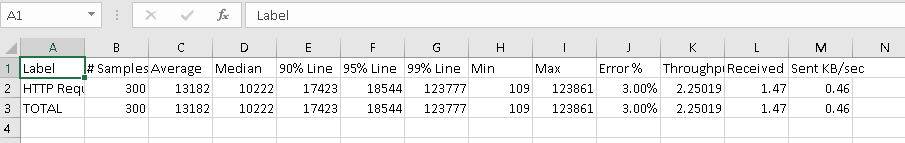
**200 users**



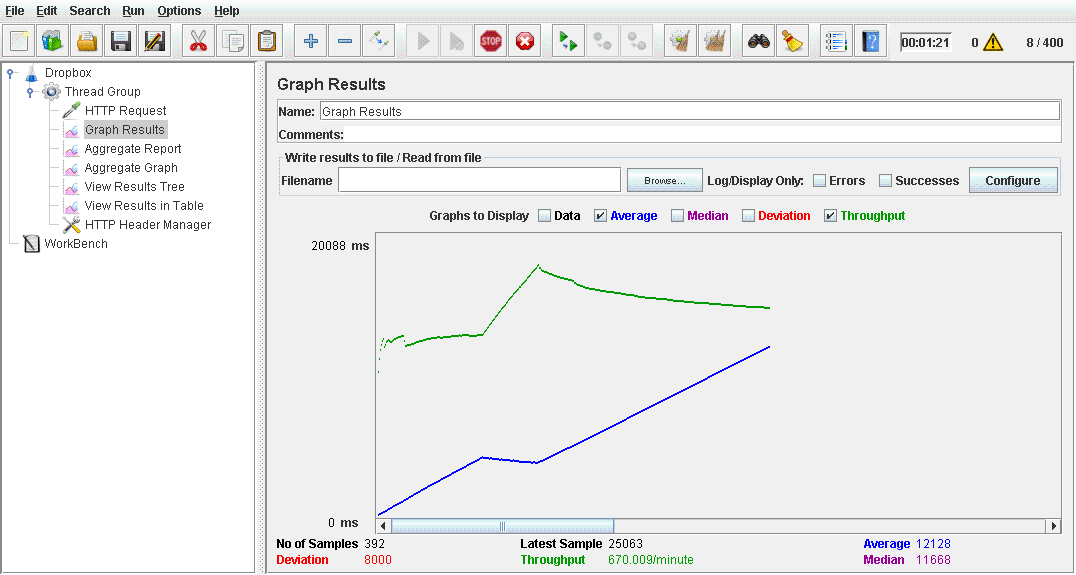


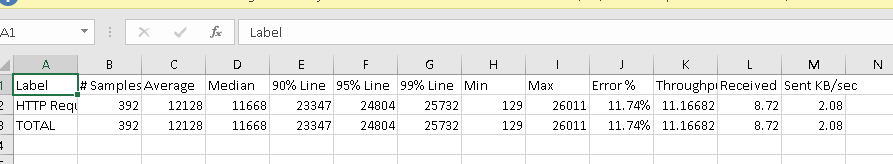
**300 users**



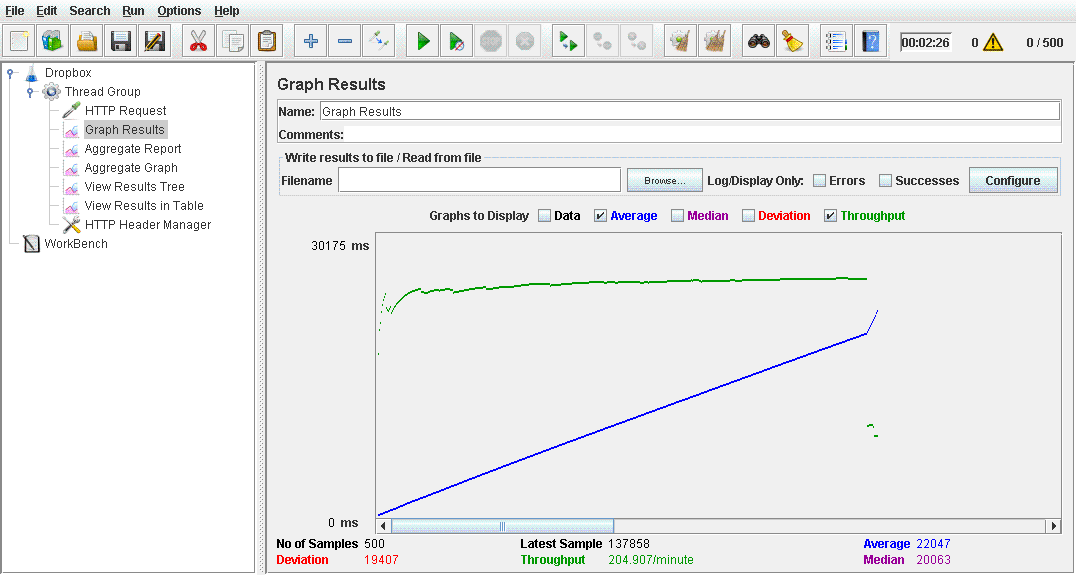


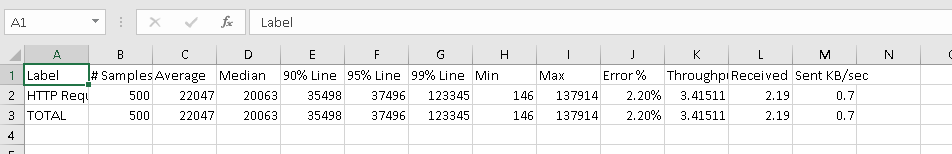
**400 users**





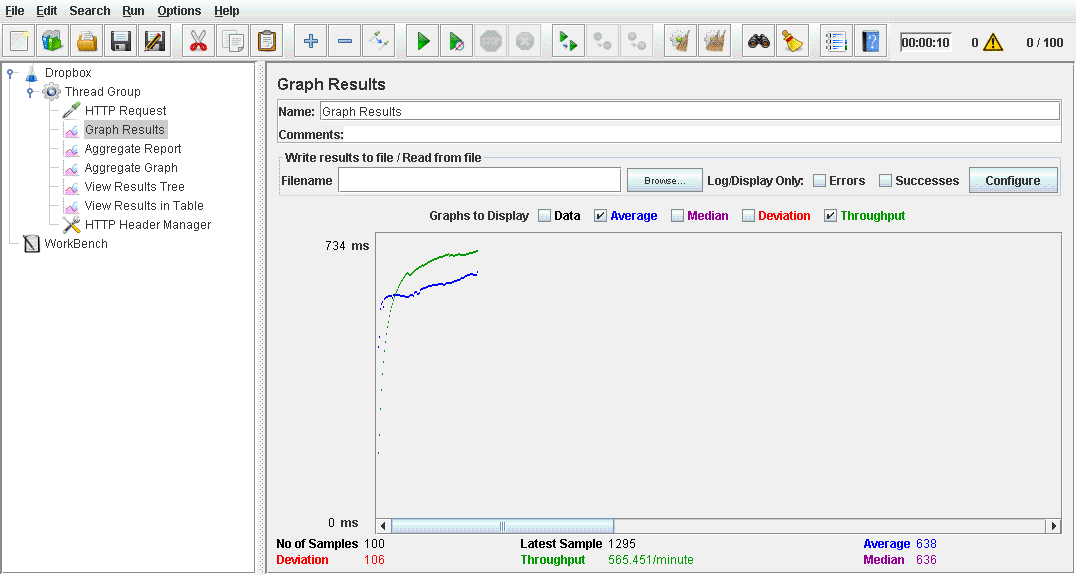
**500 users**

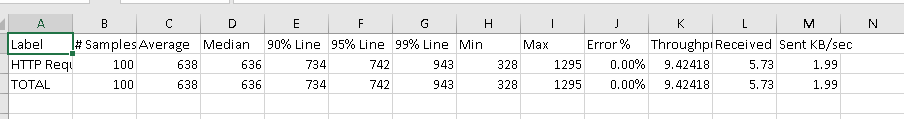




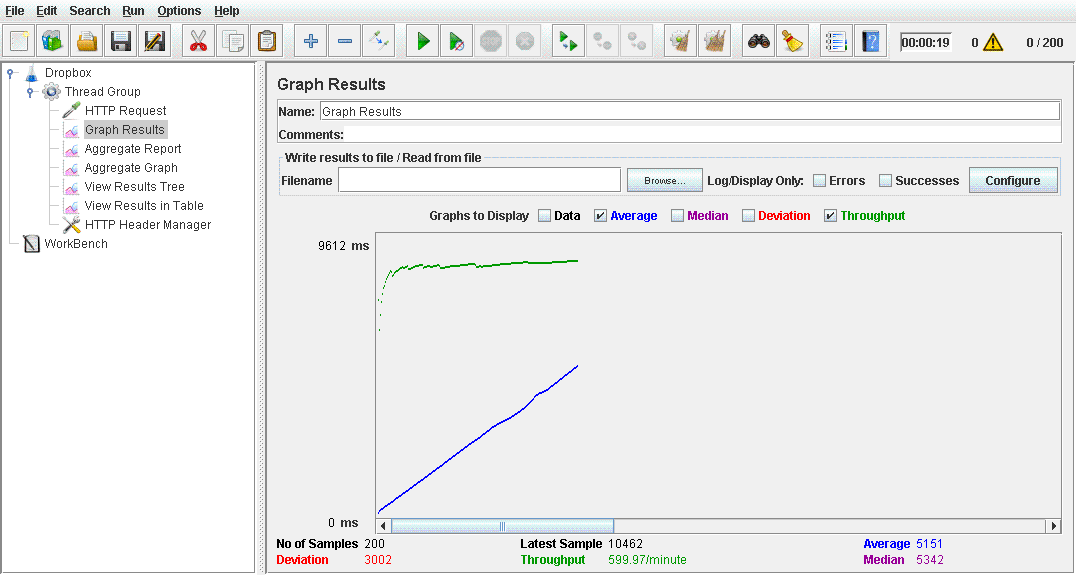
1. **With Connection Pooling**

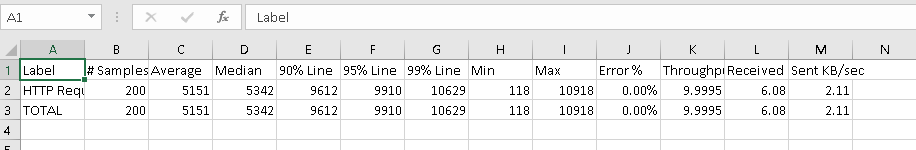
**100 users**



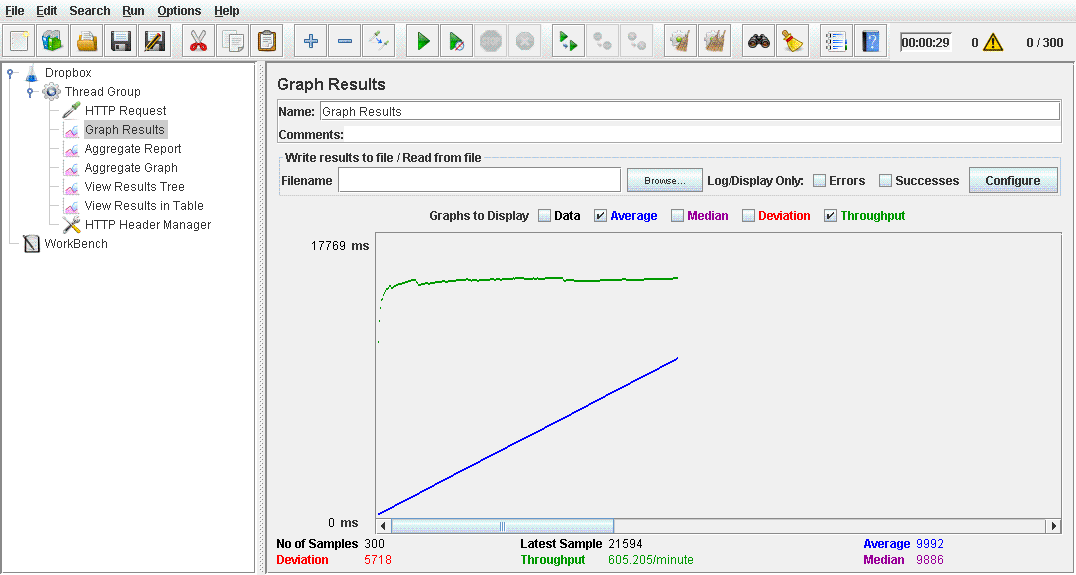


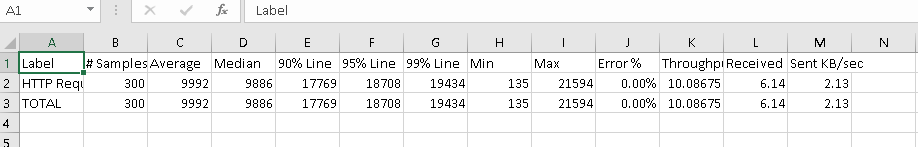
**200users**



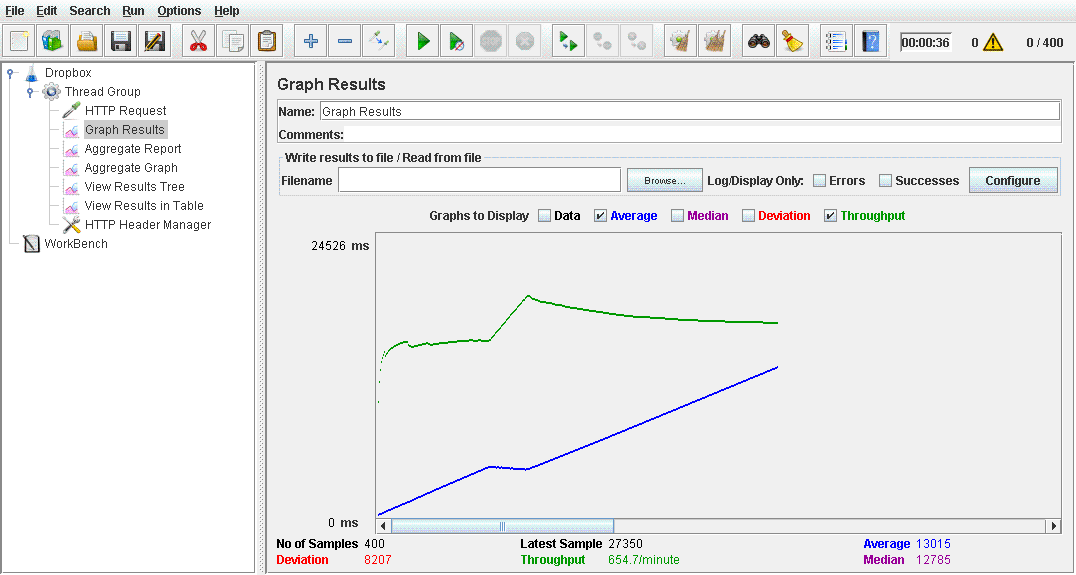


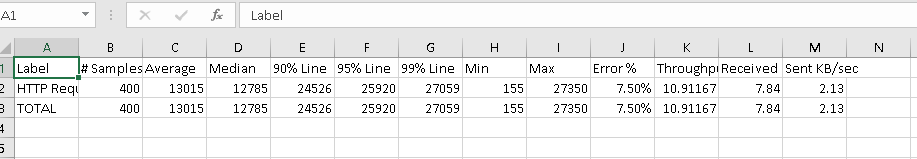
**300 users**



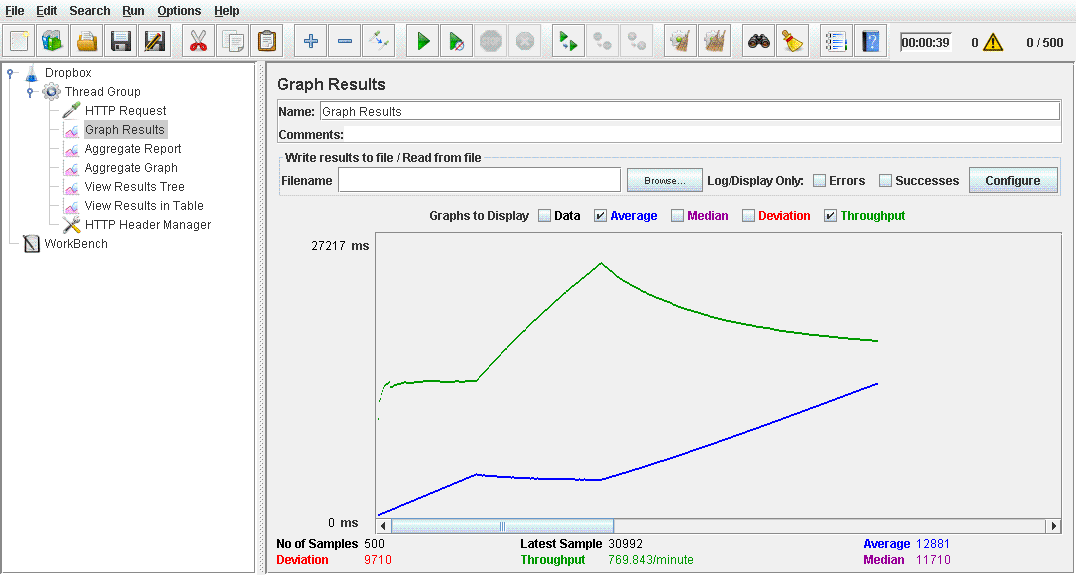


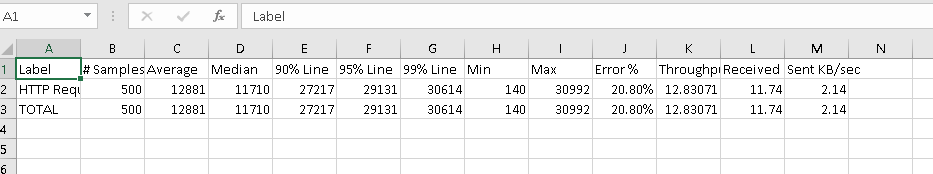
**400 users**





**500 users**

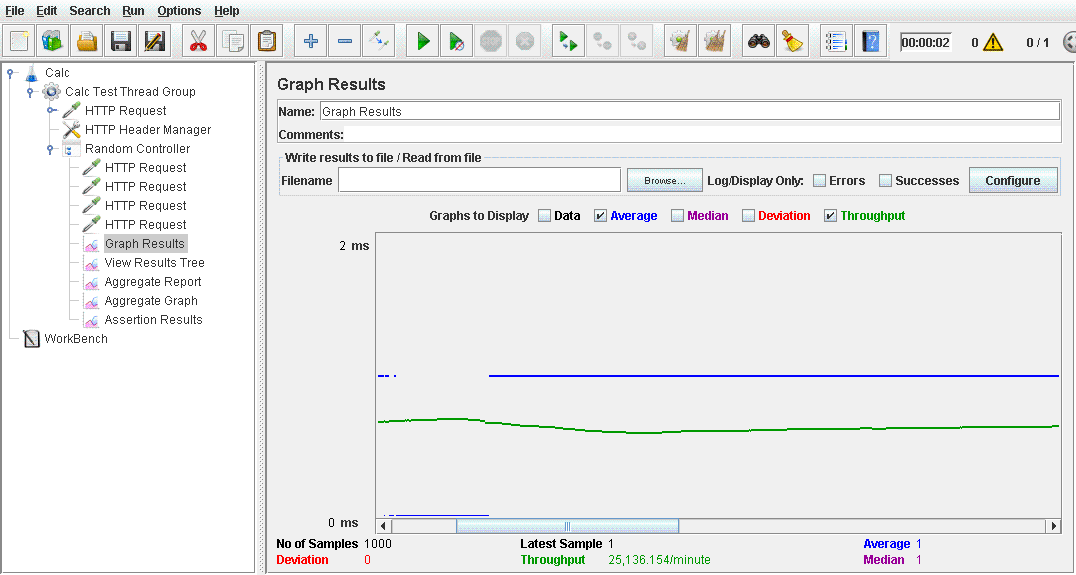


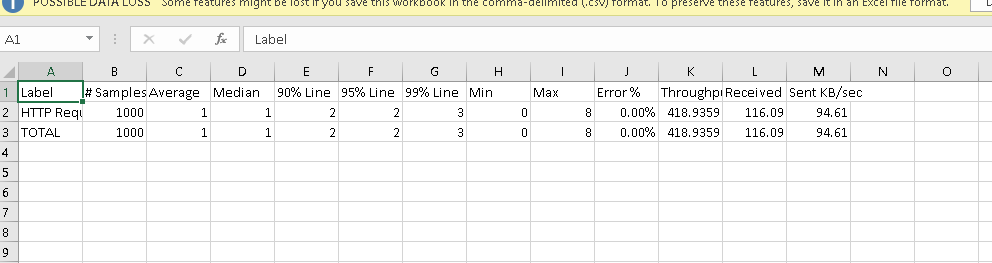


**Calculator**

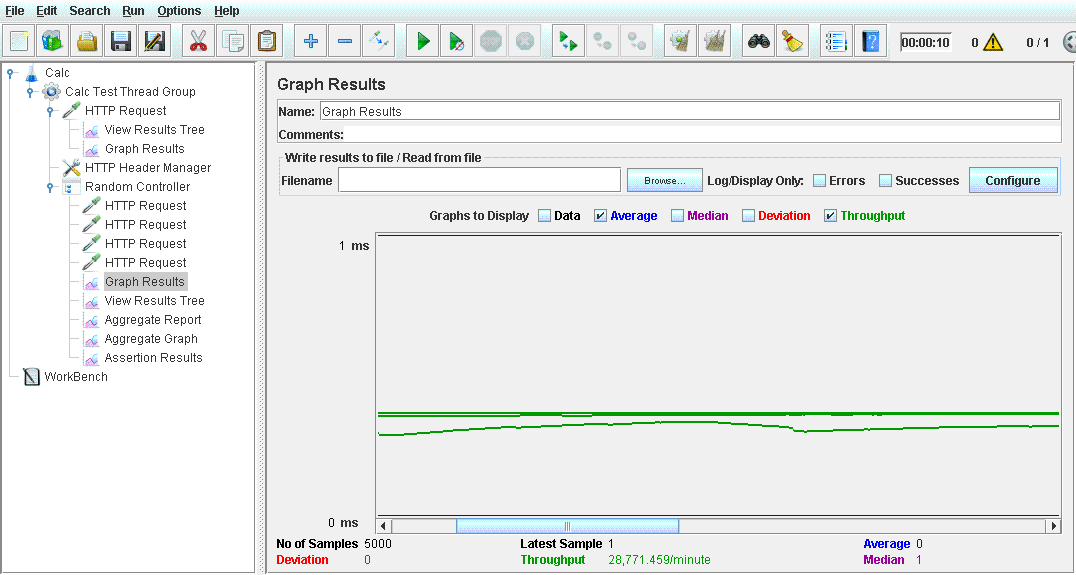
**Jmeter Tests:**

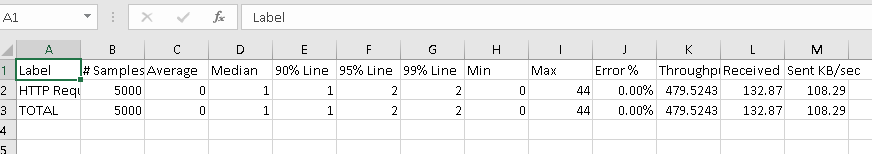
1. **1,000 calculator calls on randomly selected tasks**

****



**5,000 calculator calls on randomly selected tasks**





**100 concurrent users with 1000 calls each to calculator on randomly selected tasks**

