TLC – TP1,2,3

A. Lab Description

Task 0 Preparation (Optional): Deploy a Guestbook Web Application

As preparation for the rest of the coursework, this exercise covers the use of the Google App Engine to develop and deploy a simple web application. You will need to follow the on-line tutorial of the Google App Engine to complete this task. The Google App Engine currently supports Java and Python. You can choose either of these two programming languages for this exercise. Google provides a step-by-step tutorial to deploy the Guestbook application. You can find the tutorial at: http://code.google.com/appengine/

By the end of this task, you should upload your Guestbook application to the Google App Engine. This application should be visible to the public from the URL http://your-application.appspot.com/, where "your-application" is the name of the application you provide to the Google App Engine. You can create a maximum of 10 applications for free.

HINTS: Section B summarises some potential issues that you might encounter while deploying the Guestbook application.

Task 1 Develop a Simple Advertisement Board

As the main task of this lab, you are asked to develop a web application that implements an online "advertisement board", which should be deployed on the Google App Engine. This advertisement board should provide three main functionalities.

Bulk Add Your advertisement board should allow user to add a list of advertisements. Each advertisement should include a title (e.g. a 2 "bedroom flat to let"), a price, and the date when the small add was added. Users of your advertisement board application should be able to enter the list of advertisements on a webpage and click an "Add" button to add these advertisements to the data store.

Search Your web application should provide a search function for the advertisements in the data store. Users of your application should be able to query for advertisements that matches criteria such as keywords, the range of prices, and the date range. The search results can be displayed in a table. The users should be able to search for advertisement based on any combination of keywords, price range, and date range.

Bulk Delete The users of your web application should be able to delete all the advertisements that are returned from a search.

HINT:

- You can use textarea in an html form for users to enter a list of advertisements.

<textarea name="content" cols=40 rows=6></ textarea>

- Your are free to use either Java or Python. If you decide to use Java (and JS), you will find plenty online tutorials about JSP and Servlet. The following lists the URLs of a few online tutorials. They are simple, but cover all the features that you will need for this coursework.

http://www.jsptut.com/ A JSP tutorial that covers the key features you need for this coursework, including the use of Java to present dynamic web content; the use of a session to carry information from one webpage to another; form submission, etc.

http://docs.oracle.com/javaee/6/tutorial/doc/bnafu.html Oracle's online tutorial on Servlets.

Task 2 Measure the Scalability of Google App Engine

In this task, you are asked to investigate the scalability of the Google App Engine (as far as you are able to under the free quotas imposed by GAE). To do so, you will need to submit http requests to your advertisement board application to perform bulk add, search, and bulk delete functions that you have implemented in Task 2. You then need to measure and analyse the round-trip latency for receiving responses from the advertisement board application as the number of http requests per time unit increases, and as the advertisement data size increases.

NOTE: Latencies and throughputs are likely to vary between each individual request. You should therefore take appropriate steps to derive average values and examine the spread of the performance data under comparable conditions (request frequency, size of the data store, request type).

HINT: There are many programming languages that you can use to send http requests to a web application. The following link presents the Java approach for achieving this aim.

http://download.oracle.com/javase/tutorial/networking/urls/readingURL.html

In this simple example, the http request is sent to a static http address. In your experiments, your http requests need to carry input data for bulk add (i.e. the advertisements that need to be added to the data store) and searching (i.e. criteria such as keywords, price range, and date range provided by users). When you submit an http request via the URL address, you can carry attributes as part of the URL address. For instance, if you want to send a request to the search.jsp page to search for a flat that has a minimal price of 500 pounds, you can submit the following URL:

http://your-application.appspot.com/search.jsp?keyword=flat&minPrice=500

To record the round trip latency of http requests, you can record the start time that is just before your program sends an http request and the end time immediately after your program receives the response from the web server. In Java, you can call the following code to obtain the current time in milliseconds.

```
long currentTime = System.currentTimeMillis();
```

Deliverables

Based on your work in Task 1 and 2, write **a short report of 1,500 words** highlighting the results of your investigation an making recommendations to a client as to whether the Google App Engine is a suitable technology for very large scale applications for advertising. Your discussion is not restricted to the scalability of the Google App Engine. For instance, you might report the limitations on the current programming interfaces, discuss why such limitations can prevent the deployment of real applications in the Google App Engine, or suggest further improvements to the Google App Engine platform.

Please submit your report along with **your code** and **experimental data** electronically through the **moddle** web site by the deadline indicated on the module's page. Also, be sure to include **the URL at which your GAE application is available**.

Marking Scheme:

Correctness / quality of application (on-line):	5 points (25%)
Richness of the features your application offers:	5 points (25%)
Report:	10 points (50%)
- Overall presentation (clarity, structure):	1 point (5%)
- Design and its motivation:	3 point (15%)
- Quality of evaluation:	2 point (10%)
- Depth of analysis:	4 point (20%)
Total:	**20 points (100%)**

B. Potential Issues

Use of JSP in Google App Engine

To use JSP in the Google App Engine, you need to include the JDK in your build path instead of JRE. Otherwise you will have a compilation error. If you use Python or Java Servlet for this exercise, you should not encounter this problem.

If you use eclipse, you can go to Window -> Preferences -> Java -> Installed JREs, and add the folder that contains your JDK to installed JREs. You can then remove the JRE library from the build path of your project, and then add JDK instead.

Create Google App Account

You need a Google App account in order to upload your web application and make it visible to the public. You can create a new Google account from the following link. You can login if you already have a Google account (e.g. Gmail account).

http://code.google.com/appengine/

To activate your account, you will need to provide your mobile number and receive the activation code by SMS text message. If you have any difficulty, please contact the lecturer of this course.

Finally, you need to provide a name for each of your application. This name will be used as part of the URL of your web application. For instance, your web application will be hosted at http://your-application.appspot.com, where your-application is the name of your web application.