**LAB1**

**Introduction:**

**Goal**: To learn RESTful service using nodejs,angularjs and express framework imlementation of Calculator and Facebook is done in this lab.Calculator system performs arithmatic opeations addition,subtraction,multiplcation,division.Facebook application uses MySql to store the database and nodejs angularjs are used for front end implementation.

**Purpose of system:**

**Calculator:**

* To perform basic operations like Addition,Subtraction,Multiplication,Division.
* To perform basic checks on numbers inputs like it shoud not be null or string
* To perform basic checks on results like any number cannot be divided by 0.

**List of API's for Calculator:**

**1**.Calculate:Helps to perform all the arithmatic operations

**Facebook:**

* Creating a RESTful service.
* SignIn,SignUp,SignOut functionality
* Implementing Sessions for a user
* Displaying About,Interests functions for a user
* User can see friendlist,send friend request,accept friend request
* User can create and delete Group,add and delete,view members to group
* Provide news feed functionality
* Performing connection pooling to increase the performance

**List of API's for Calculator:**

* Login:Helps to login into the system and maintains a session
* SignUp:User is signed up with the help of this function
* logout:User can logout and session will be deleted for the user
* UserProfile:Displays The main frontpage of facebook with news feed
* FriendList:Displays List of Friends
* Interests:Displays Interests
* ProfilePageNavigate:Displays the user profile page
* GroupPageNavigate:Displays the list of groups
* Groups:Usre can add groups from this API

**System Design:**

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

Data design: It represents how data is stored within the system.

User interface design: It represents how user’s information flows to the system and back using the view provided by the User interface.

Process design: It defines how data moves through the system.

I have used Model-View-Ccontroller (MVC) architecture so that database logic,the business logic, presentation represents different decoupled layers.



**Calculator:**

When user hits the url and find 2 textbox to enter the number and one dropdown list for selecting the operation to be performed between addition,subtraction,multiplication,division

* User have to give 2 inputs
* User should select one of 4 operations(Addition is set bydefault)
* As soon as the user hits the calculate button the result will be visible.

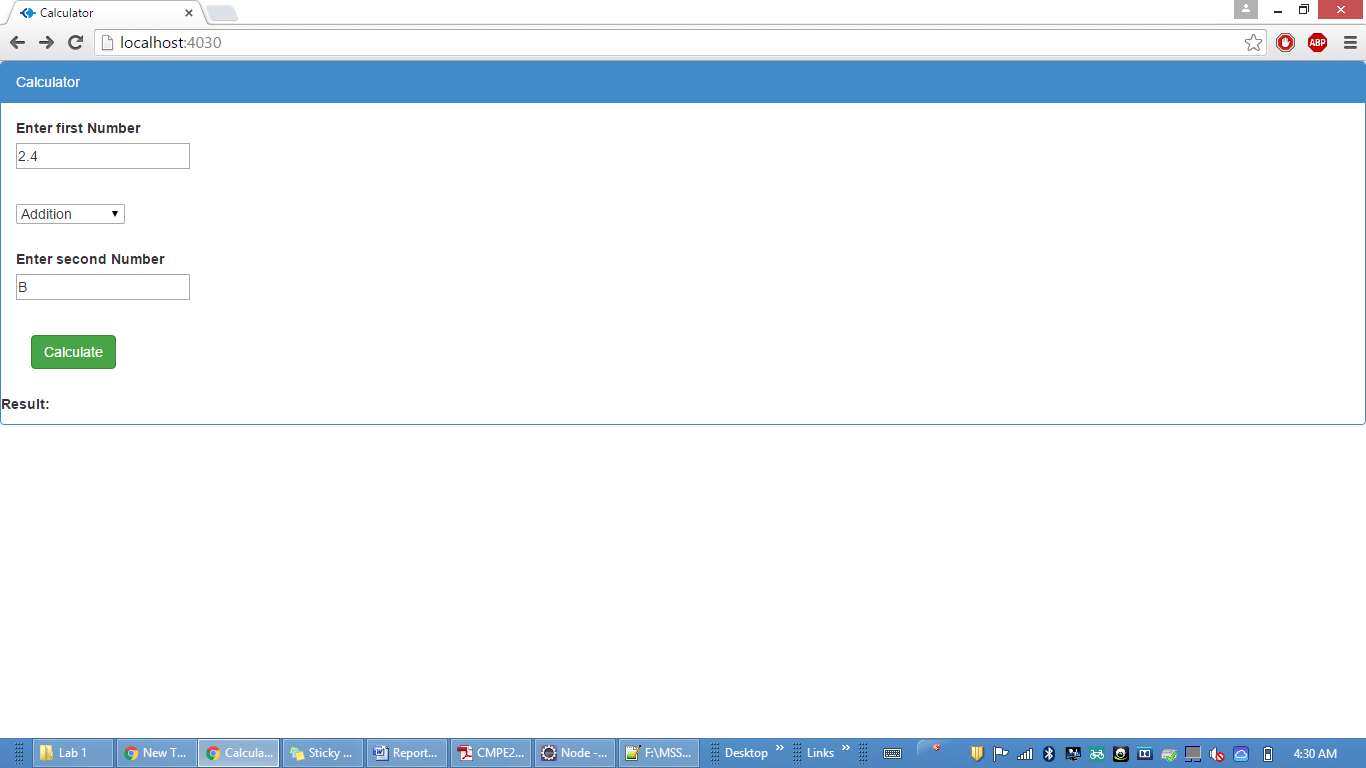
**Facebook Application:**

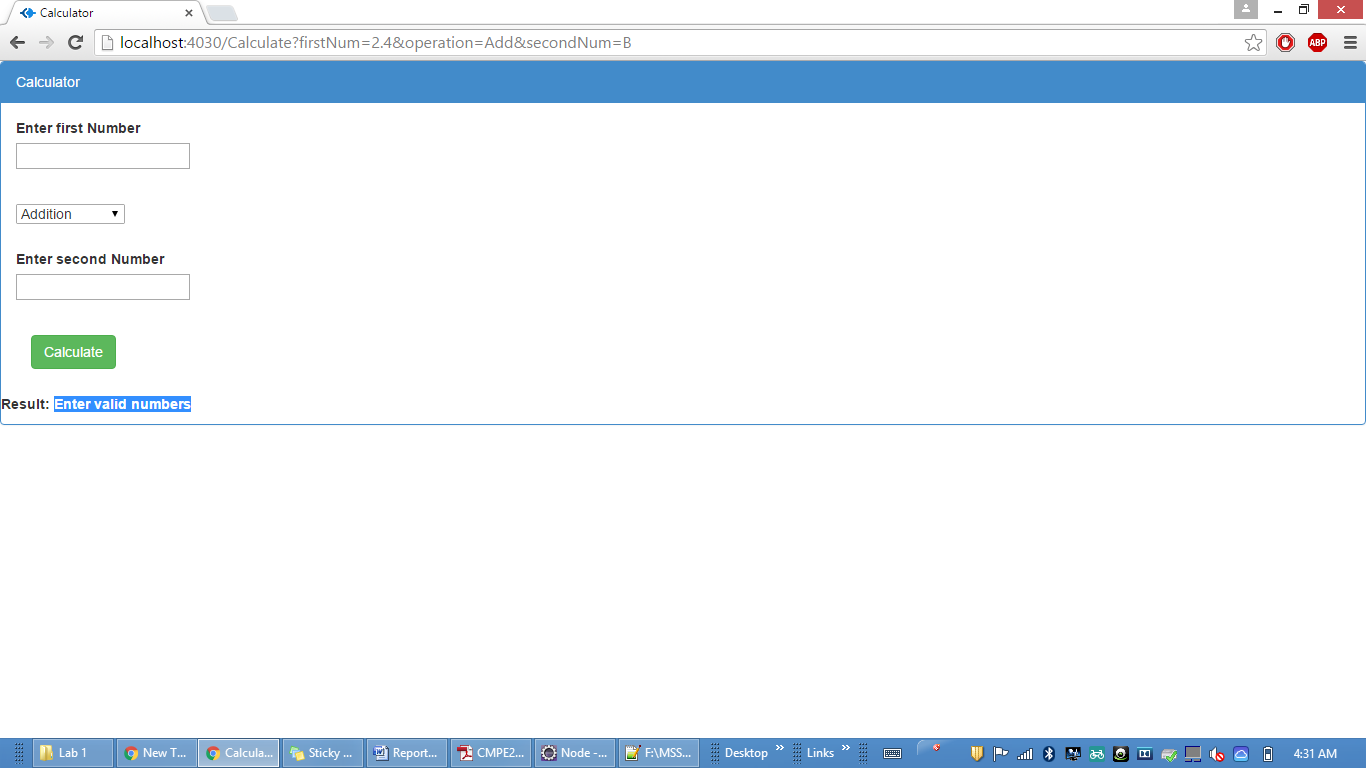
* When user hits the url signIn/signUp page will be displayed
* When user is not registered with the system he has to signUp first
* After signUp user can login using their own credentials
* When the user is successfully signed in he/she will see the home page which displays news feed and buttons to navigate to About page/Friends page/Groups page etc
* When user selects Profile page(About) he/she will be able to view all the details of the user like about user,interests of a user etc.
* Friends page displays the list of friends and friend requests if any to accept
* Groups page will display the list of groups of user
* User can create or delete a group and also add/delete/view group members.
* After Logout button click user session will be terminated and user will be logged out.

**Results:**

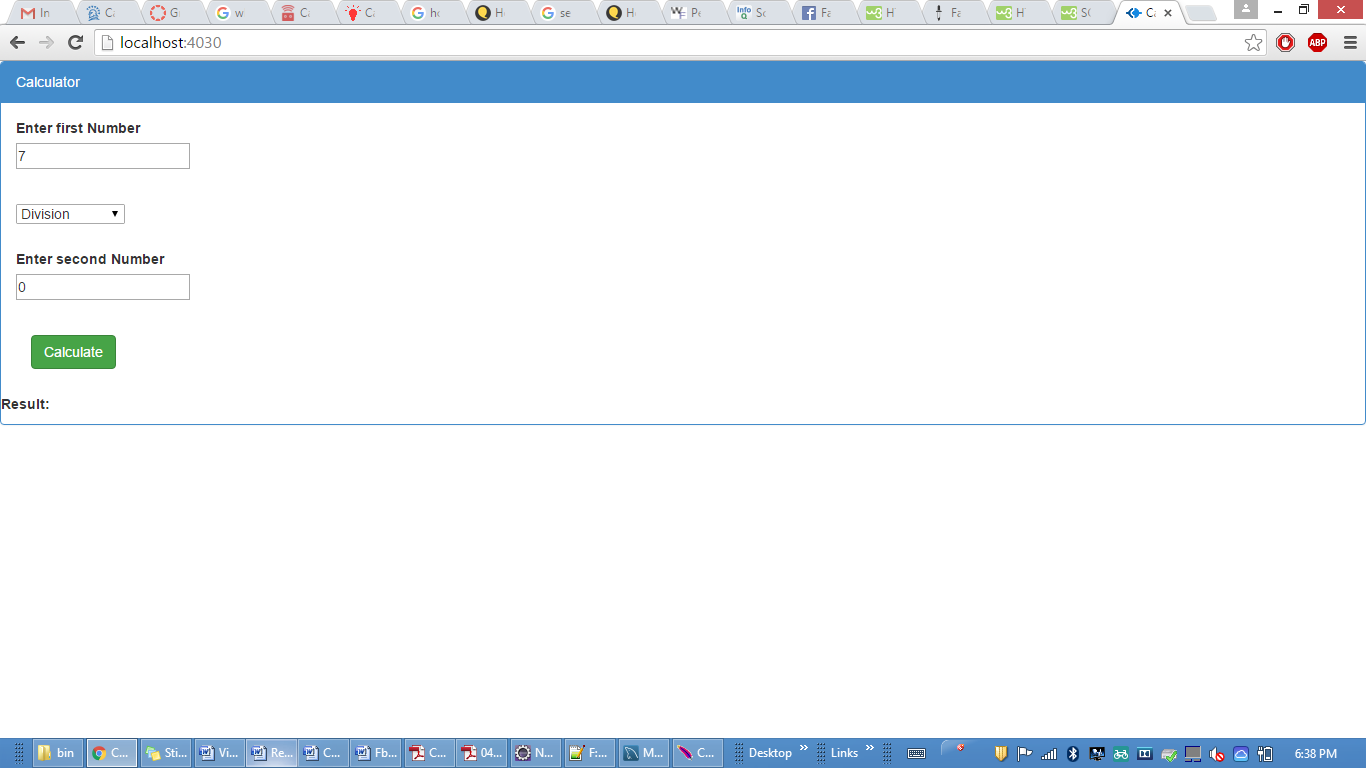
Result for wrong input(input 2:B)

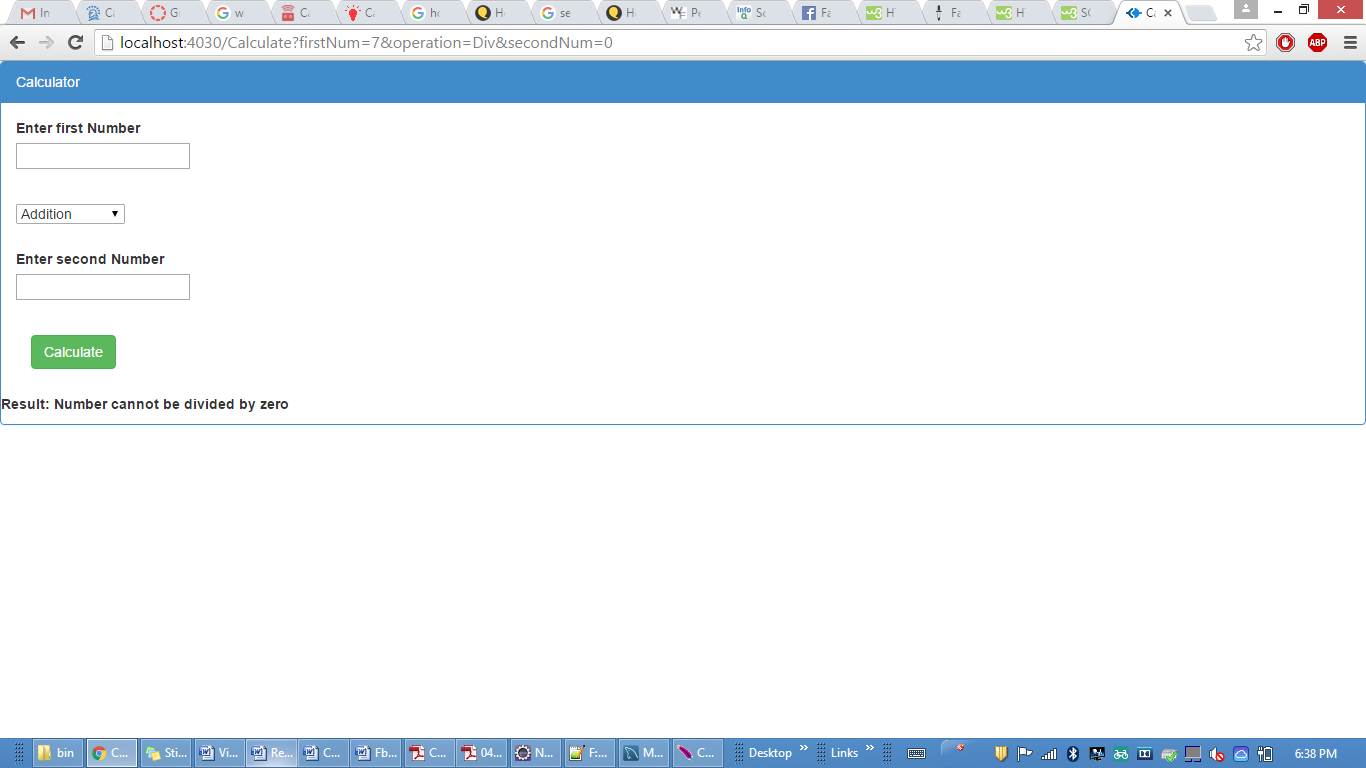
**Calulator:**





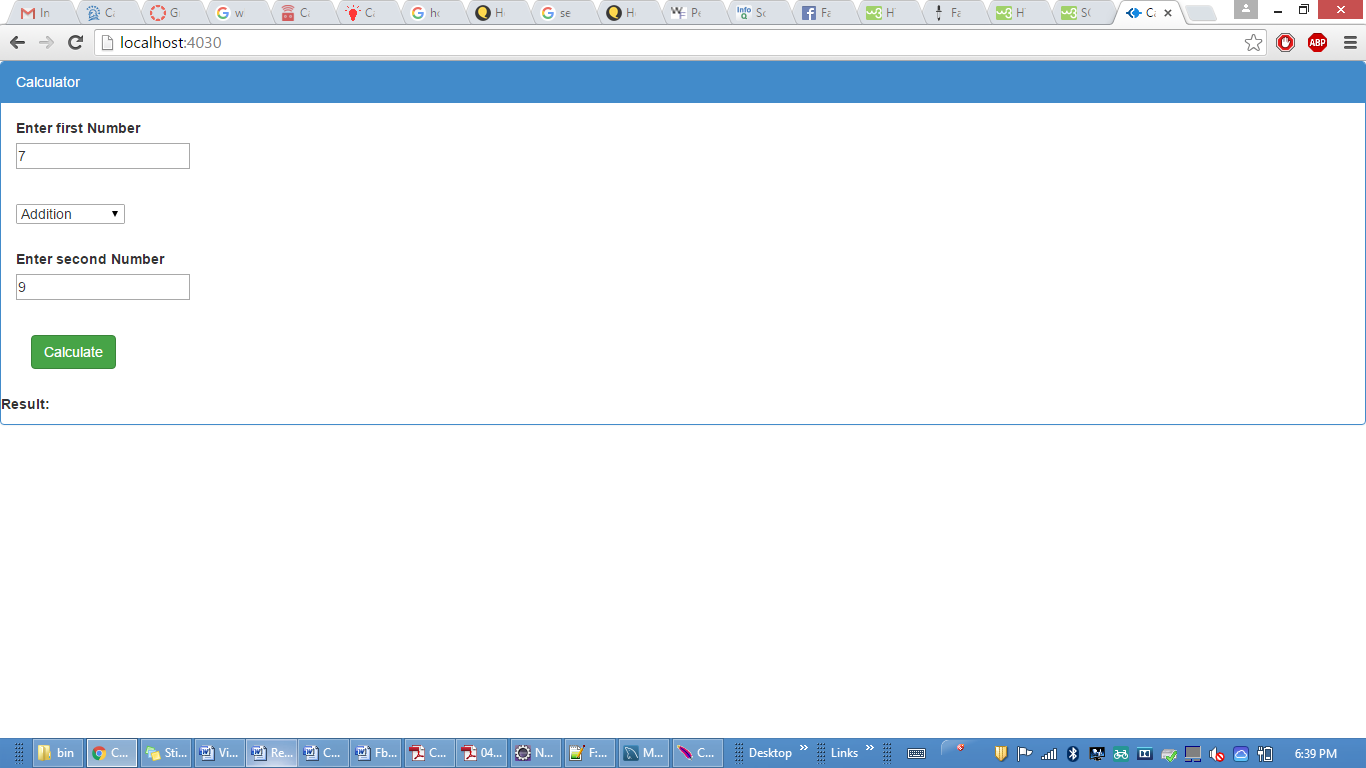
Division when second number is 0

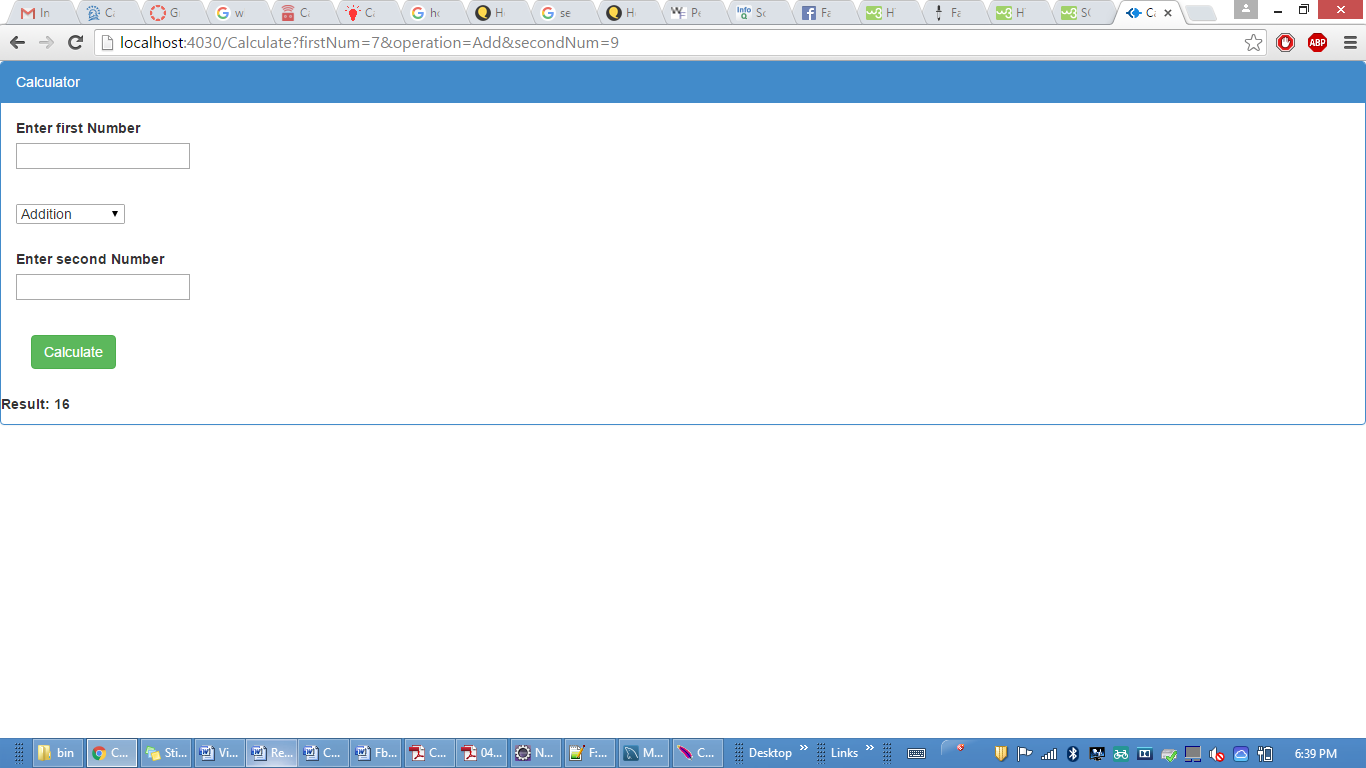




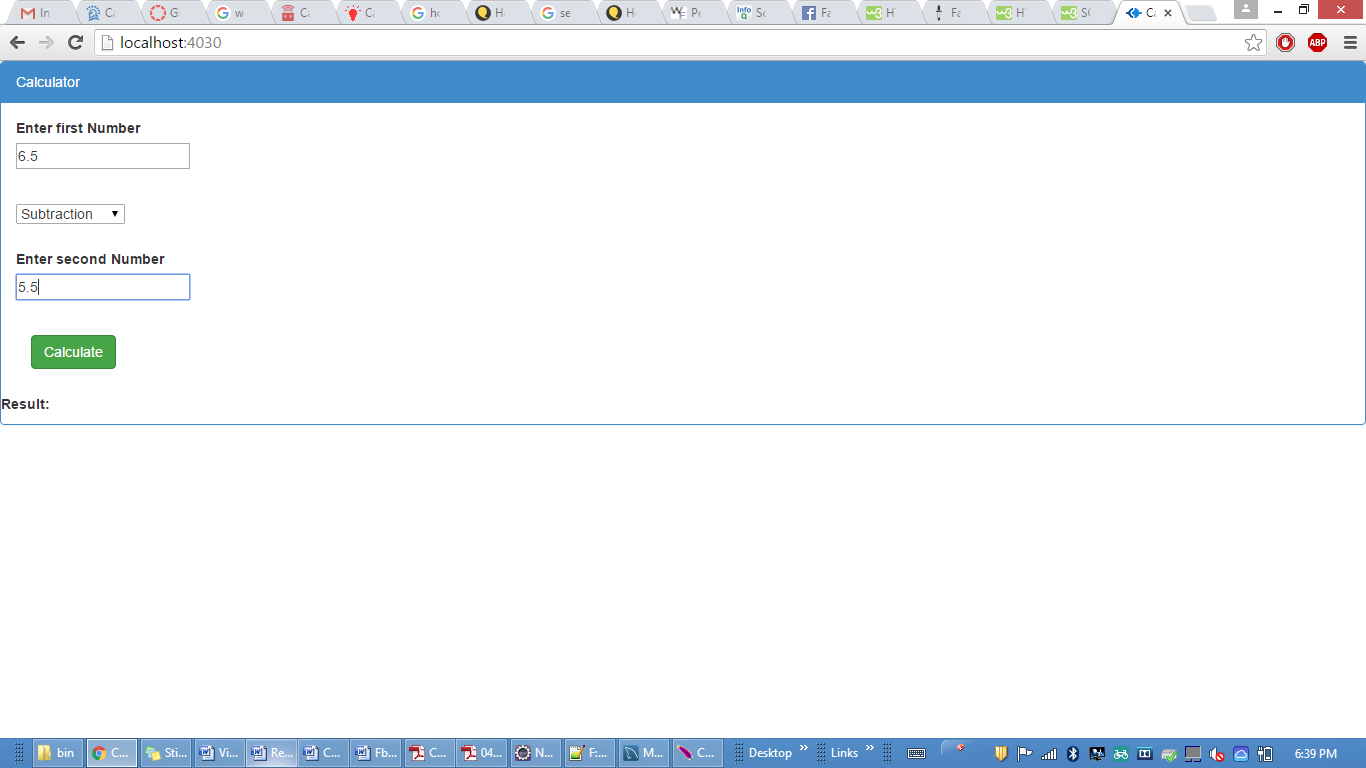
Correct Results

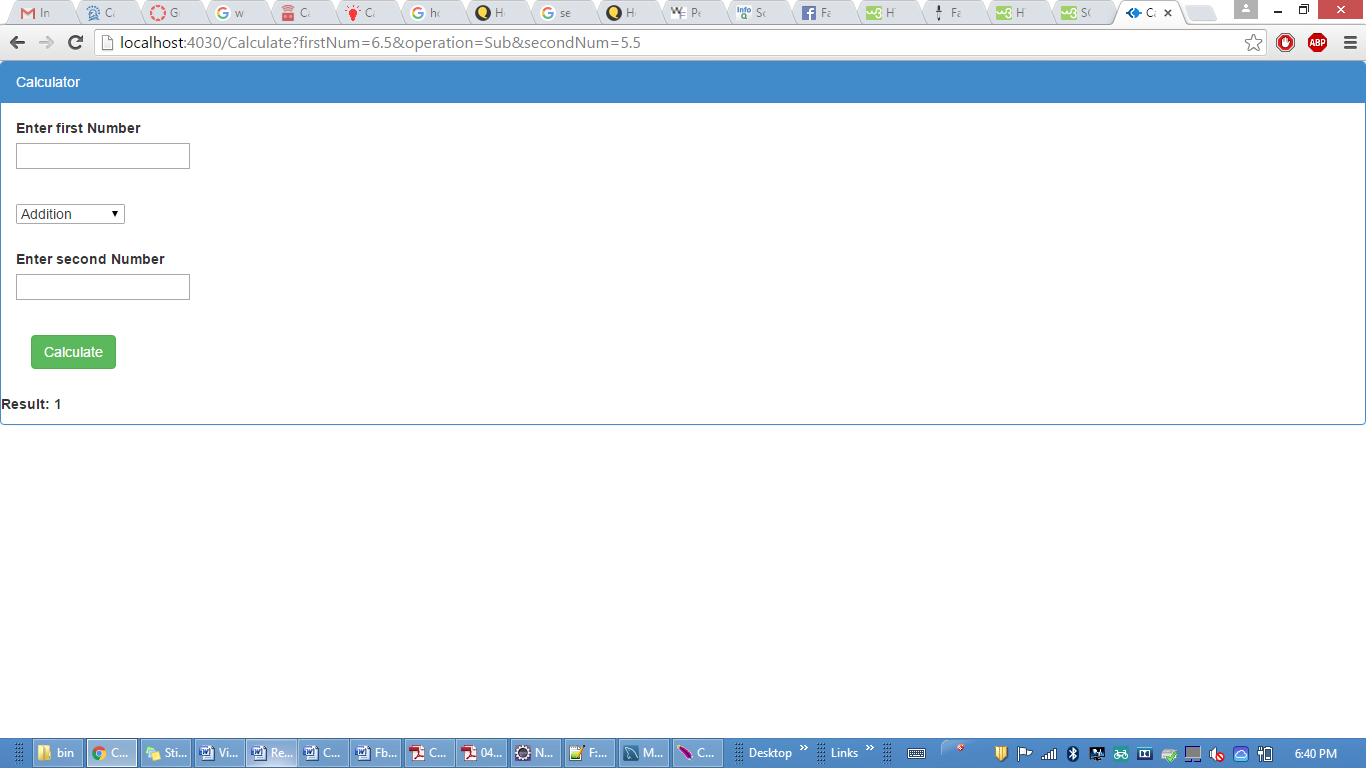
Addition



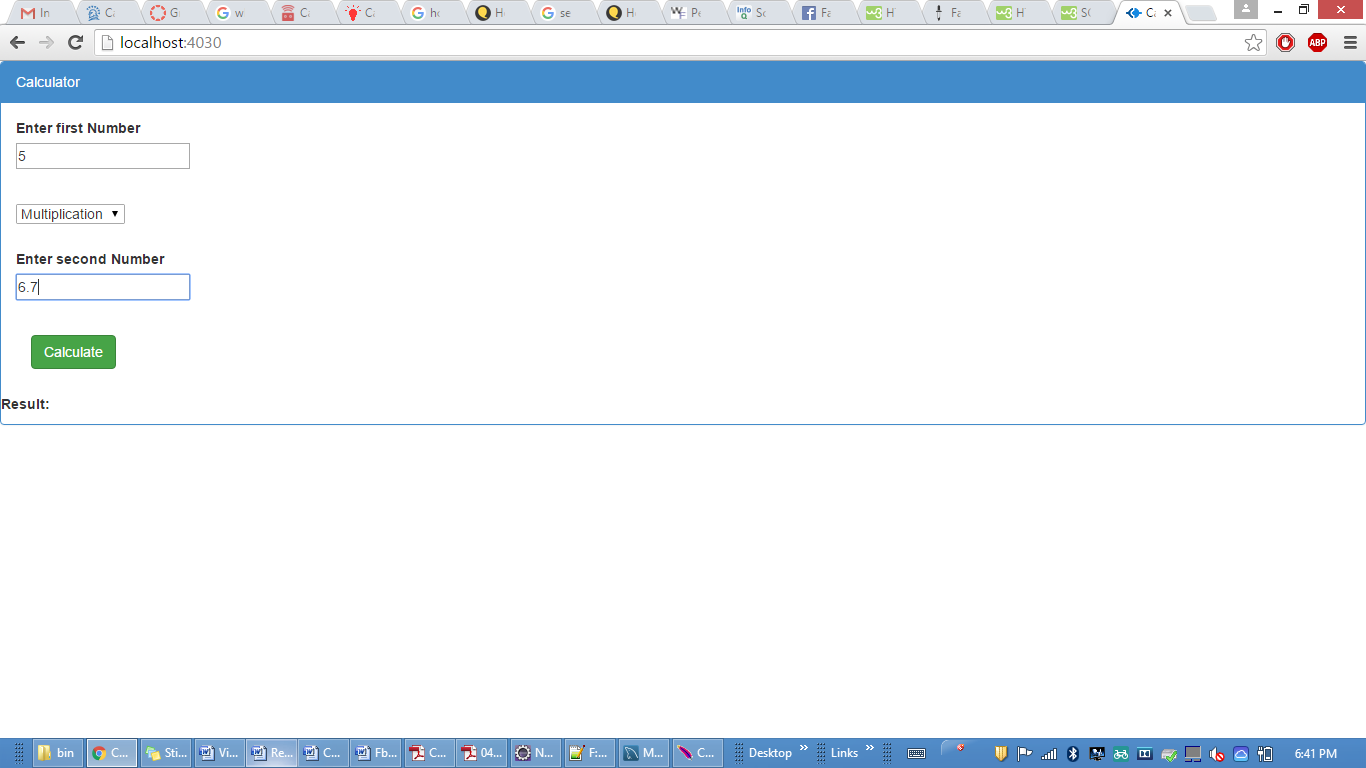


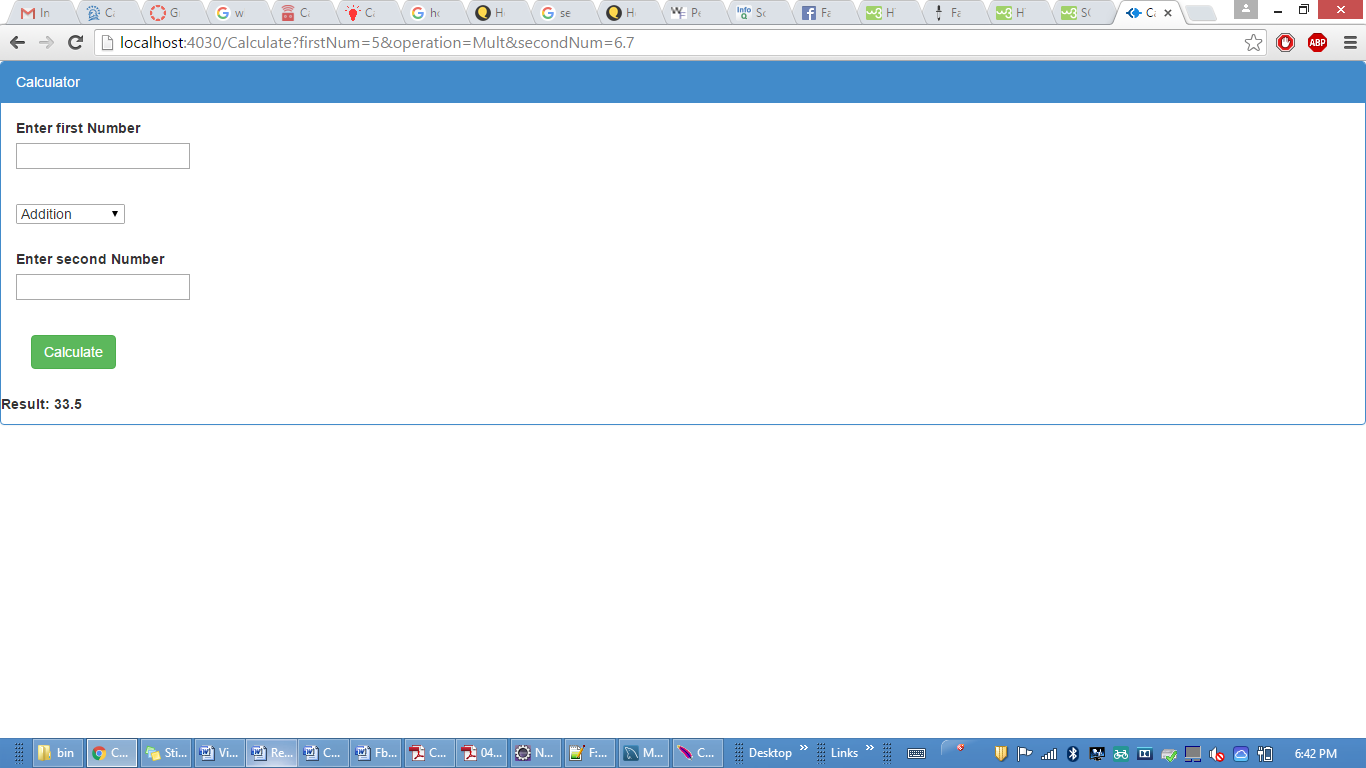
Subtraction



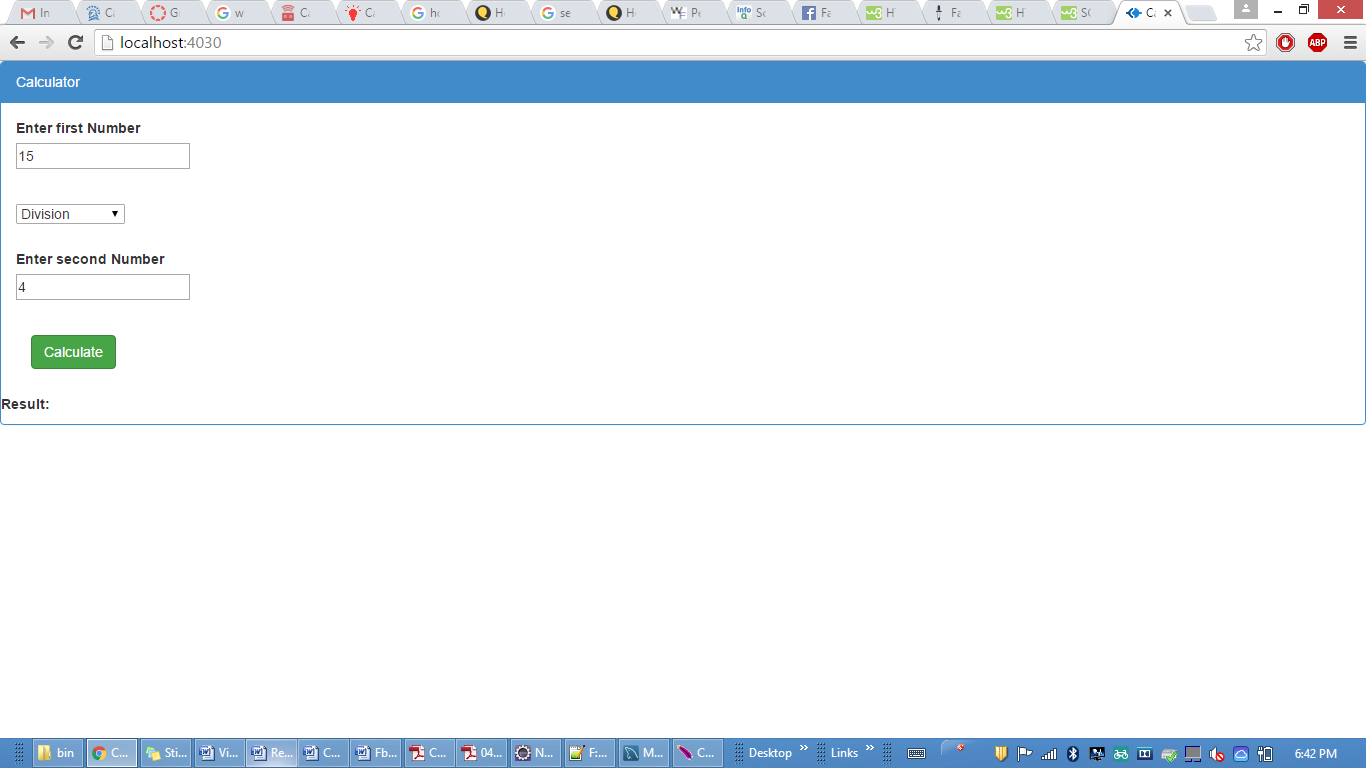


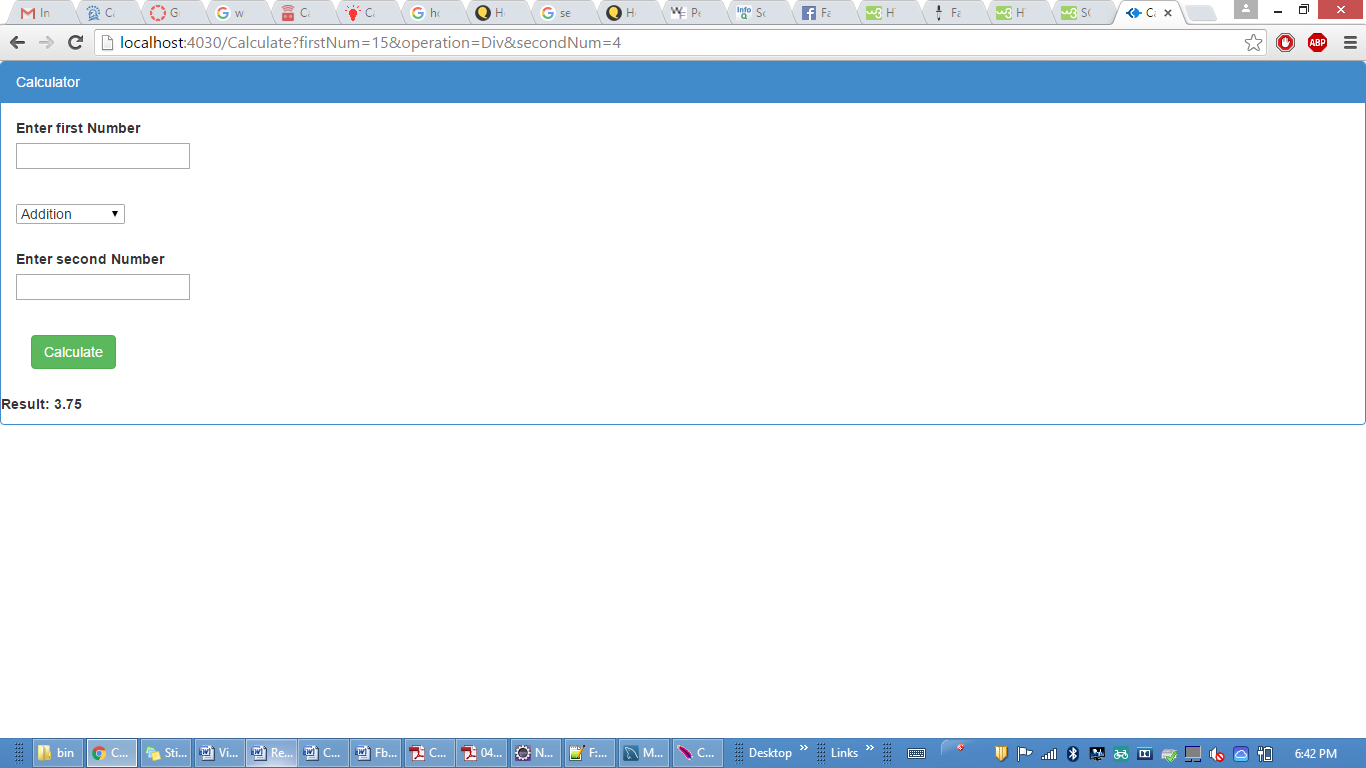
Multiplication



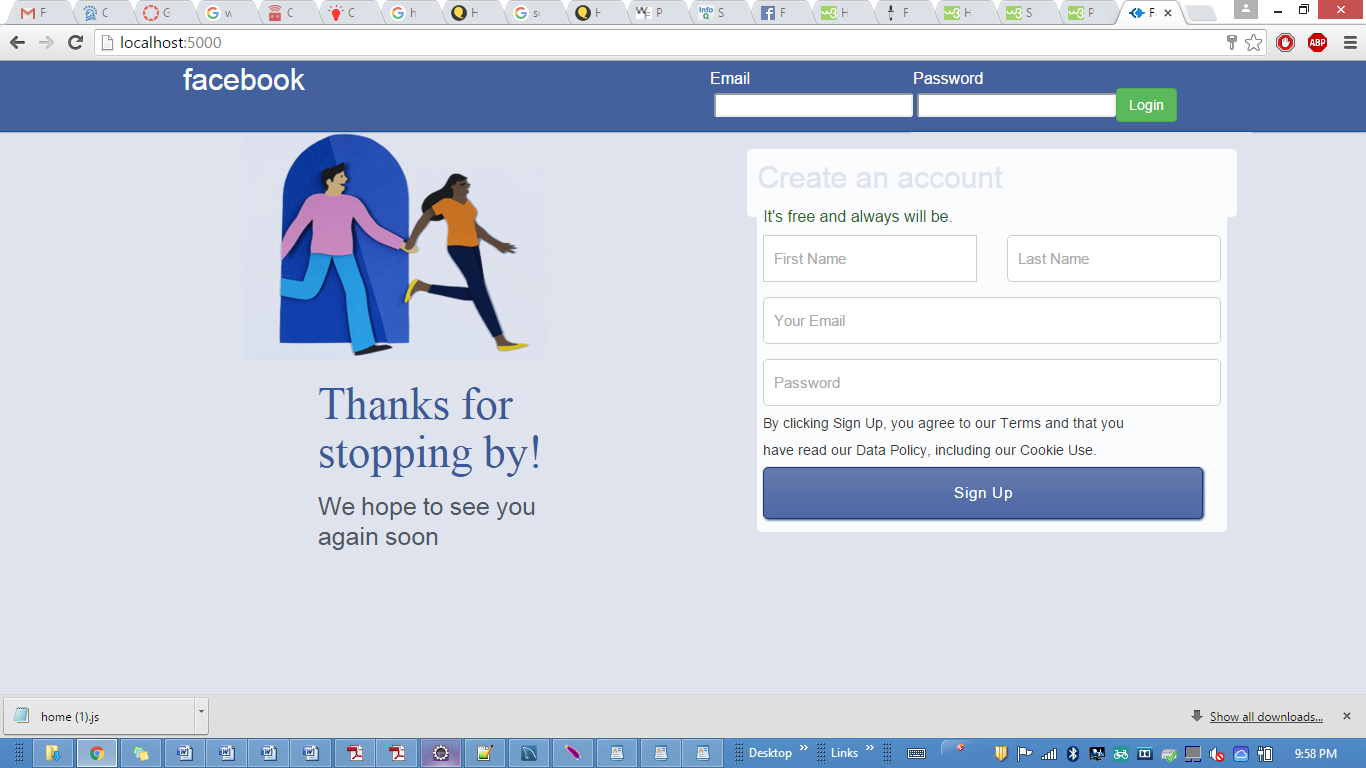


Division

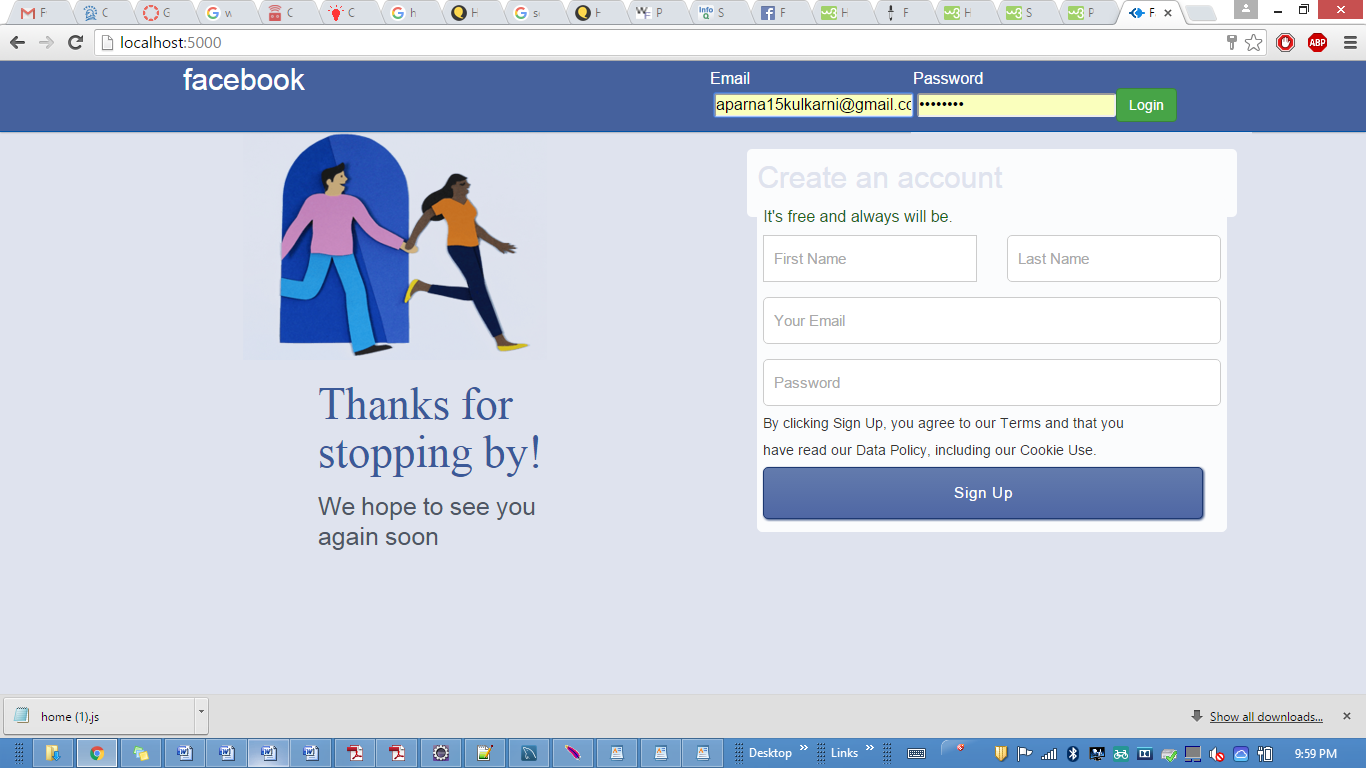




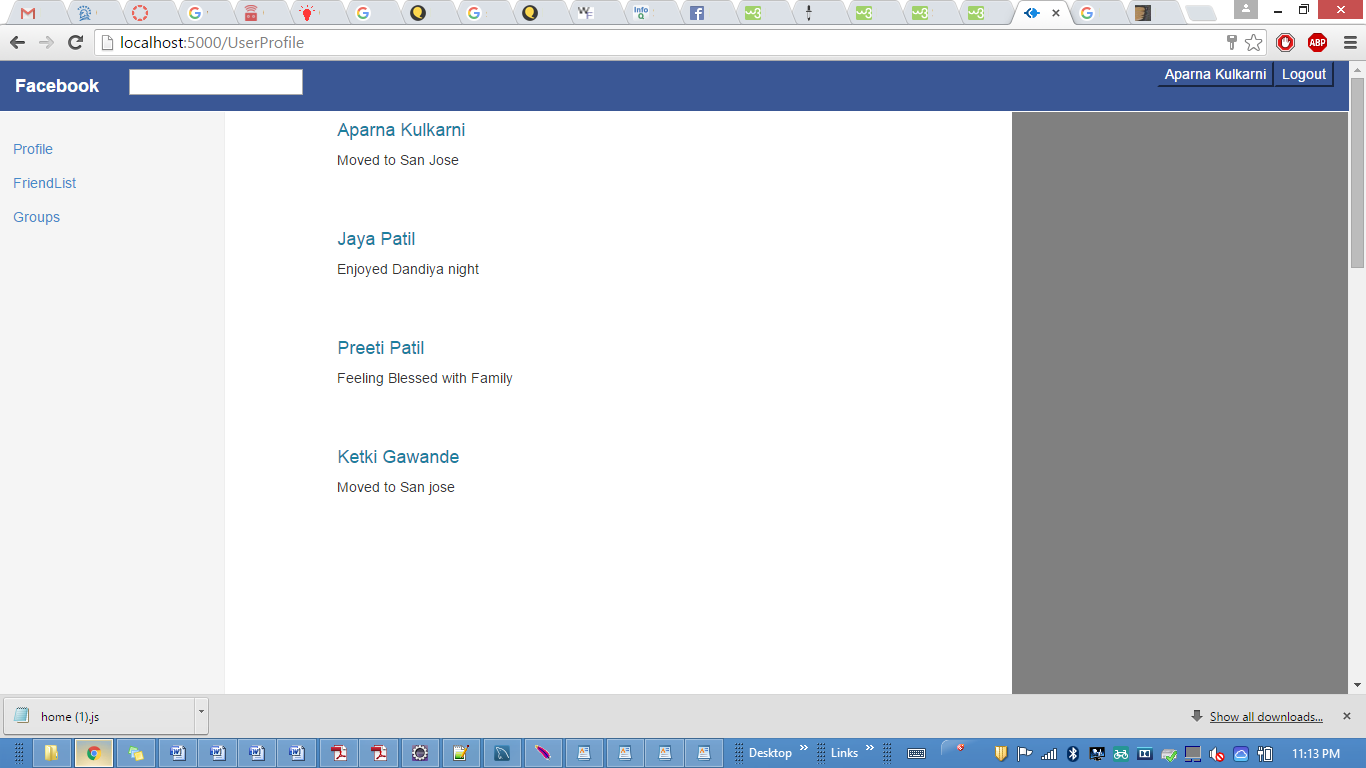
**Facebook**



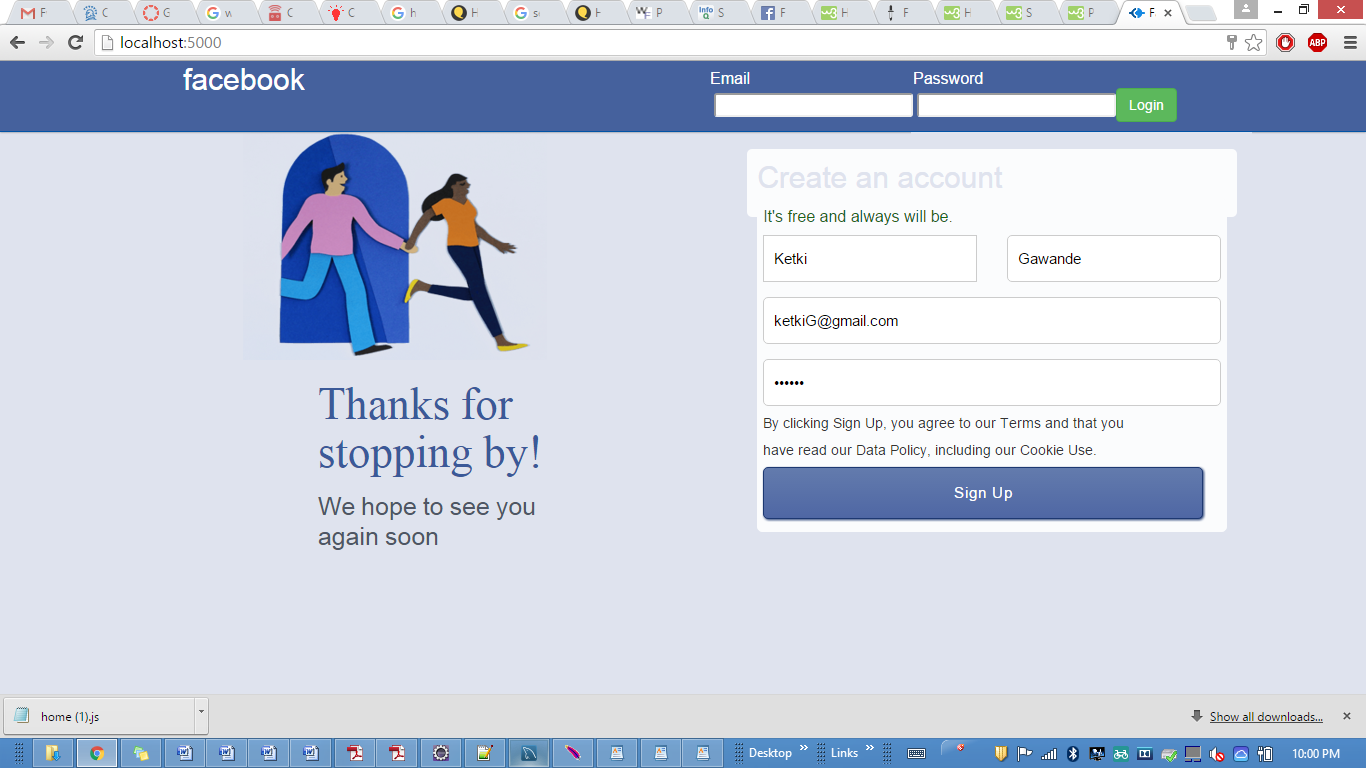
SignIn user

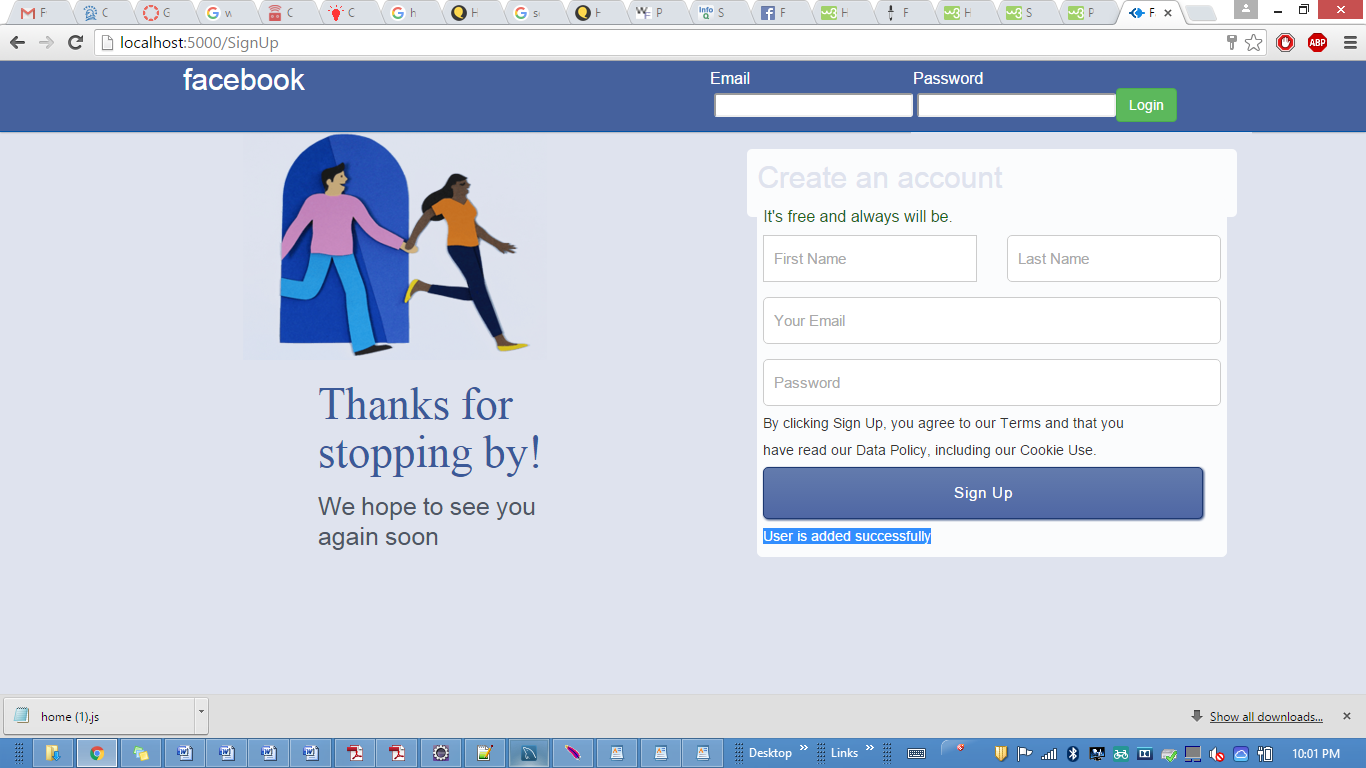


after signIn

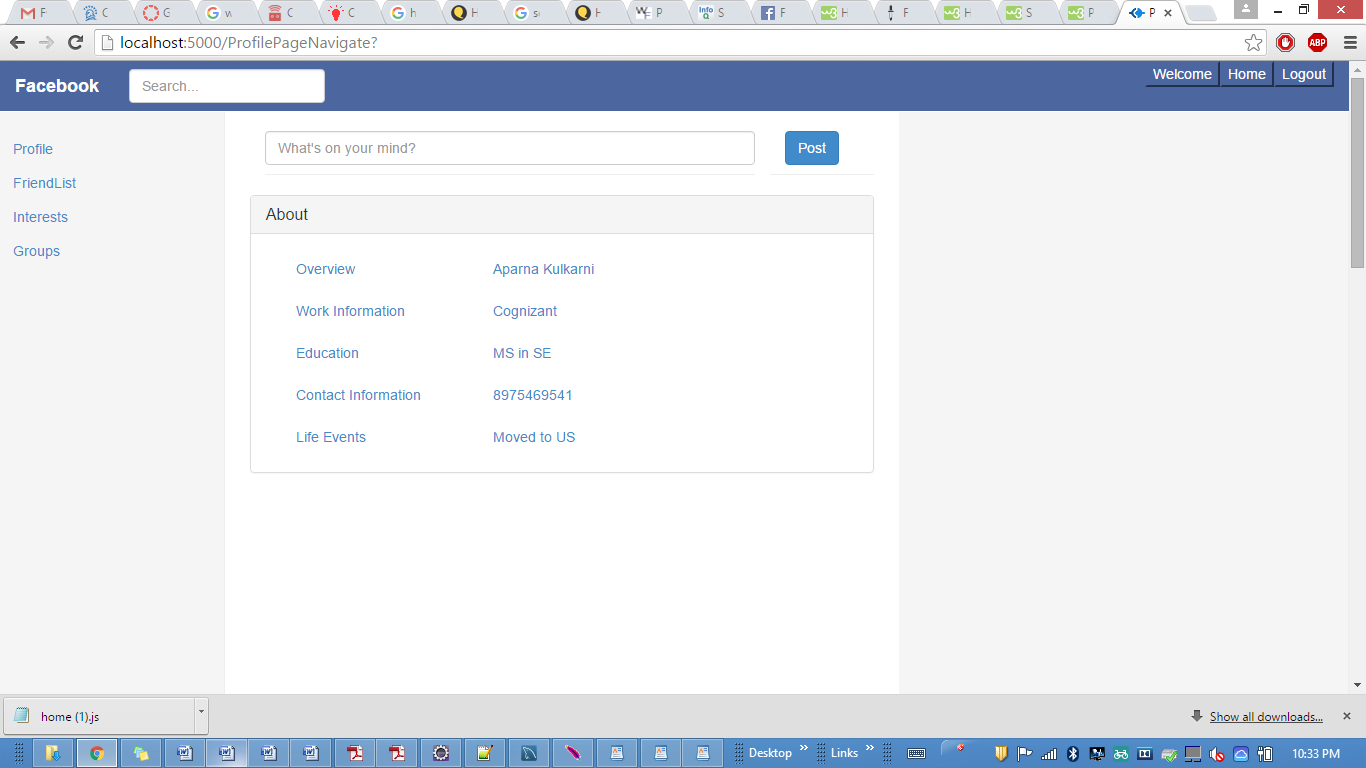


SignUp User

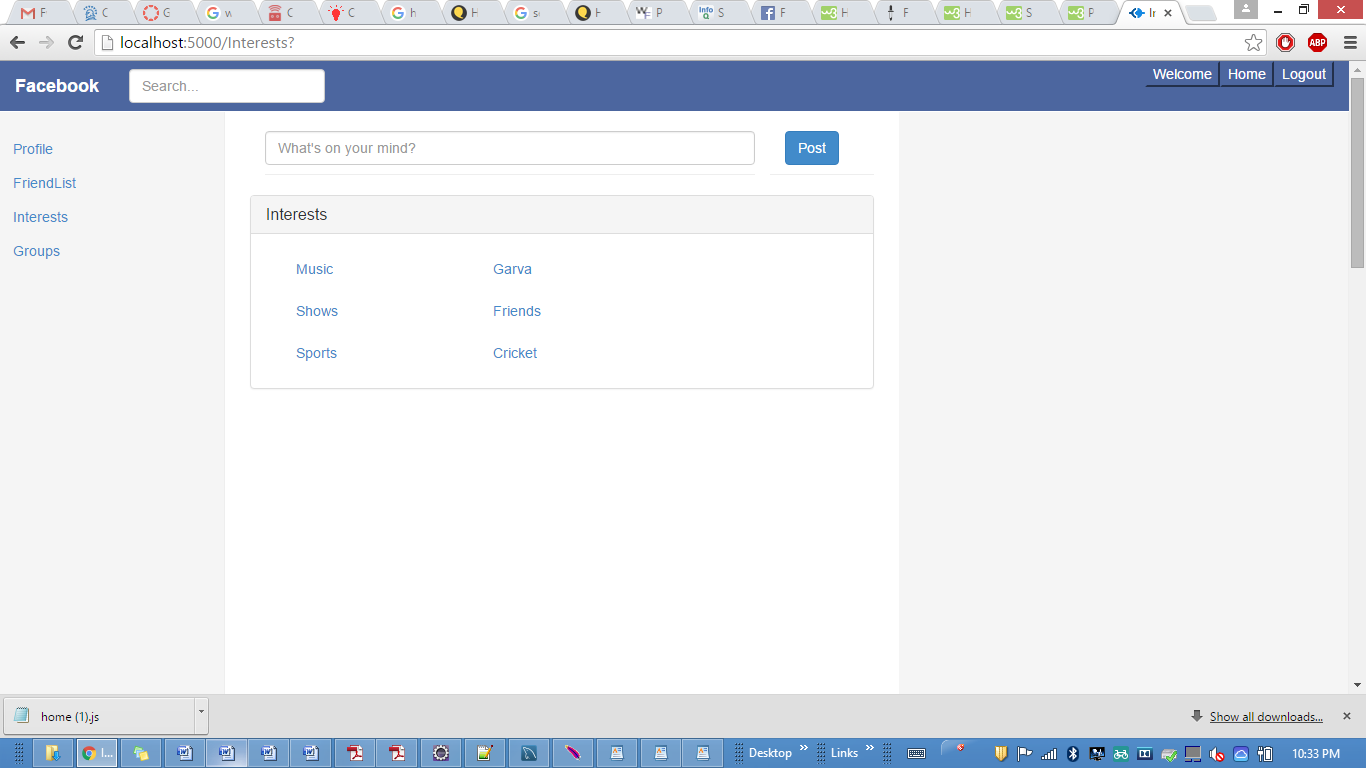




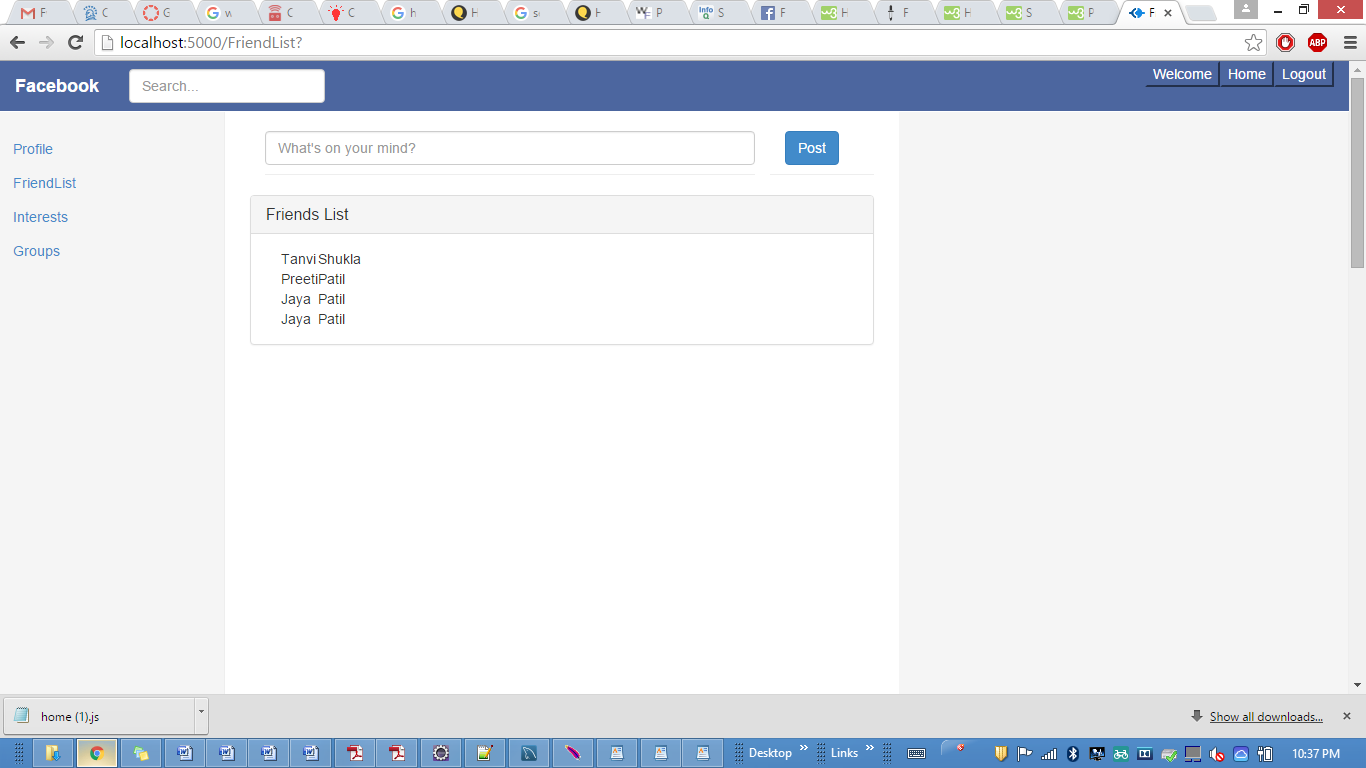
Profile page-About



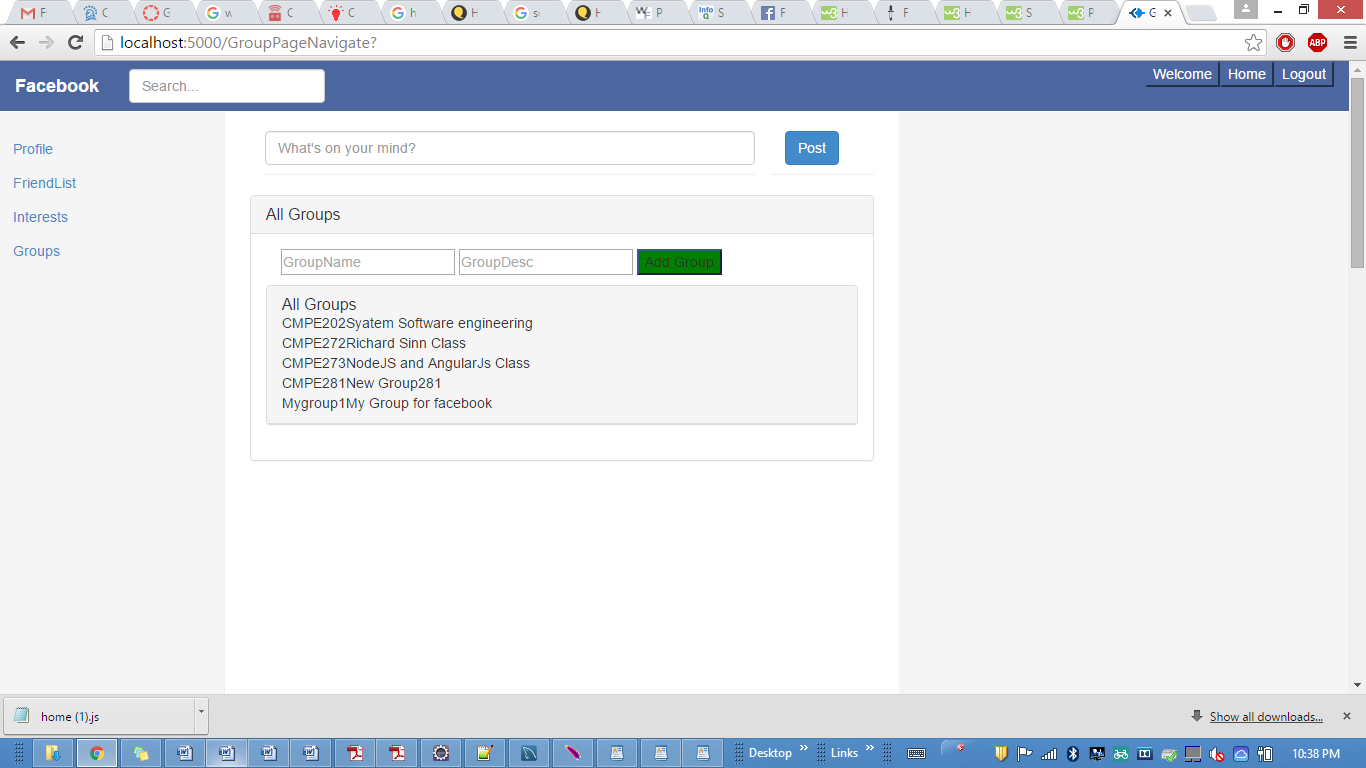
Interests



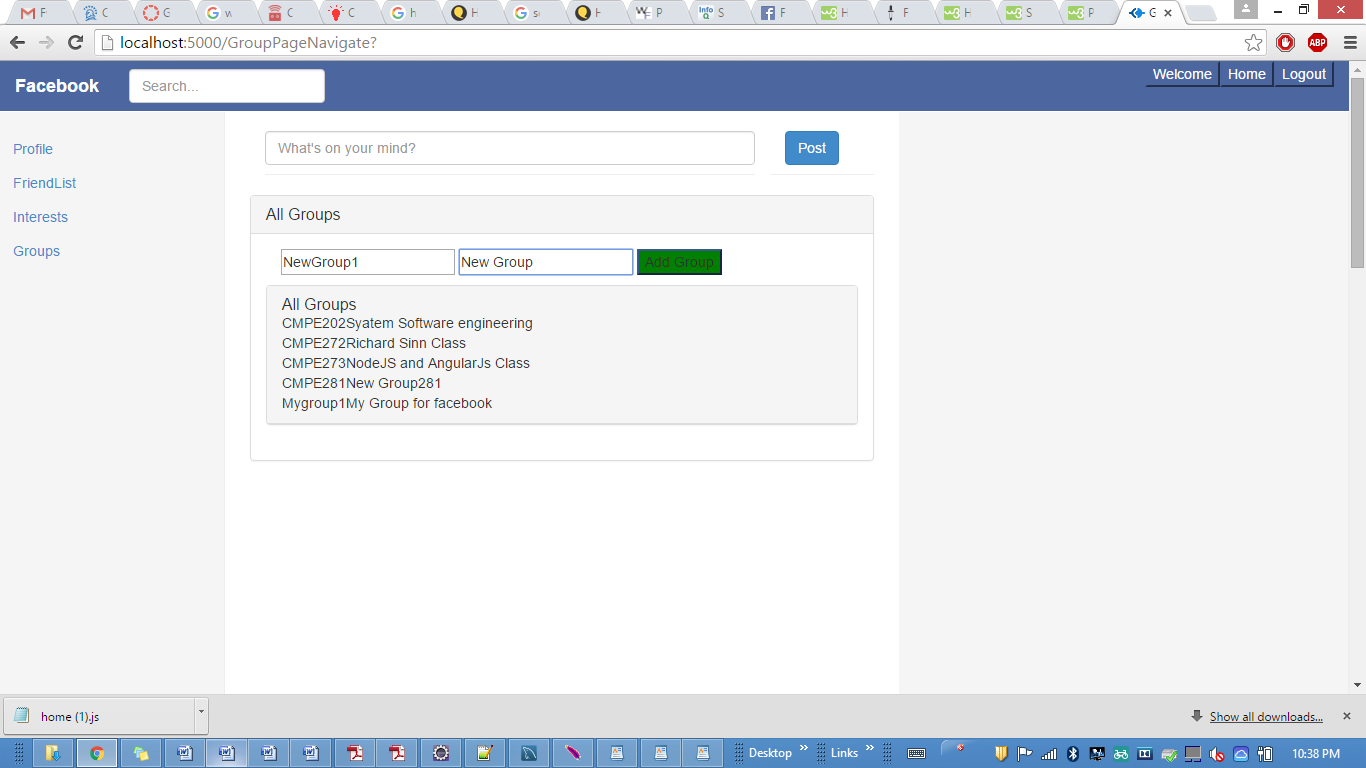
FriendList

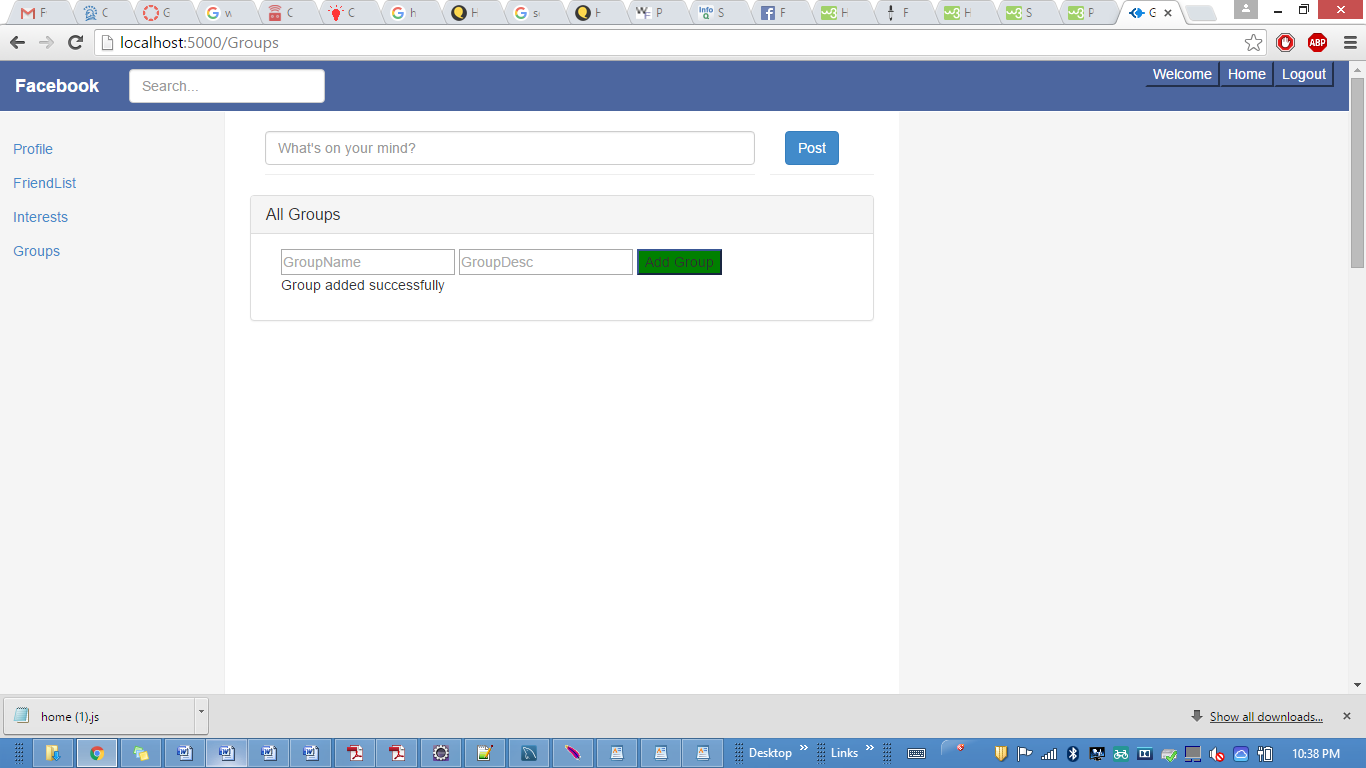


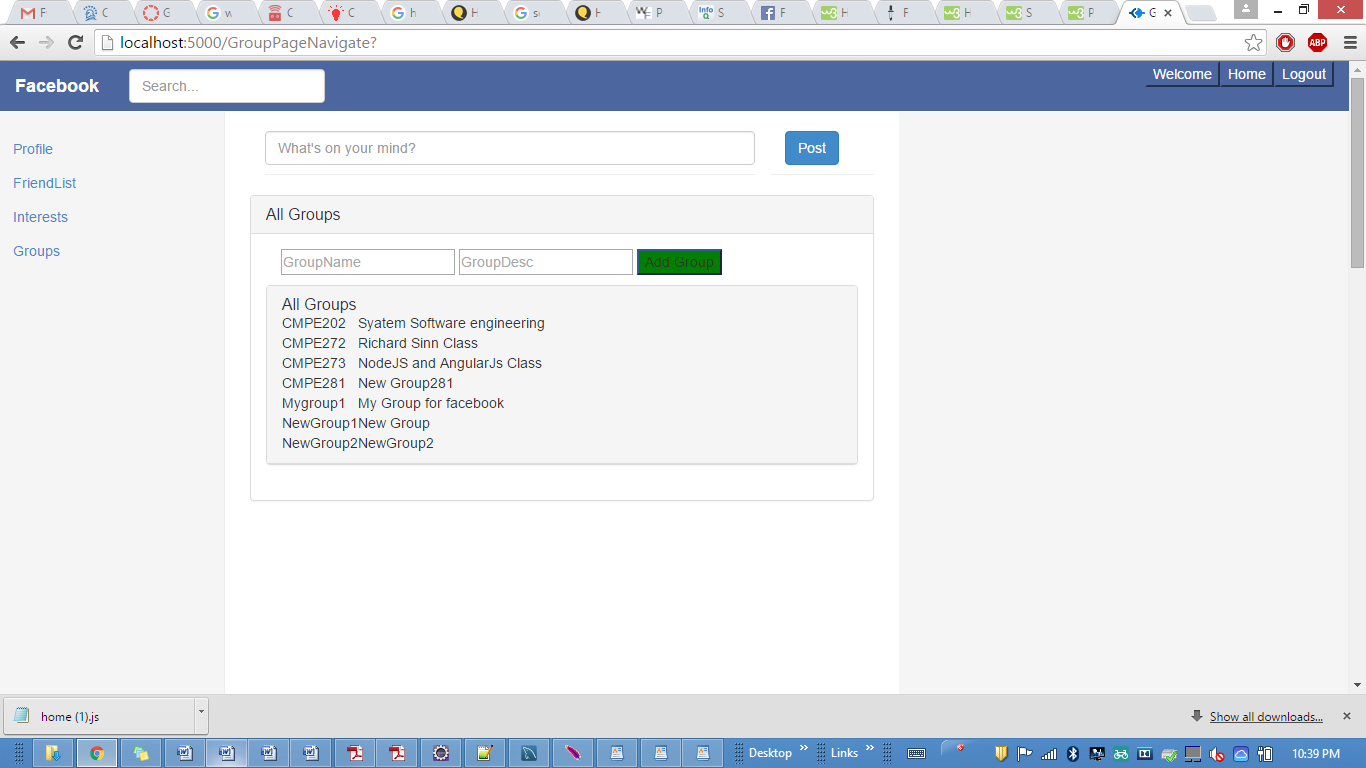
Groups



Add Groups



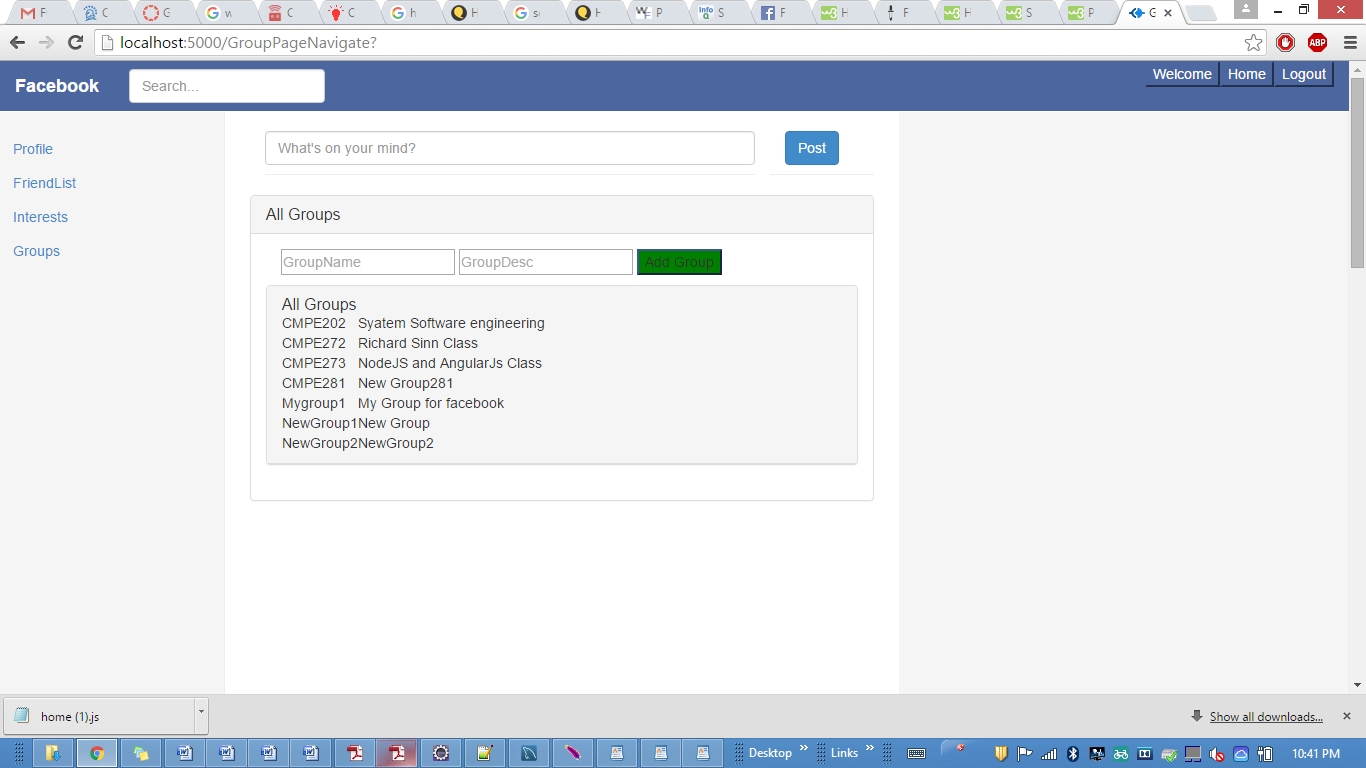


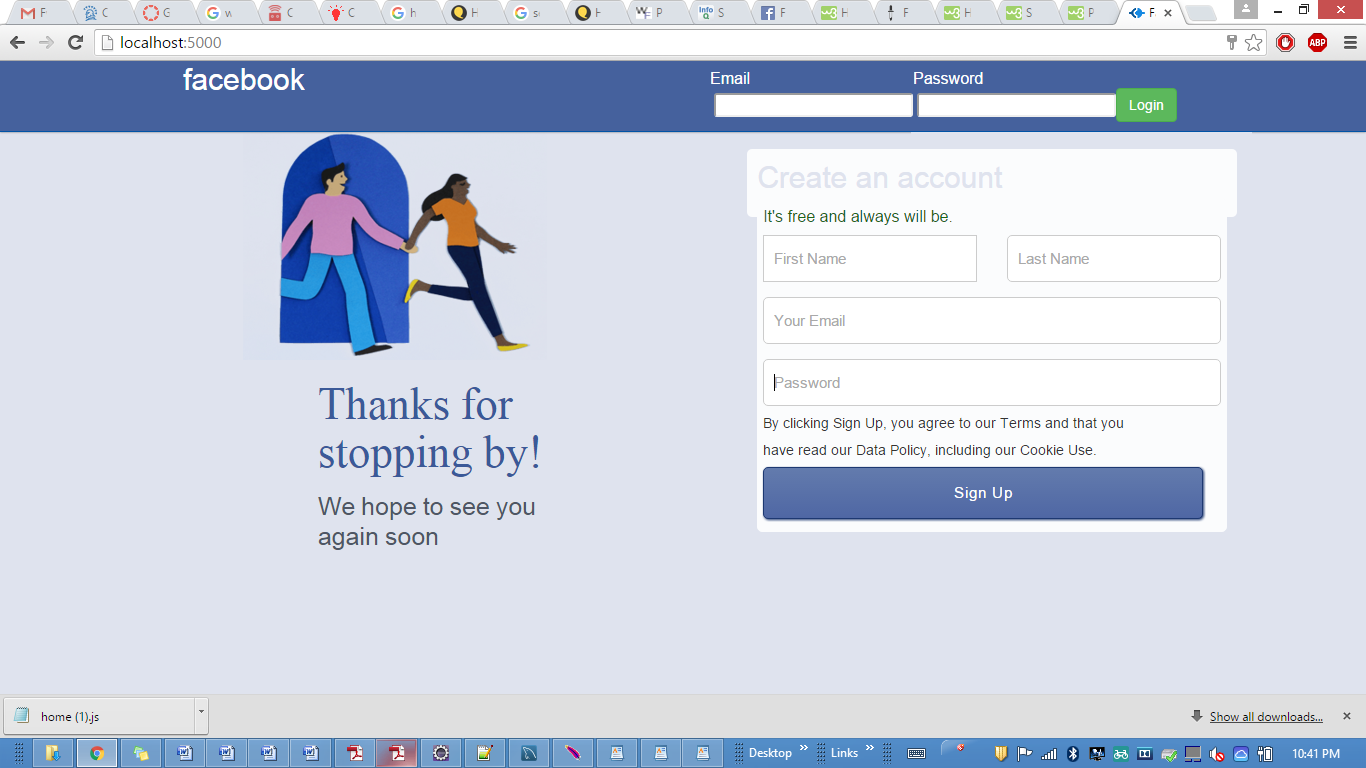


Delete Group

SignOut

clicking logout

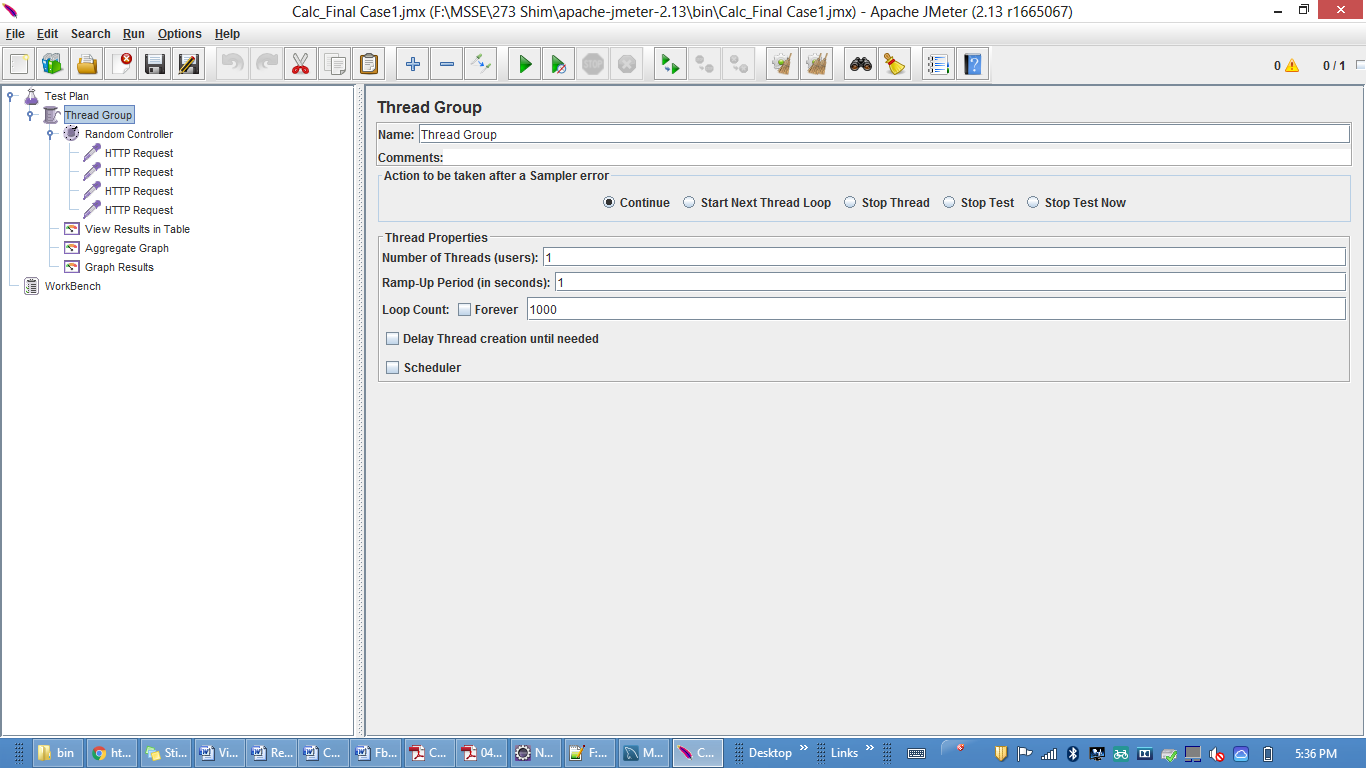


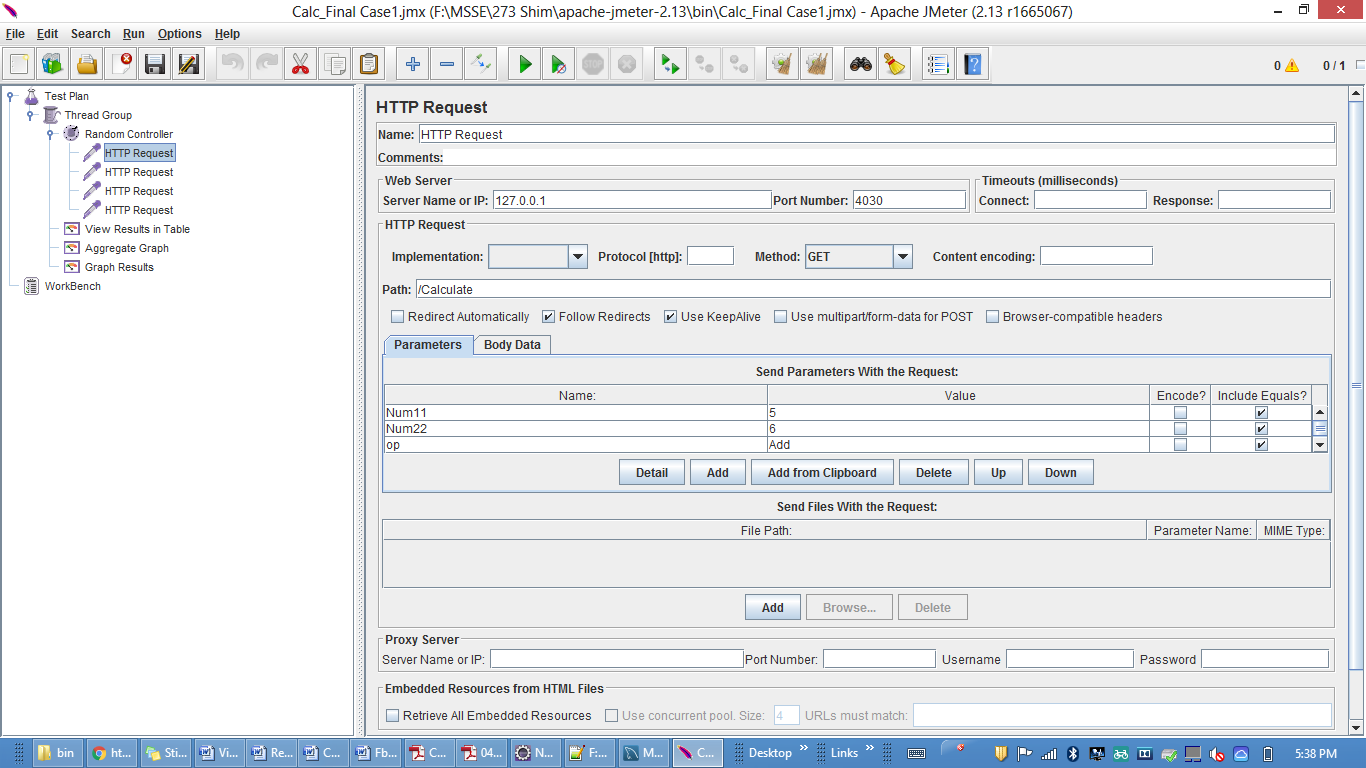
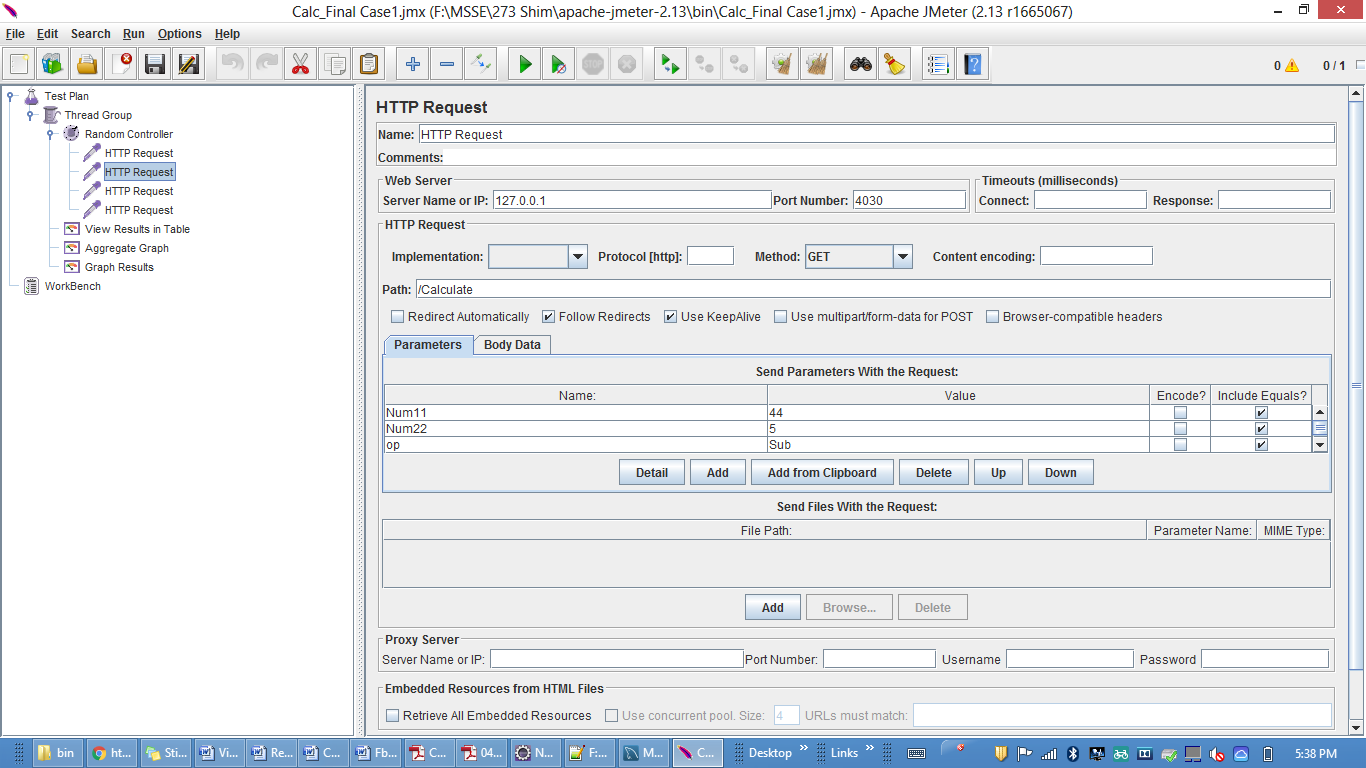


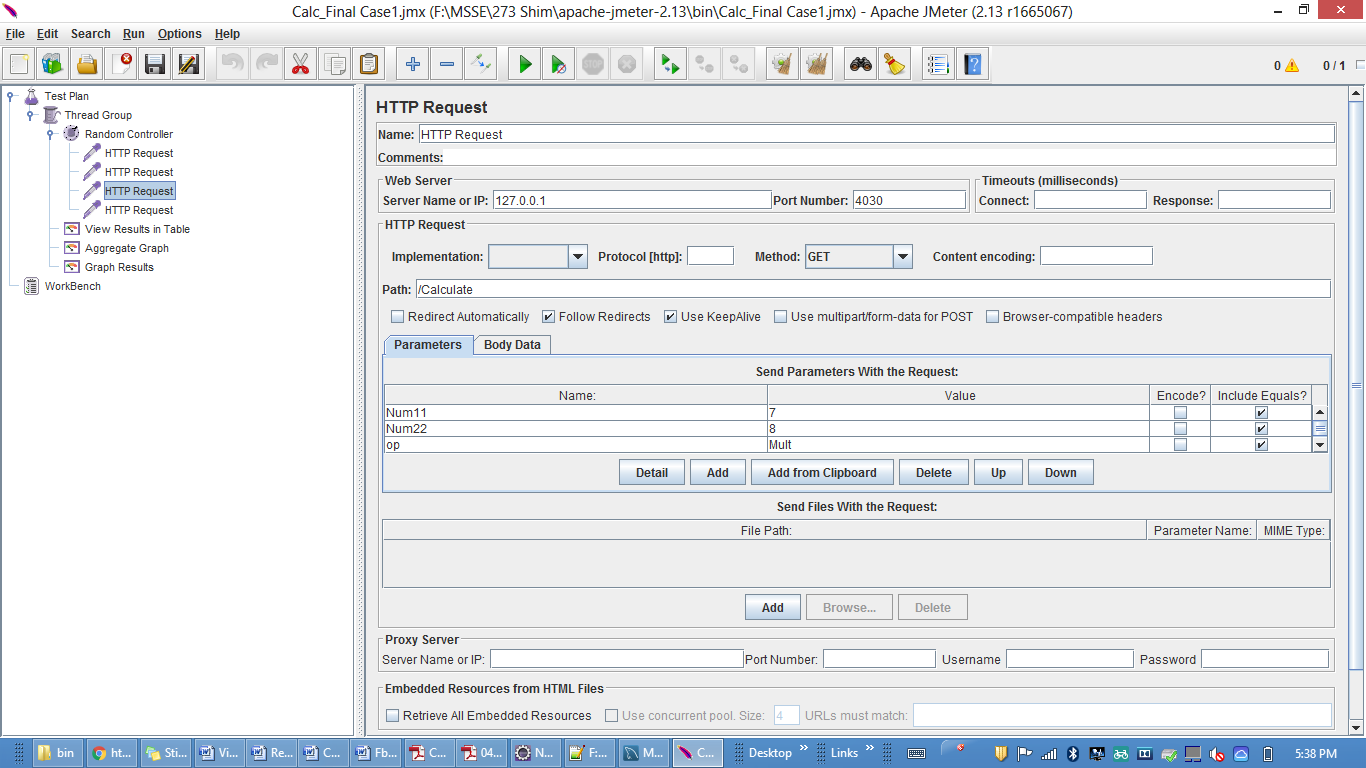
**Performances:**

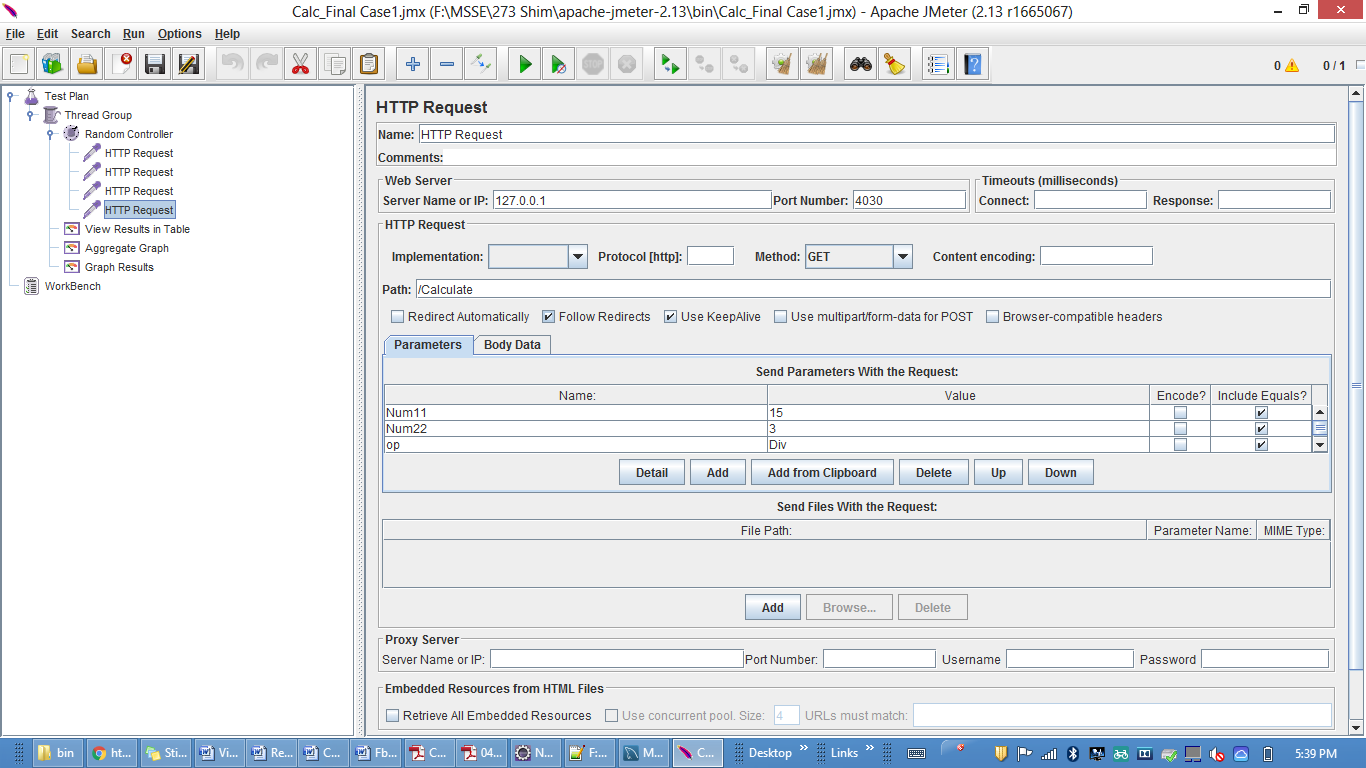
**Calculator**

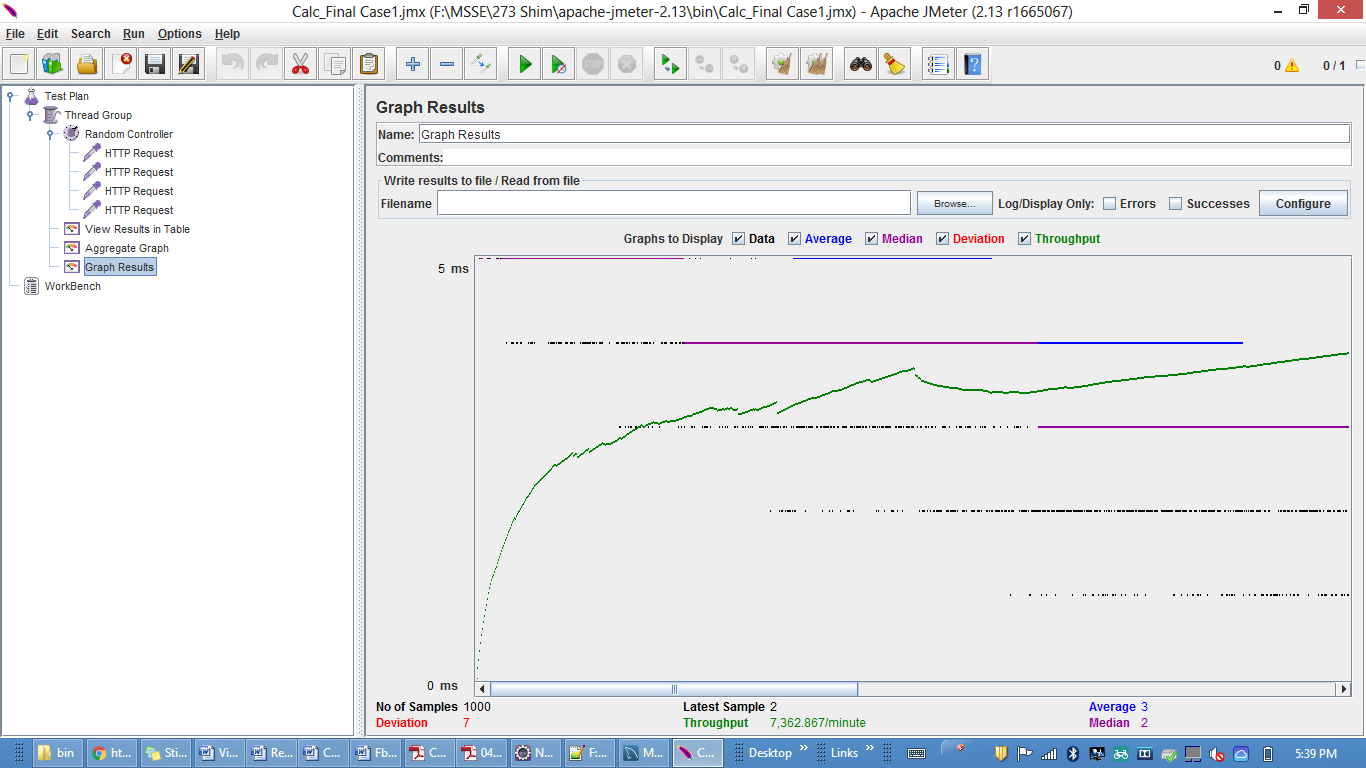
**Test 1 for 1000 randomly selected tasks:**

****

** **

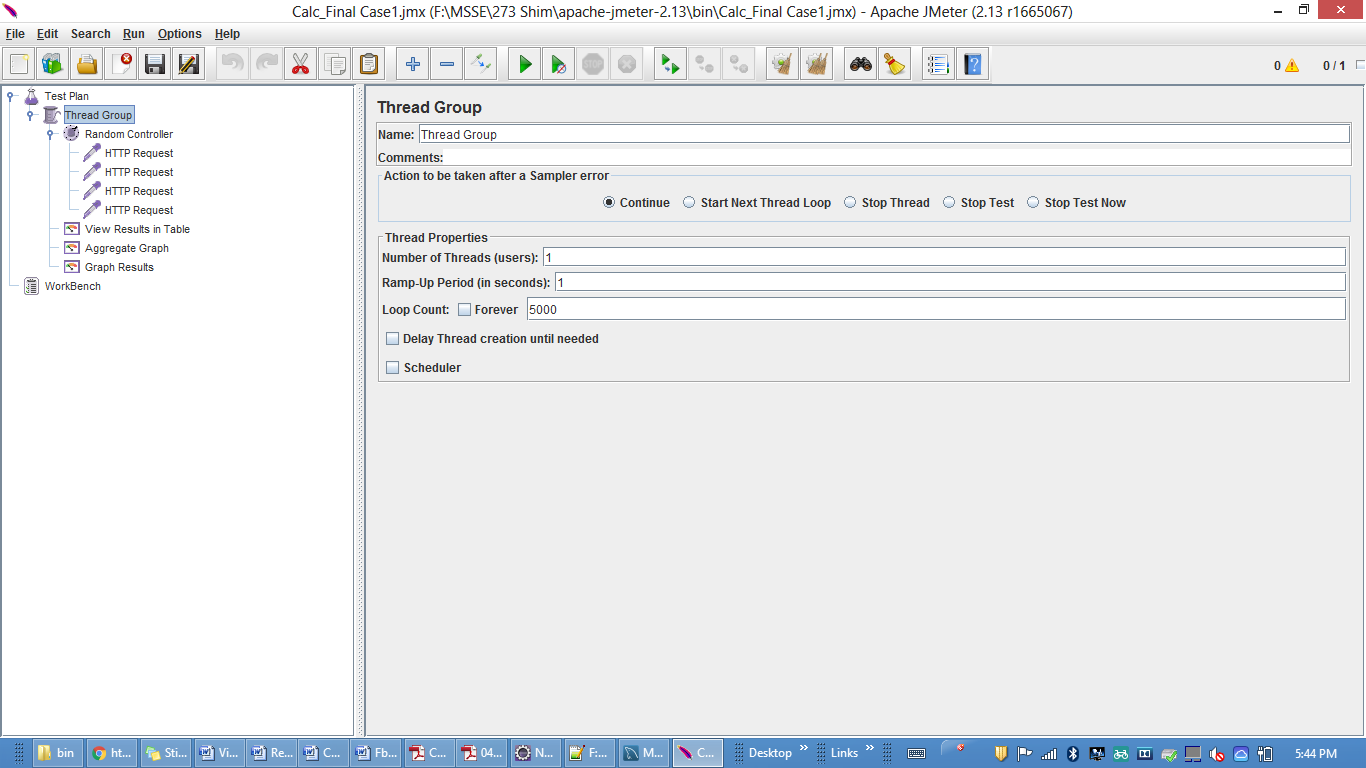
****

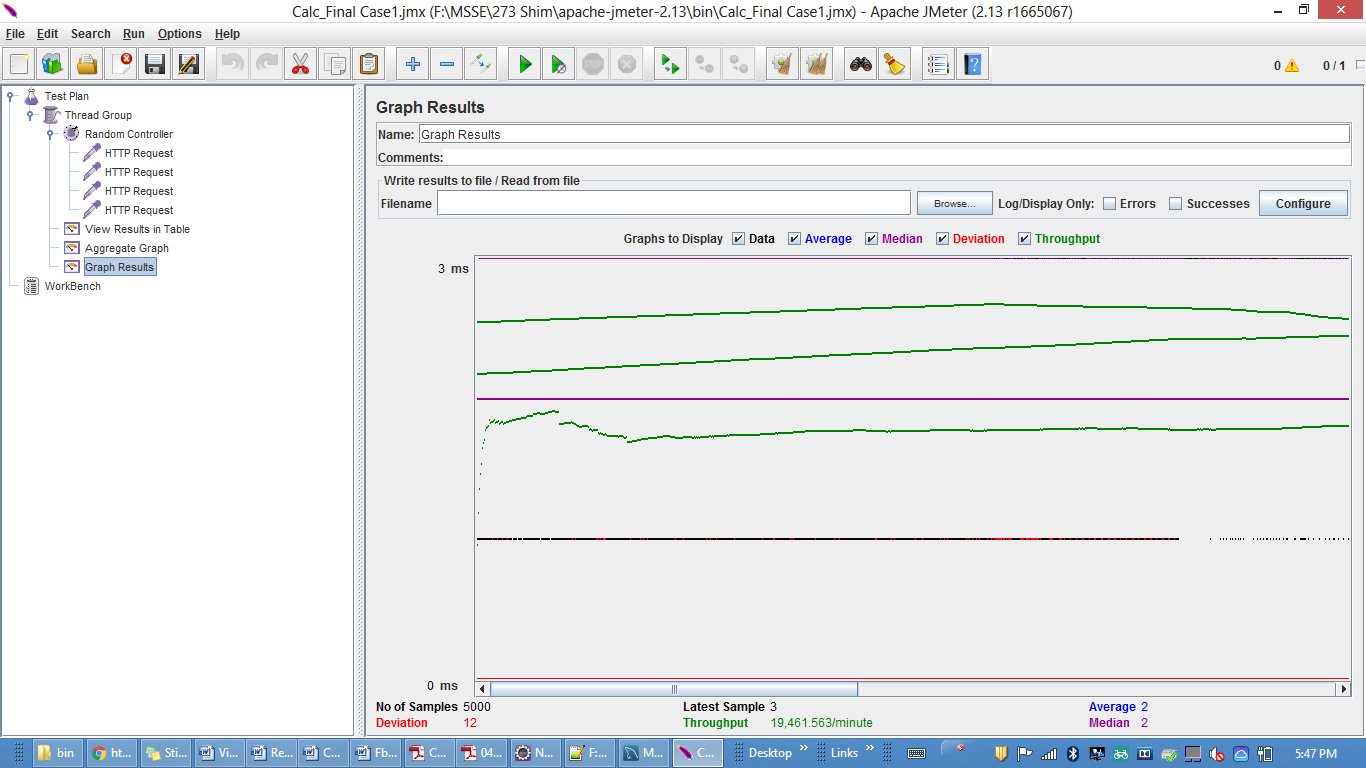
****

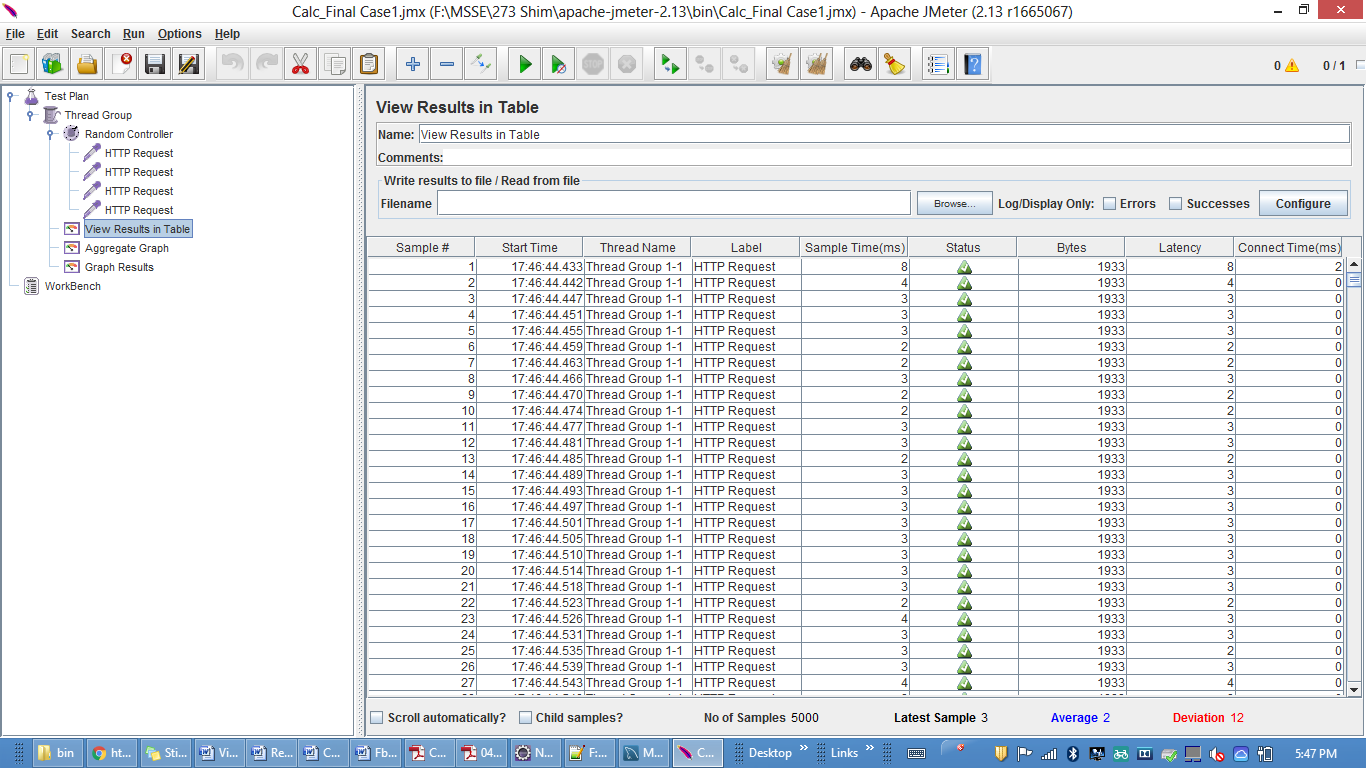
****

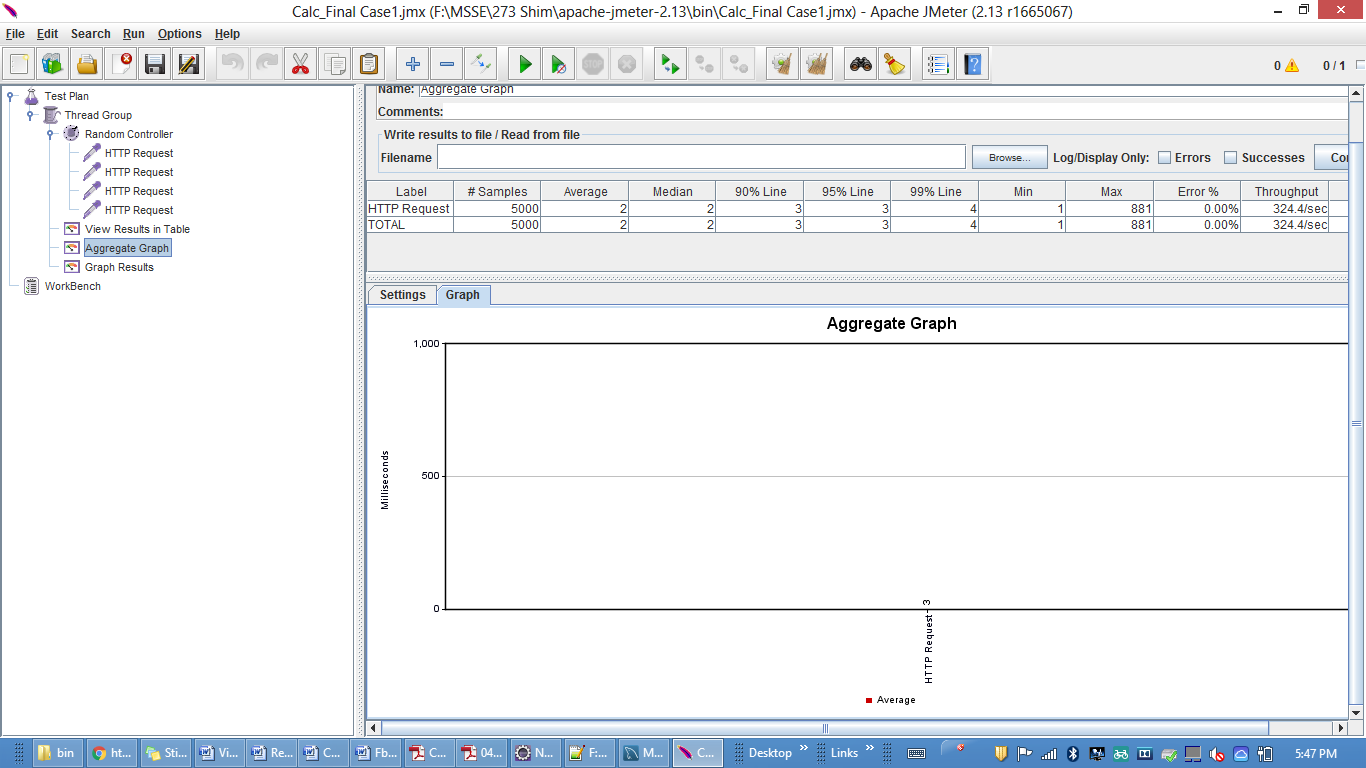
****

**Test 2 for 5000 randomly selected tasks:**

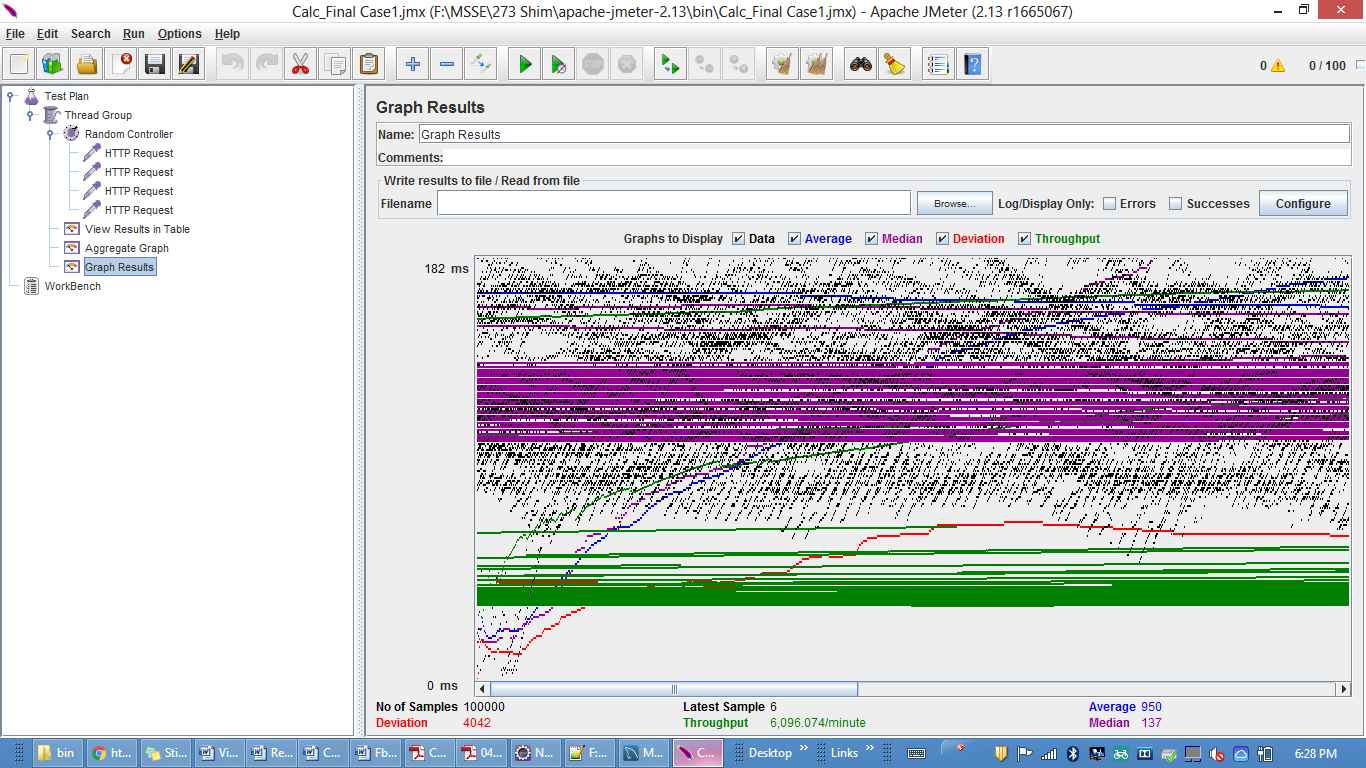
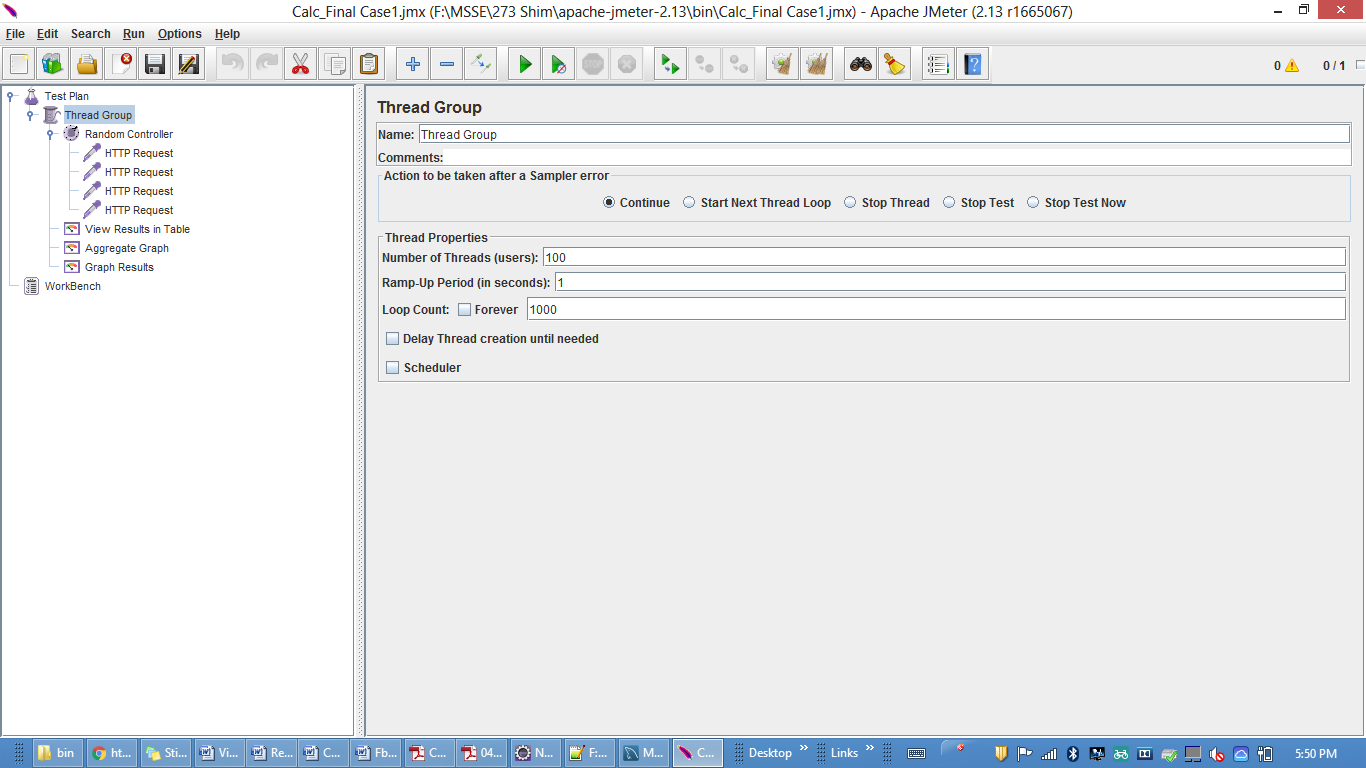
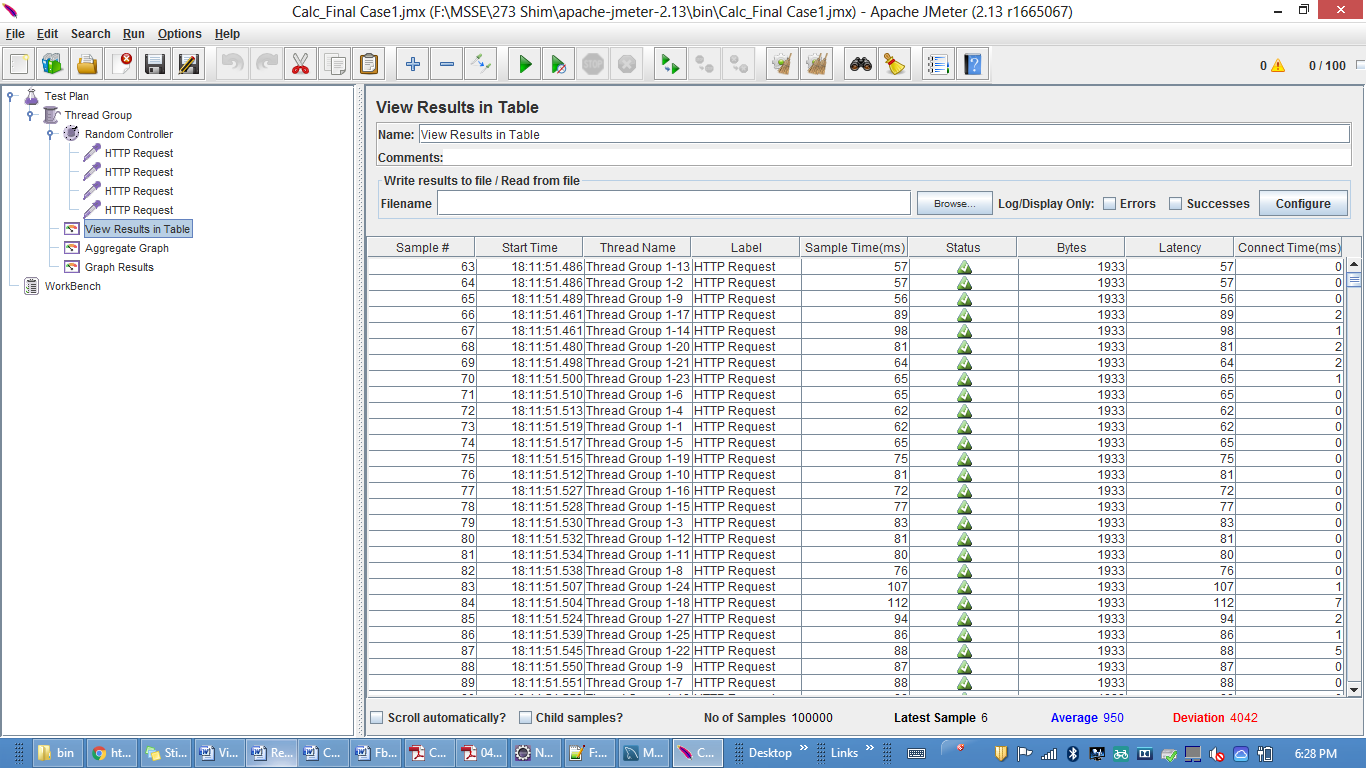
****

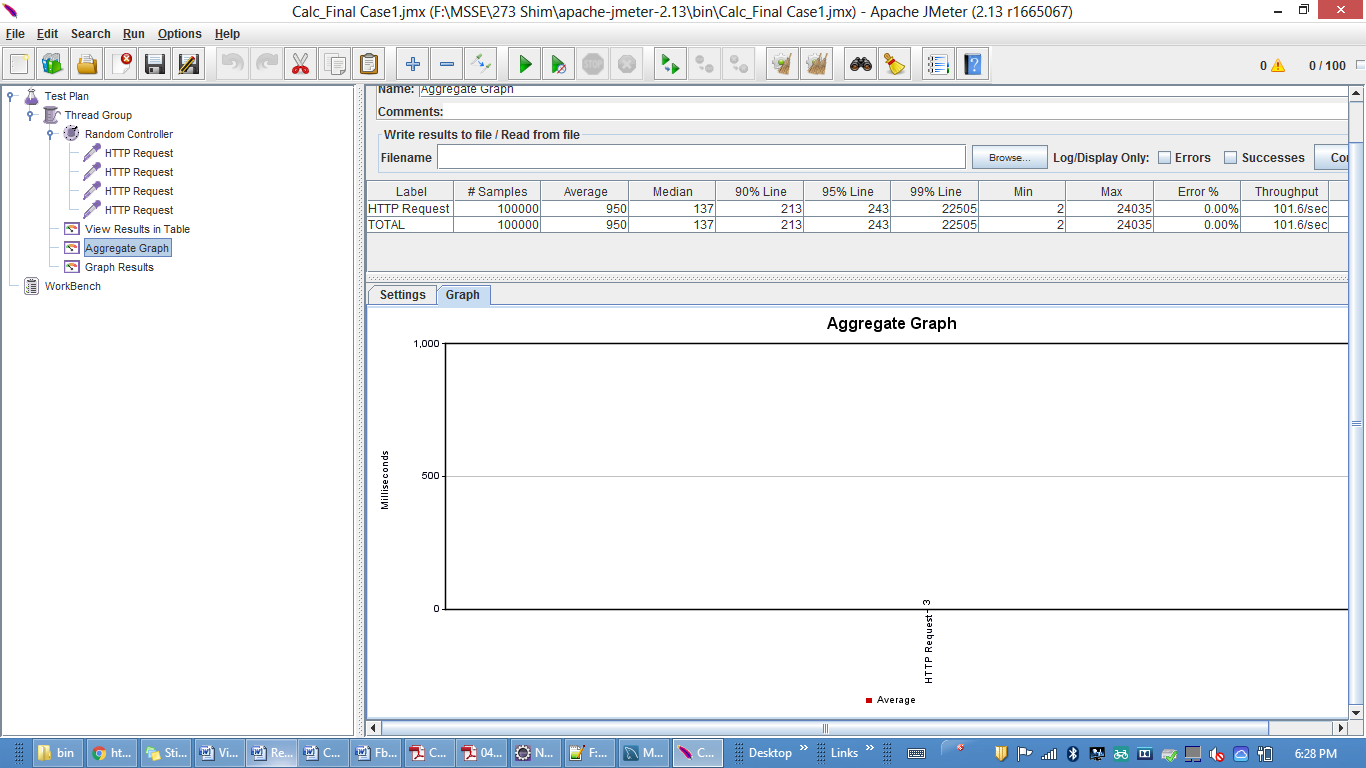
****

****

****

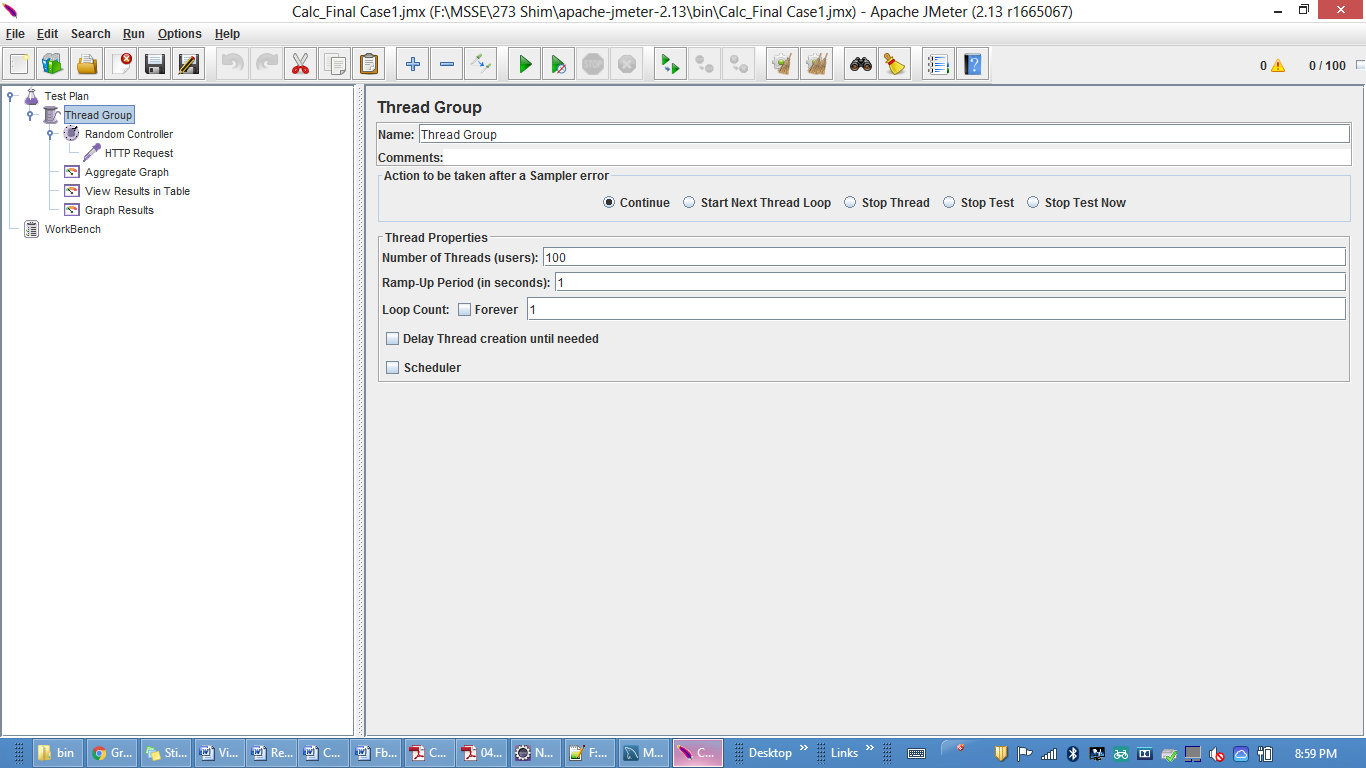
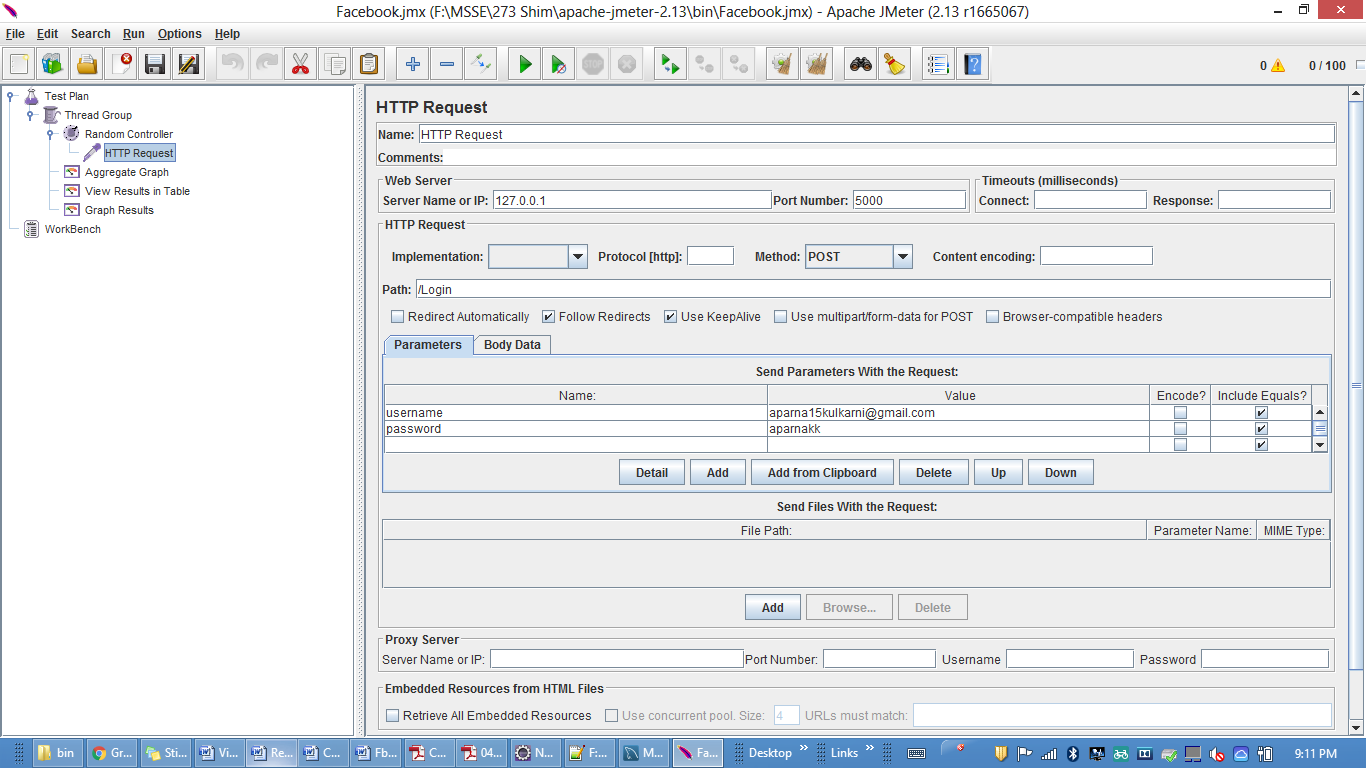
**Test 3 for 100 concurrent users 1000 randomly selected tasks:**

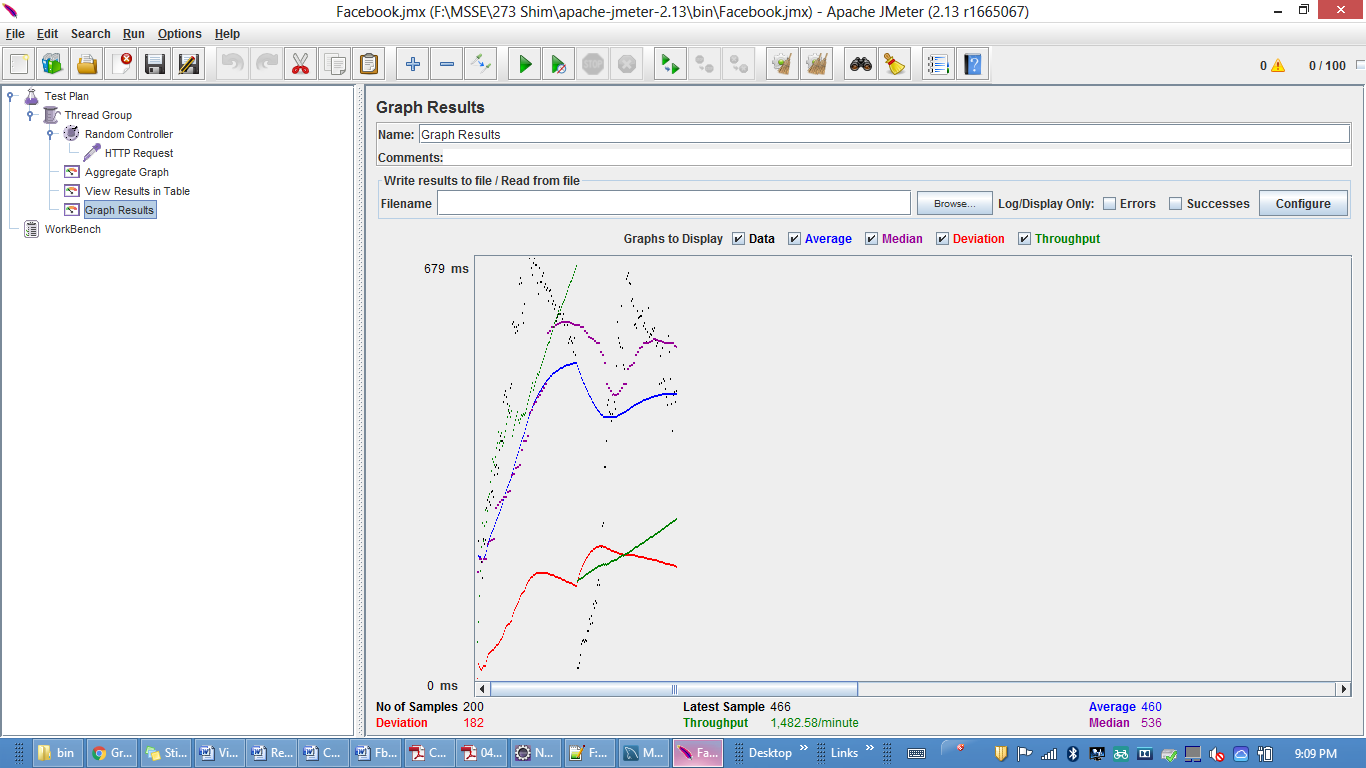
** **

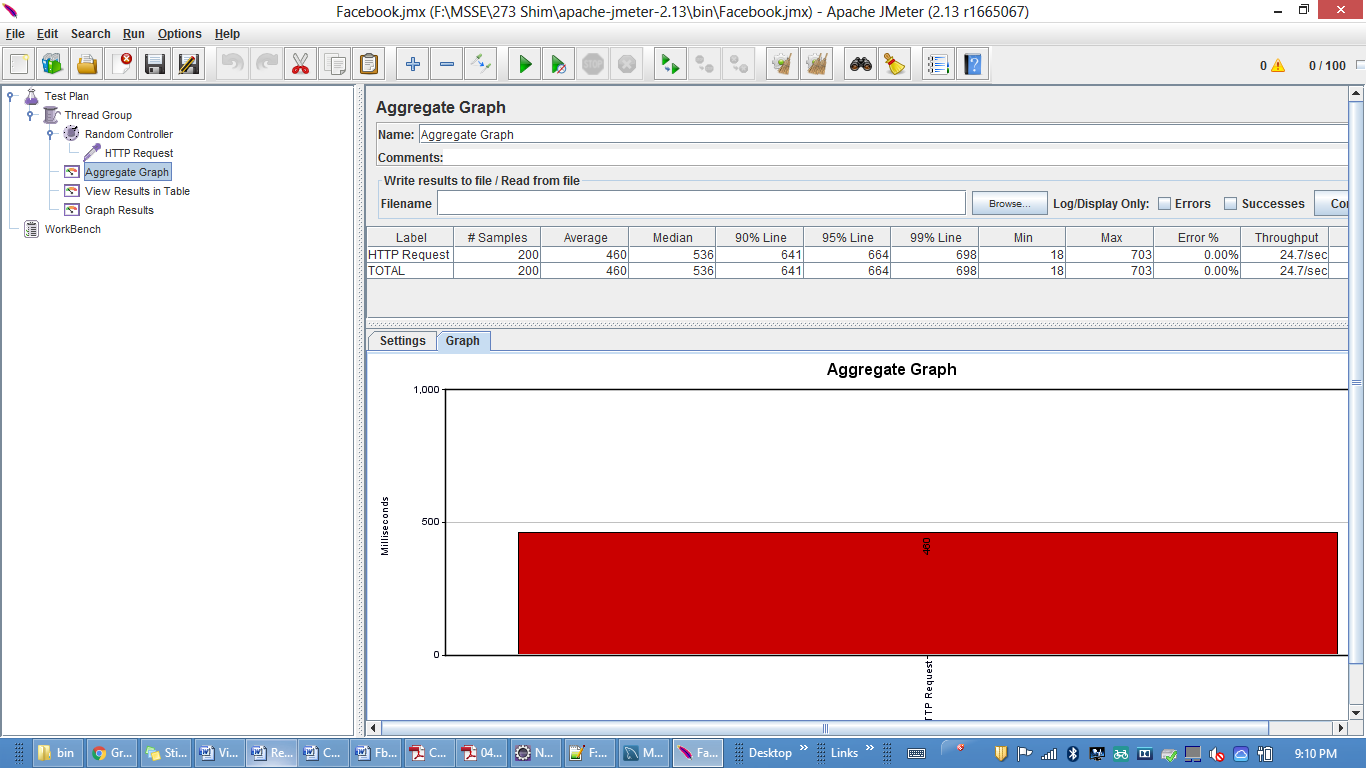
****

**Facebook**

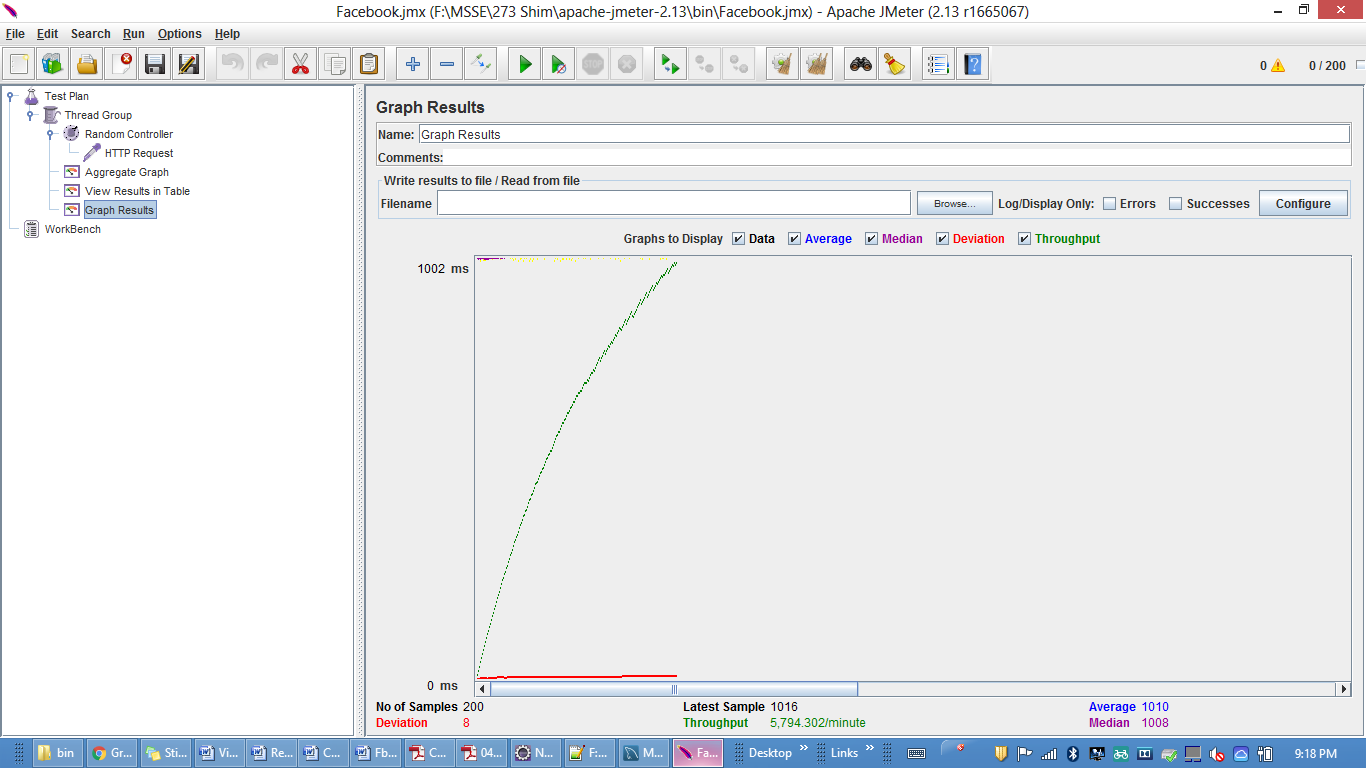
**Test 1: 100 concurrent users**

****

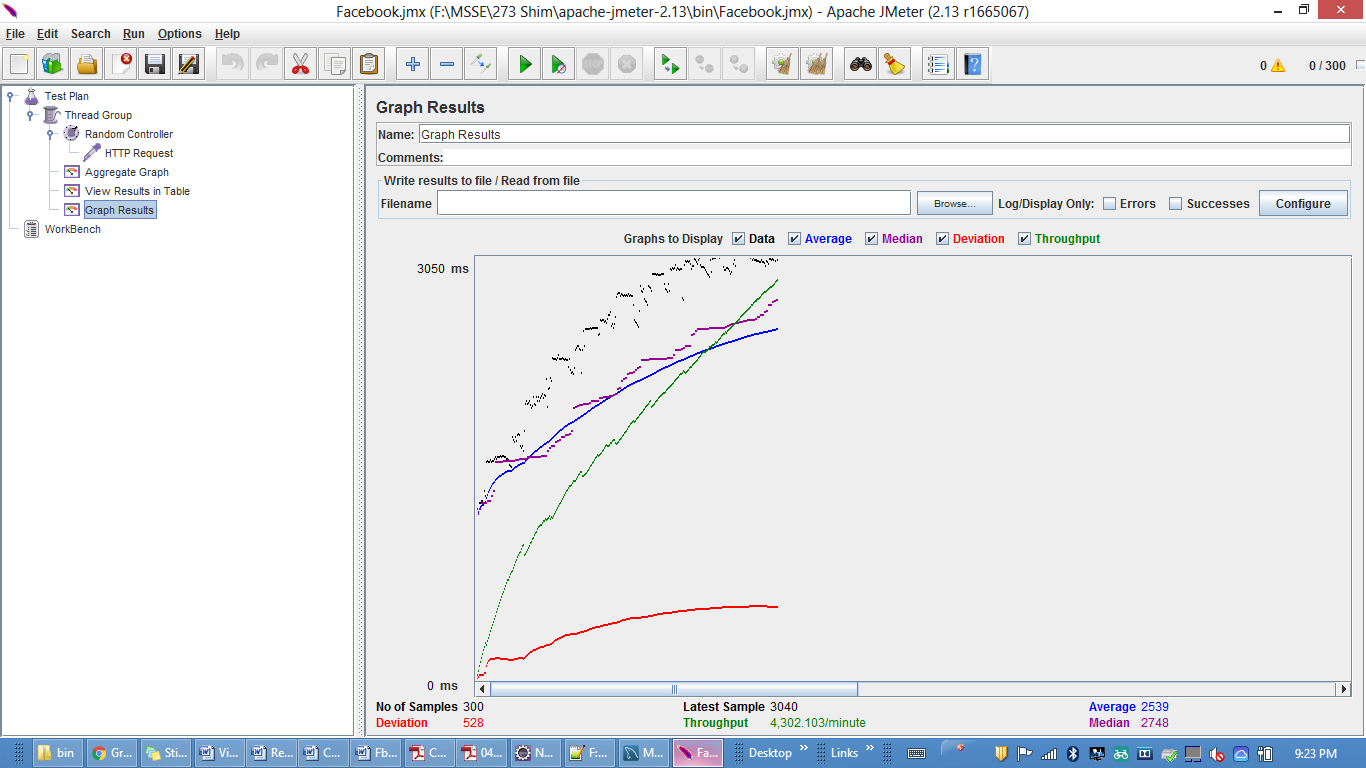
****

****

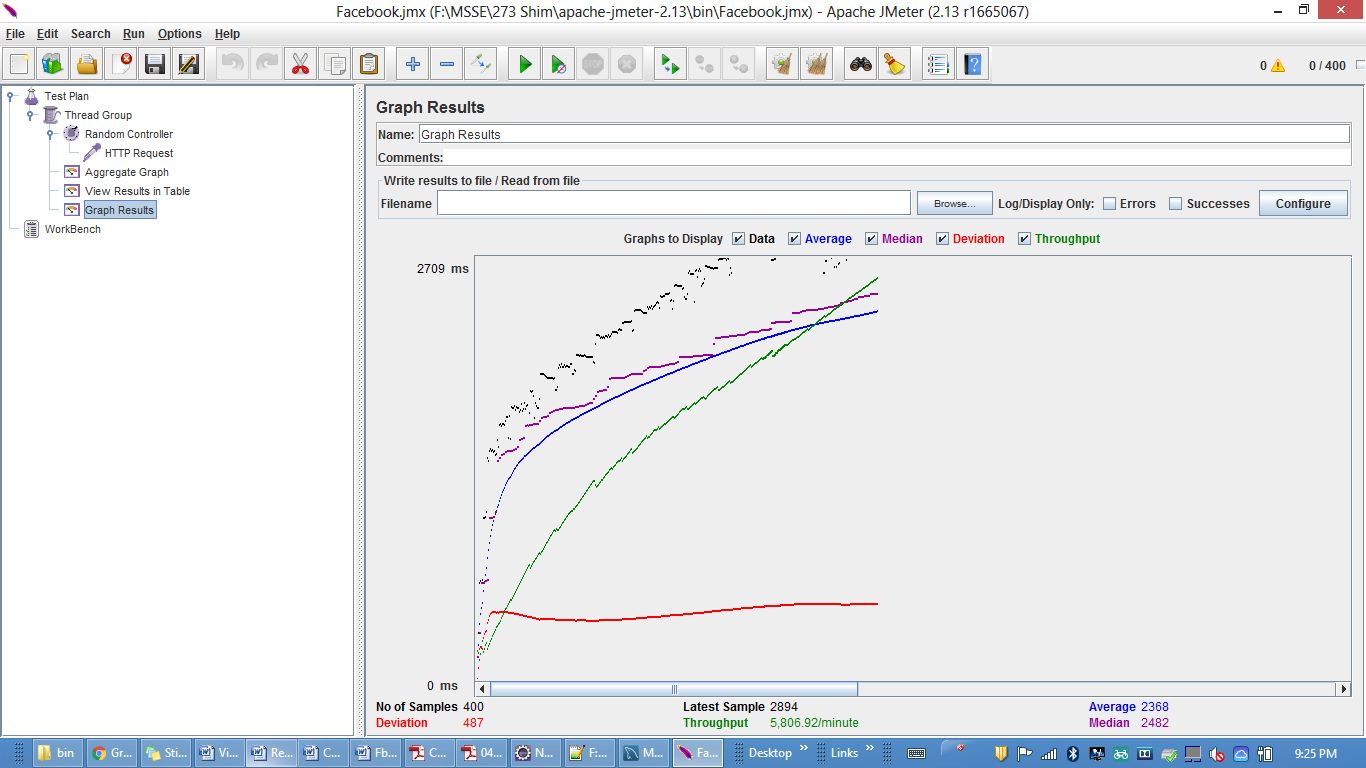
**Test 2: 200 concurrent users**

****

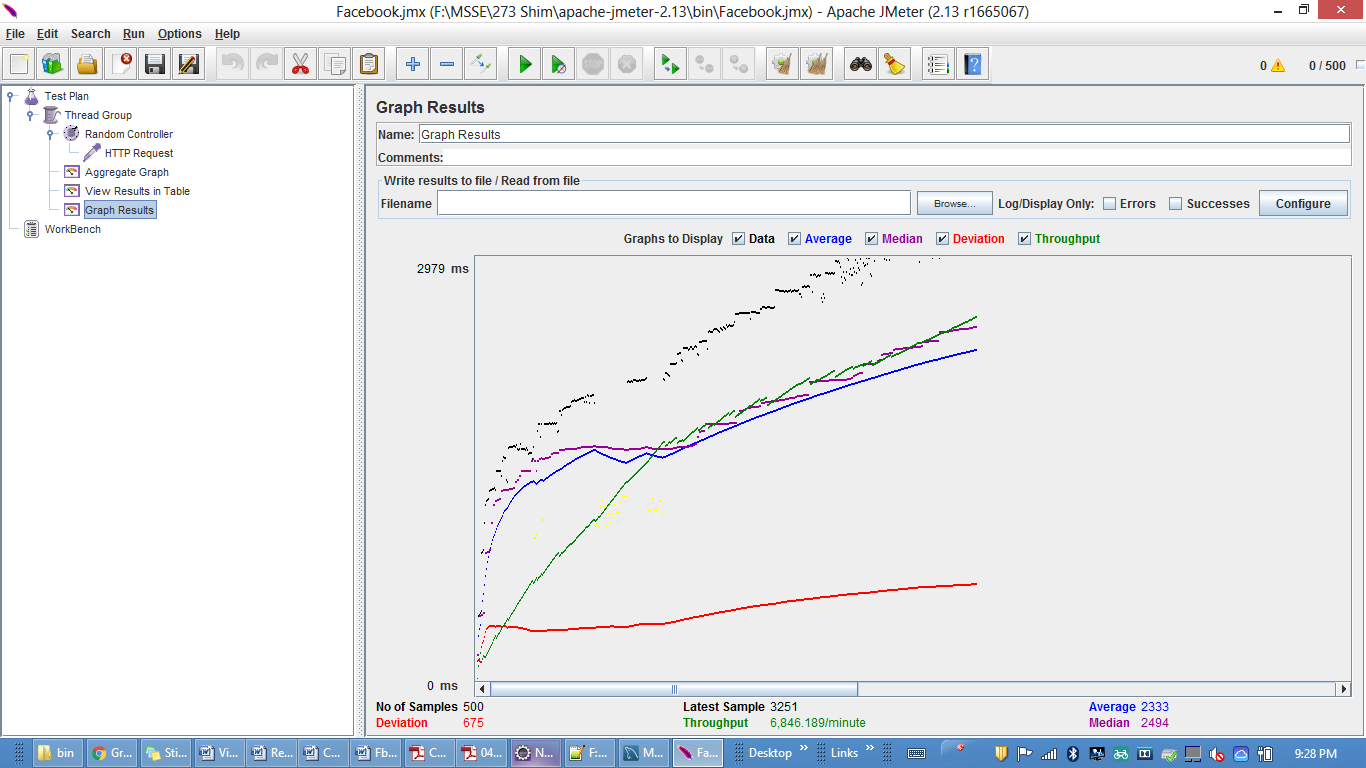
**Test 3: 300 concurrent users**

****

**Test 4: 400 concurrent users**

****

**Test 5: 500 concurrent users**

****

**Comparison:**

Using Connection pooling the response time decreases by a huge amount and hence increasing the performance of the system.

**Question & Answers:**

1.Describe your algorithm and strategy for data caching. Discuss why you choose the algorithm/strategy.

Answer:

1. Data caching is very useful in an application because it improves performances, it’s readily available.
2. Since data is stored into a Json variable when the server starts it gives the increased performance whenever the user logs in again.
3. Records from the user table will be stored in the Nodes of the Doubly Linked List at the front end of the queue. When the object is referenced and is not present in the Doubly Linked List of a queue, it will be added to the front end of the queue.
4. The frequency of the user is used to store the data into queue.

2. Compare the performance characteristics (draw graphs) of the servers without connection pooling and with connection pooling. Repeat the same with data caching. Use a graph or table to chart your results visually. Explain why you see such results.

Answer:Using connection pooling the performance of the system is improved a lot.Also if I have used caching the performance would have increased a lot.

3. Considering data caching, how you would implement Request Caching strategy. Explain in detail. You do not need to implement this new function – pseudo code or a detailed description will be fine.

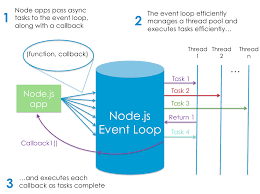
Answer: Whenever that resource is required multiple times, its retrieval would be fast using caching. Conditional request by client can be used to check if there is any updated copy of the local resource .Using Request Caching Strategy, web browsers would store the local copies of the web resources. Cache headers are added to the resource response to specifying its behavior. So if the resource requested is fully cached it could be retrieved from its local copy rather than sending request to the server. Cache header in the response would control the response strategy as it enable caching in the browser. Server would send the response whether the browser copy is latest or not. This would be done by checking the content of the resource sent in the request to server. If the browser copy is latest, a blank response is sent by the server- saving the cost of transferring resource back to the client. Content based and Time based are the two techniques can be used to identify whether the browser copy is latest or not.

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

Answer: Yes, I have implemented Horizontal scaling using Clustering in Node.js. Clustering in node.js allows several separate processes to share same server port. Several processes can be created by using different core of processor. So different servers are running on the same port. No of servers connected are equivalent to no of core in the processor. If one of the server goes down, then it can be replaced by other server. Node.js also serves as internal load balancer for these servers.

5. Explain how node.js handles multiple client requests simultaneously even though it is single threaded. Explain using block diagram.

Answer:



When a client sends http requests to node.js server, these requests are placed on the event queue. Non-blocking requests are handled by the main event loop and response or result is sent back to the client. Blocking requests are passed to the non-blocking worker which is called as thread pool to handle long requests like I/O request. Event loop doesn’t sleep and continues to handle other non-blocking requests. Responses of long requests are sent to the main event loop via callback functions which is further sent to the client. This is how node.js handles multiple client requests even though it is single threaded.