Name: Madhav Vij

Please read this:

I understand that I am on my honor during this quiz, I will not collaborate,

use non-allowed sources, and I will not discuss this quiz with anyone for

the next 4 hours. In submitting this quiz, I agree to this honor code.

Please initial here:\_\_\_\_\_\_\_\_\_\_M.V\_\_\_\_\_\_\_\_\_\_

C&P means copy and paste only those relevant lines from your program(s) into this quiz.

You may include lines of code, HTML, CSS, or any "textual" artifact (no libraries, executables, utilities,

pictures, diagrams, songs, or video, please)

NOTE: Please submit on time, late submissions will have points deducted!

Please submit to Blackboard - IF unavailable, email to class account.

Due to technical logistics, we can not look at any demo (the "show us" part) in the last 5 minutes of the quiz.

The icon "--->" means that you should place an answer at this location in the quiz, either textual, pieces of code

(or HTML,XML,CSS or similar), or numeric results.

(Where appropriate, for example meter settings and results, please include screenshots)

1. Good luck!

2. Get all files from this same folder.

3. Rename this quiz and your programs with your name and last digits of your ID.

4. For each of the following parts, please implement, cut and paste the relevant code (no binaries,

libraries, etc, just source code and any textual configuration files, html, and similar) into this file.

If possible, show each part (with an output) to a GTA, when complete.

Submit this quiz (to Blackboard) when complete, you may make multiple submissions.

5. You do not need to implement error handling, unless needed for this quiz, (file already exists, empty files, etc)

but your program should not abnormally terminate (crash).

6. + All the following implementation should be done on AWS.

+ All functionality should use a browser and web pages (or dynamically created content),

+ and EACH page should have:

your name, class, and ID number at the top.

CODE:

[FILE: index.html]

<div class="container">  
 <div class="media-left">  
 <img src="{{ url\_for('static', filename = 'image.jpg') }}" class="media-object img-circle" height="150px" width="150px"></div>  
 <div class="media-body">  
 <h1><b><center>Madhav Vij: 1001440170; class: Advance DB</center></b></h1>  
 </div>  
</div>

[FILE: quiz7.py]  
  
@app.route('/')  
**def main**():  
 #createDB()  
 **return** render\_template('index.html')  
  
  
port = os.getenv('PORT', '80')  
**if** \_\_name\_\_ == '\_\_main\_\_':  
 app.run(host='0.0.0.0', port=int(port))

7. + Upload a picture of you Or the class picture "class.jpg" (anywhere on AWS) and then create a web page that shows part 6 and that picture.

Then upload the "Classes.csv" file and create a SQL table. We would like to query this table on either course number or instructor name.

You may do this setup manually or through a web interface.

+ There is no "user" concept, no login, all people using this service are "users".

+ Create a web interface that allows a (any) user to query the table (by course number or instructor) and show the results

on the web page (there may be a few matching tuples). Show the (AWS) time and the time it takes to respond to the request.

--> Show us. 15 points

--> Put code here. 20 points

CODE

[FILE: index.html]  
<!DOCTYPE html>  
<html>  
 <head>  
 <meta charset="utf-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <meta name="description" content="">  
 <meta name="author" content="">  
 <!-- Latest compiled and minified CSS -->  
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">  
  
 <!-- Optional theme -->  
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp" crossorigin="anonymous">  
  
  
  
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>  
 <title>Quiz 7</title>  
  
 </head>  
 <body>  
 <div class="container">  
 <div class="media-left">  
 <img src="{{ url\_for('static', filename = 'image.jpg') }}" class="media-object img-circle" height="150px" width="150px"></div>  
 <div class="media-body">  
 <h1><b><center>Madhav Vij: 1001440170; class: Advance DB</center></b></h1>  
 </div>  
 </div>  
  
 <div class="container" >  
 <div class="well">  
 <div class="form-group">  
 <div class="page-header">  
 <h2>Details:</h2>  
 </div>  
 <form action="getAll" method="post">  
 <label for="instructor">Instructor: </label> <input class="form-control" type="text" name="name"  
 placeholder="Eg: Smith" /><br/>  
 <label for="instructor">Course# : </label> <input class="form-control" type="text" name="course"  
 placeholder="Eg: 5351" /><br/>  
 <input class="btn btn-success btn-block btn-lg" type="submit" value="query">  
 </form>  
 </div>  
 </div>  
 </div>  
  
 </body>  
</html>

[FILE: quiz7.py]  
**import** os  
  
**import** hashlib  
**import** time  
**import** memcache  
**import** pymysql  
**from** flask **import** Flask, render\_template, request  
  
app = Flask(\_\_name\_\_)  
memC = memcache.Client(['memcache6331.zz49ah.cfg.use2.cache.amazonaws.com:11211'], debug=0)  
  
  
**def connectDB**():  
 **return** pymysql.connect(host='madhavrds.cwnoqs3zlcce.us-east-2.rds.amazonaws.com', port=3306, user='madhavvij',  
 password='system123#', db='cloud6331', local\_infile=**True**)  
  
**def createDB**():  
 conn = connectDB()  
 cur = conn.cursor()  
 cur.execute("""DROP TABLE IF EXISTS data""")  
 conn.commit()  
 query = """ CREATE TABLE Classes (  
 `Branch` **VARCHAR**(4) CHARACTER SET utf8,  
 `Course` **INT**,  
 `Section` **INT**,  
 `Course\_Title` **VARCHAR**(31) CHARACTER SET utf8,  
 `Instructor` **VARCHAR**(12) CHARACTER SET utf8,  
 `Day\_s` **VARCHAR**(4) CHARACTER SET utf8,  
 `Start\_time` **VARCHAR**(8) CHARACTER SET utf8,  
 `End\_Time` **VARCHAR**(8) CHARACTER SET utf8,  
 `Max` **INT**,  
 `Enrolled` **INT**  
 ); """  
  
 cur.execute(query)  
 conn.commit()  
 query = """ LOAD DATA LOCAL INFILE '/home/ubuntu/quiz7/input/Classes.csv' INTO TABLE  
 Classes FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' ESCAPED  
 BY '"' Lines terminated by '\n' IGNORE 1 LINES; """  
 cur.execute(query)  
 conn.commit()  
 cur.close()  
 conn.close()  
  
  
**def fromDB**(sql):  
 conn = connectDB()  
 cur = conn.cursor()  
 cur.execute(sql)  
 data = cur.fetchall()  
 conn.commit()  
 cur.close()  
 conn.close()  
 **return** data  
  
  
**def fromMemcache**(sql):  
 conn = connectDB()  
 cur = conn.cursor()  
  
 hash = hashlib.sha256(sql).hexdigest()  
 # print(hash)  
 key = 'cache:' + hash  
 # print("Key= ")  
 # print(key)  
  
 **if** memC.get(key):  
 # print("used memcache")  
 **return** memC.get(key)  
  
 **else**:  
 # print("add to memcache")  
 cur.execute(sql)  
 data = cur.fetchall()  
 conn.commit()  
 cur.close()  
 conn.close()  
 memC.set(key, data, time=500)  
  
 **return** memC.get(key)  
  
  
@app.route('/getAll', methods=['POST','GET'])  
**def exampleDB**():  
 start\_time = time.time()  
 result = 0  
 output = []  
 **if** request.method == 'POST':  
 #instructor = request.args.get('name', '')  
 instructor = 'xxxxxx'  
 course = '0000'  
  
 **if** request.form['name']:  
 instructor = request.form['name']  
 **if** request.form['course']:  
 course = request.form['course']  
 query = 'select \* from Classes WHERE Instructor like "%'+instructor+'%" or Course = "'+course+'"'  
 result = fromMemcache(query)  
 **for** row **in** result:  
 tuple = (row[0],row[1],row[2],row[3],row[4],row[5],row[6],row[7],row[8],row[9])  
 output.append(tuple)  
 end\_time = time.time()  
 total\_time = end\_time-start\_time  
 print(total\_time)  
 output.append(total\_time)  
 **return** render\_template('display.html', output=output)  
  
  
  
  
@app.route('/')  
**def main**():  
 #createDB()  
 **return** render\_template('index.html')  
  
  
port = os.getenv('PORT', '80')  
**if** \_\_name\_\_ == '\_\_main\_\_':  
 app.run(host='0.0.0.0', port=int(port))  
  
#####################################################################

[FILE: Display.html]  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
<div class="container">  
 <div class="page-header"><center><h1>Data From Table</h1></center><br/>  
 <center><h1>Time Taken: {{ output[-1] }}</h1></center>  
  
 </div>  
<table>  
{% for outputList **in** output[:-1] %}  
 <tr>  
 {% for result **in** outputList %}  
 <td>{{ result }}</td>  
  
 {% endfor %}  
 </tr>  
  
  
{% endfor %}  
</table>  
</body>  
</html>

8. + using part 7, using jmeter, or similar, show the performance of your application.

We (assessors) will the number of users, which course (or instructor), and any other necessary parameters.

+ When we request, create sufficient traffic to require creation of additional (at least one additional) instances.

+ We wish to guarantee a QoS of at most 1 second from site/page request until that page is displayed.

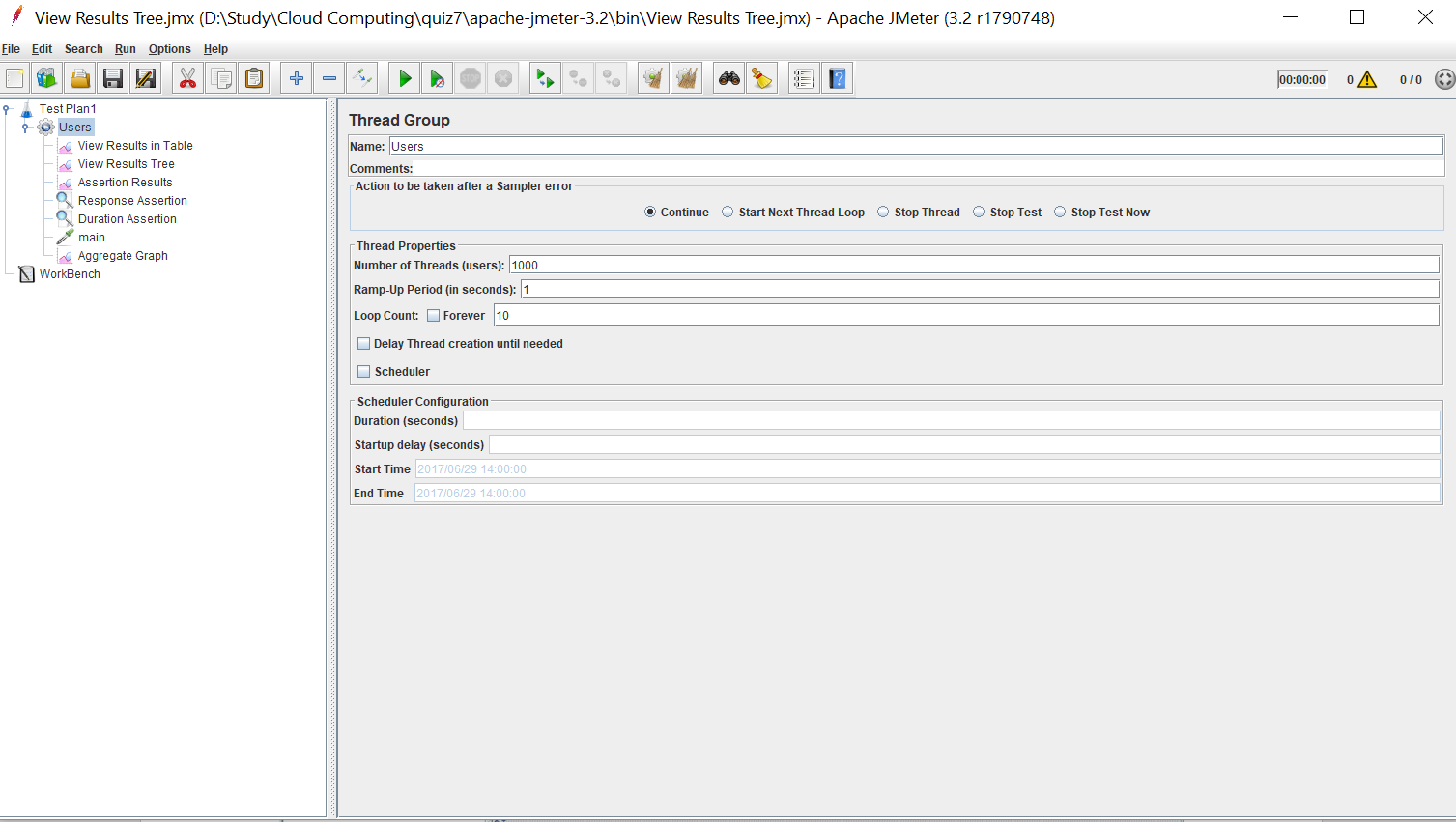
Show how we can determine if that goal is met. (You do NOT need to meet that goal, just show whether you do or don't)

(Either show us a method, and include screen shots, OR, code, or clearly explain how we know whether you meet that goal.)

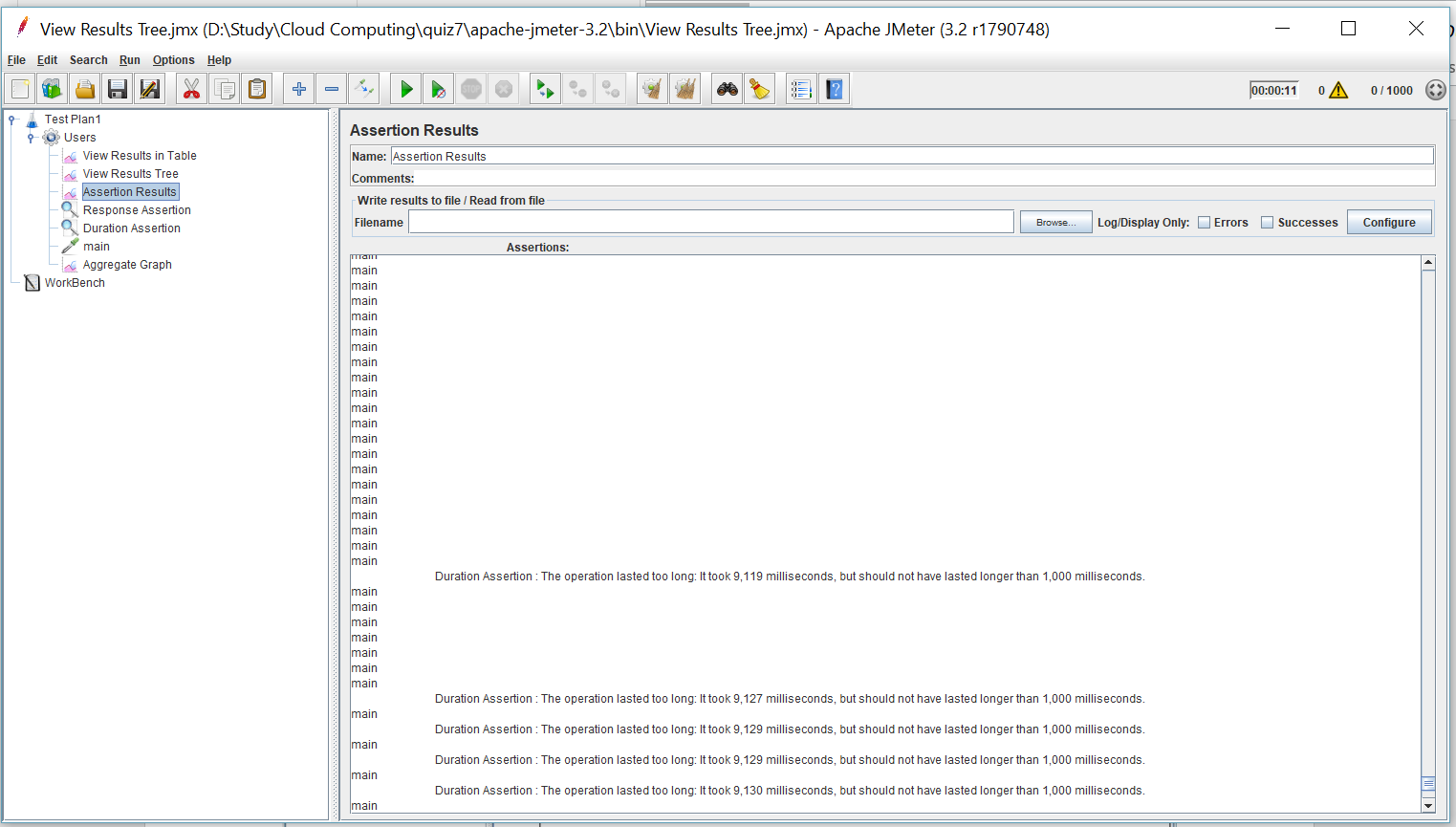
+ Give one method (explain or give code) that we could show whether (or not) we meet a goal of 95% availability for a 24 hour period, starting now.

--> Show us. 35 points

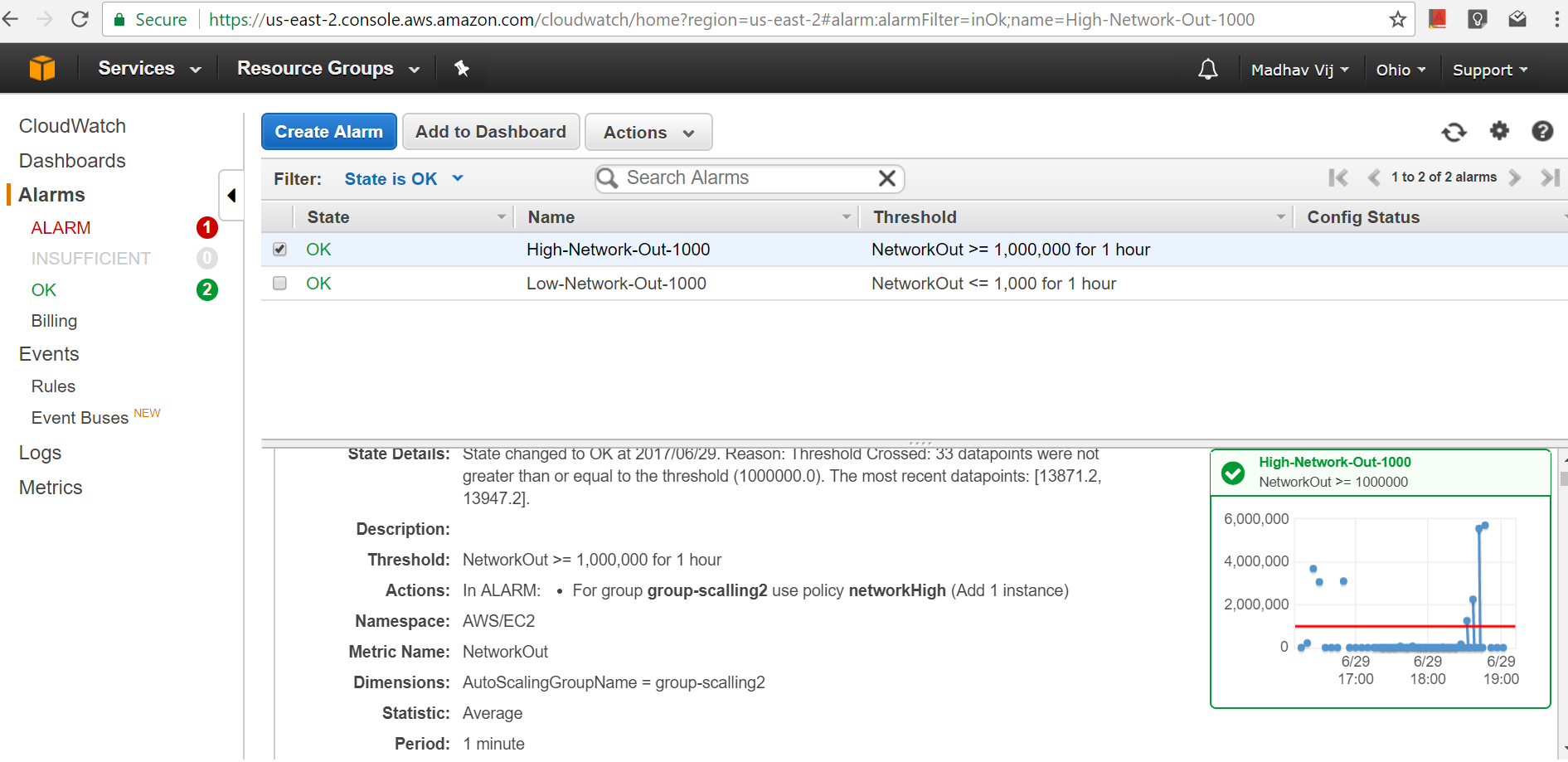
--> Put code here. 30 points



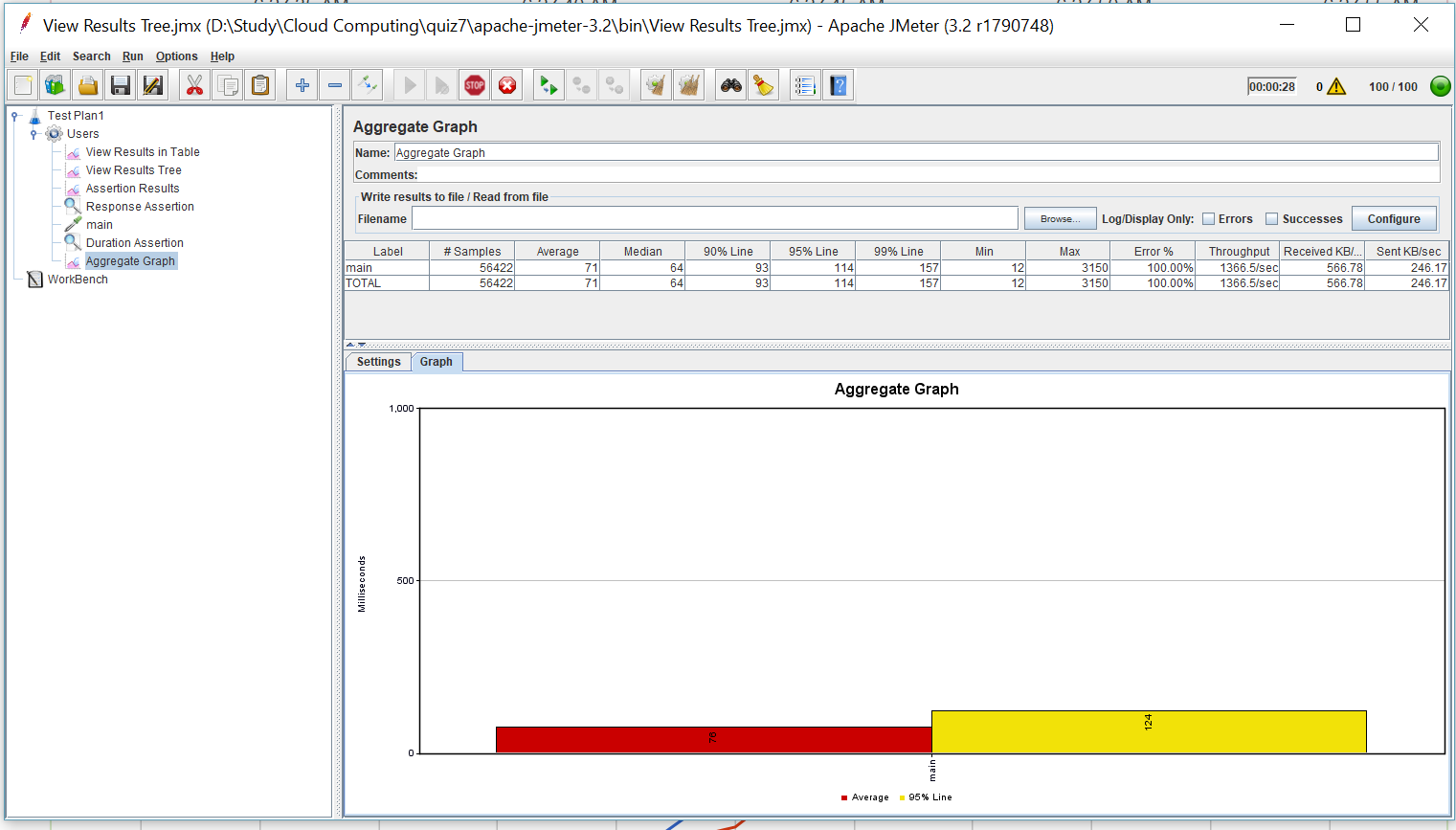
Assessor provided the above parameters, and test results run fine.



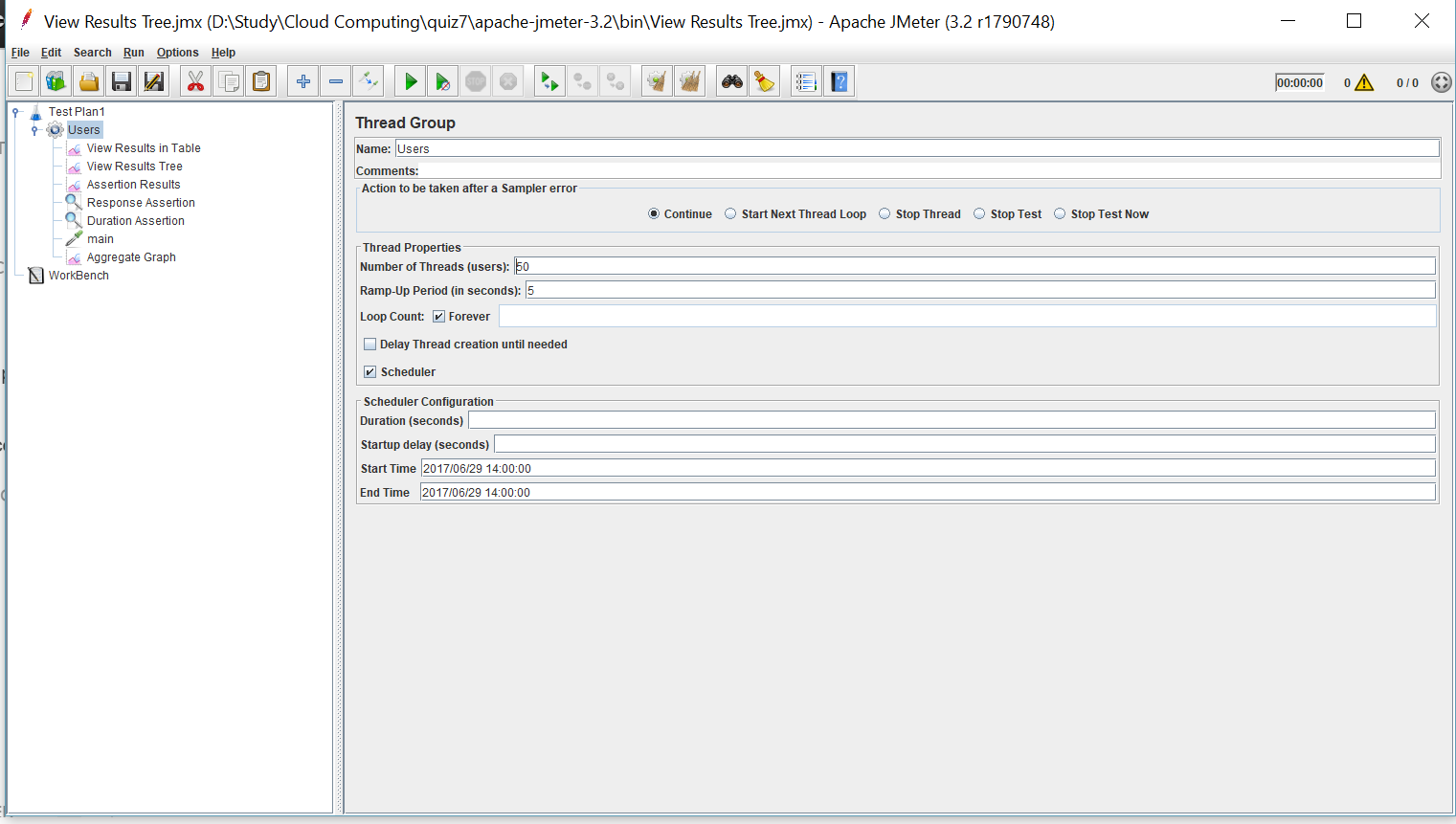
When duration assertion was set to 1000 mili-second (1 second) some responses failed (took longer ~9 seconds)



Network threshold set to 1000000 Bytes (Auto-scaled up, when run tests from JMeter)  
number of instances increased from 1 to 3, in next 5 minutes.

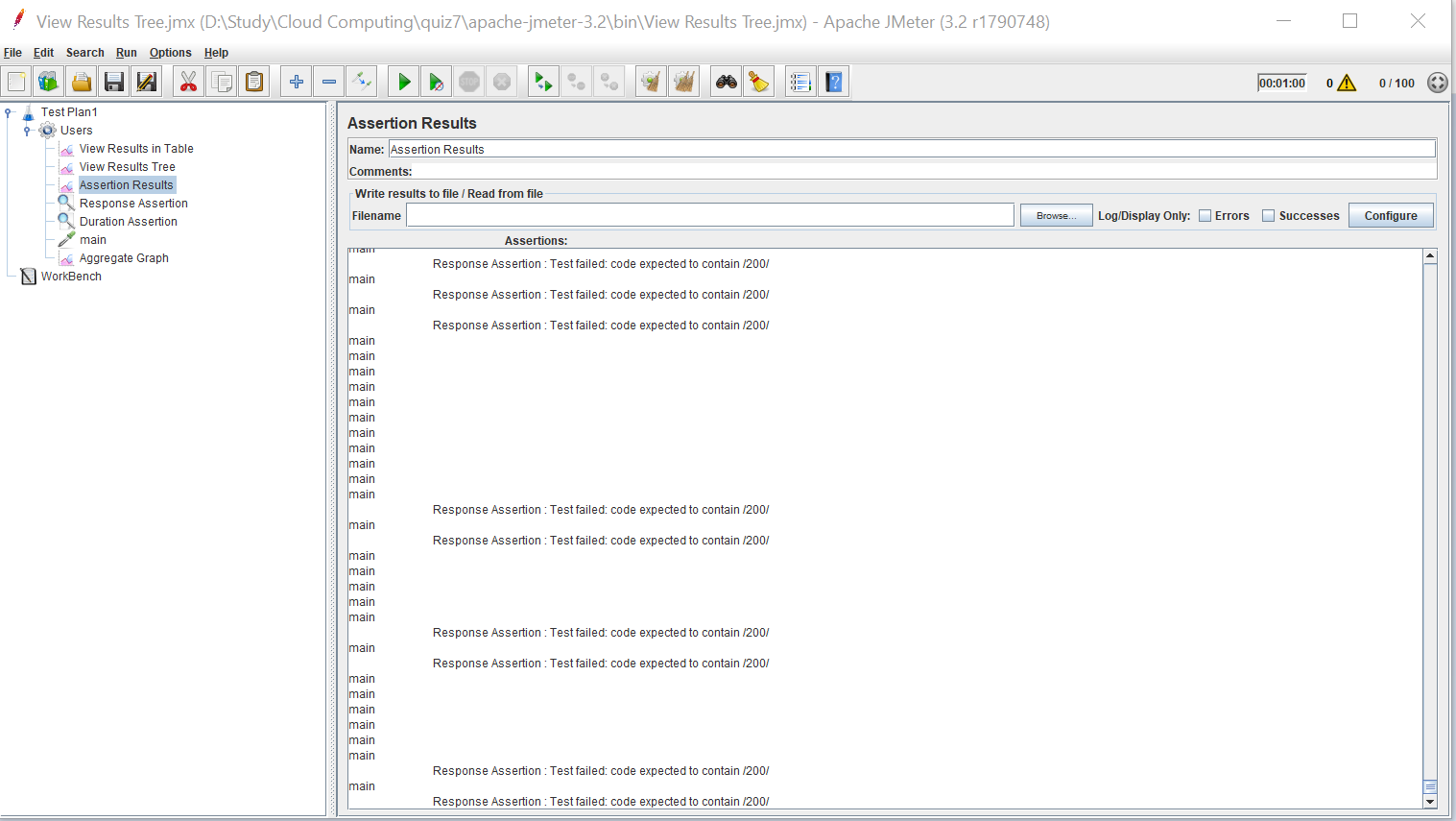


95% availability test for 24 hours, test failed (Red bar is the result, yellow is the threshold)



(Parameters for 95% availability)

Method used: add a schedule to run for 24 hours (or required time) where, it goes over the threshold limit and the loop count being forever. Generate the Assertion Graph over responses.  
This will produce above graph.



Some responses do not send 200 response code when the server gets loaded with 100 Threshold users  
every second for 10 times.

Done. When complete, return (send) this quiz

If you finish early, send this immediately, otherwise send on-time.