CSI3344 Distributed Systems

Assignment 2: A Report

(You need to choose one topic from the five options)

General Information:

- This assignment is a team based report for up to two students (depending on the topic you selected, the report may be a group one or an individual one, however). It consists of two parts therefore requires two submissions.
- The first part requests you complete a report based on a research into the topic (individual work), or a mini programming project (team work for up to two students), that you selected from a number of choices (see the listed topic options in the following pages).
- The second part requests you give a seminar/presentation to the class, highlighting the main points (i.e., the main findings) or outcomes of your research/project. The presentation should not exceed the 15-minute time limit. Additional question time will be given after each presentation. The presentation is scheduled in the workshop sessions of the last two teaching weeks (see unit schedule). Please make sure you are ready for your seminar presentation by 10:30 on Monday 16 May 2016.

Due date: Submission 1: Seminar resource due on Friday @5:00 pm of Week 11

Submission 2: Seminar presentation PPT slides by Friday @5:00 pm of Week 11

Weight: 30% of unit mark

(GO TO NEXT PAGE)

Submission Requirements:

- The first submission (for seminar resource) includes a report and any additional documentation associated with the report. The second submission (for seminar presentation) includes PPT slides, and some possible additional handouts. These should be submitted via Blackboard (note that a separate submission link will be created for you to submit the Presentation PPT slides). For detailed submission procedure, please refer to "How to submit your Assignment online" item in the Assignment section of your Blackboard.

- If you like to submit your seminar resource and presentation slides in one lot, you may do so on or before the first submission due date (in such a case, please let your tutor know it beforehand).

- The report file must be in Word or PDF format (see next page for detailed Report format requirement). If you submit multiple files, or if you take the mini project (thus need to include separate runnable code file/s), please zip your files into one file before submission. Please name your submission file in the format of <your_student_ID>_< your_full_name>_A2_CSI3344.zip. For example, if your student ID was 1234567 and your full name was Abcd Xyz, your submission file should be named 1234567_Abcd_Xyz_A2_CSI6111.xxx (where xxx is the extension of your submission file).

- NO hard copy submission is required.

- Note that files found to contains viruses or other infecting mechanisms shall be awarded a zero mark, as well submissions found contain any form of cheating/plagiarism.

- Your attention is drawn to the university rules governing cheating and referencing. In general, cheating is the inclusion of the unacknowledged work of another person. Plagiarism (presenting other people's work and ideas as your own) and cheating will result in a zero mark.

(GO TO NEXT PAGE)

Research report format requirement:

Report Must contain	**Cover/Title page** Must show assignment title, your student ID and name/s, due date and the title of your chosen topic
	Executive Summary This should represent a snapshot of the entire report that your tutor will browse through and should contain the most vital information requested. The Executive Summary should be in between Cover page and ToC.
	Table of Contents (ToC) This must accurately reflect the content of your report/project and should be generated automatically in Microsoft Word with clear page numbers.
	I**ntroduction** Provide background information of the topic/report; Introduce the report, define its scope and state any assumption; Use in-text citation /reference where appropriate.
	Main body The report should contain (but not limited to): • Understanding of concepts/topics involved in the report. • The problems/ questions being solved/answered • Significance of the research/report • Strategies used to solve the problem (e.g., provide a solution or an approach to develop a solution?) • Discussion or a critique of the solution developed/achieved, etc.
	Conclusion Outcomes or main finding/s from the research/project?
	References A list of end-text reference formatted according to the ECU requirements using the APA format. It is recommended that EndNote be used to organize/manage references, which should be ideally comprise of books, journal articles and conference papers (A few Web references are OK but they should account for a small part of the whole references).
Other requirement	The report should be no more than 3000 words (excluding references and diagrams) for individual assignment, and be no more than 5000 words for team work, and be named as per Submission requirement, and should be in a single file unless you chose option 4, in which case, a zip file. The text must use font **Times New Roman** and **be not smaller than 12pt**.

Option 1 (for those who are good at comparison of technologies)

SURVEY REPORT: A COMPARISON AMONG *DISTRIBUTED COMPUTING*, *GRID COMPUTING* AND *CLOUD COMPUTING*

Background Information

Distributed computing (DC) refers to a collection of hardware and software systems that contain more than one processing or storage element but appearing as a single coherent system running under a loosely or tightly controlled regime. A key feature of such a system is that the computers/devices in the system do not share a memory but they coordinate their work/s by passing messages asynchronously or synchronously. *Grid computing (GC)* is a form distributed system whereas *cloud computing (CC)* is a style of grid computing. While the DC, GC and CC share similar key features as systems, they can be quite different from each other. For example, some people simply view cloud computing as an environment in a user's machine that uses the internet to access someone else's software running on someone else's hardware in someone else's data center.

You are requested to conduct a survey on *cloud computing* (i.e., a survey primarily on cloud architecture/s, modes of clouds, key features/properties and advantages and disadvantages of using CC, etc.), and make comparison among DC, GC and CC in terms of similarity/difference, scalability, benefit and cost, security/privacy, and so on. Your view on how to move cloud computing to its full potential is also requested.

Based on your research, write a report presenting your research outcomes. Your report should not be longer than 15 A4 pages, including text, figures and tables; Figures/diagrams/tables should be with captions and numbered and cited in the text; the references should be denoted in text and fully listed in 'References'.

END of Option 1

(GO TO Next Page)

Topic option 2

A report on:

The Development History of Distributed System

The concept and the practice of *distributed system (DS)* has evolved along its long development history in the areas of network and data distribution. For example, the first widespread DS was the so-called *local-area* network, such as Ethernet, which was invented in the 1970s. After a few decades of development, a DS can now be defined as *a collection of independent devices that appears to its users as a single coherent system, in which networked components communicate and coordinate their actions only by passing messages*. Some significant characteristics of DSs include concurrency of components, lack of global clock, and independent failure of components.

When studying the DS development history you would never miss the terminologies that have historically contributed to the DS development, from *local-area network, internet, email, distributed database, distributed computing*, all the way to *cloud computing*. Similarly, when tracking DS's technical terminologies, you may never miss keywords such as RPC, RMI, communication protocol, distributed naming services, middleware services, synchronization, concurrency control, distributed replication, fault tolerance, and DS security, and so on.

Based on the above keywords (or concepts or terminologies), you are requested to conduct a research on the topic of *history of DS development*. You should use (but not limited to) the abovementioned keywords to guide your research.

Based on your research, write a report presenting your research outcomes. You are recommended to write the key DS development events in a chronological way. A comparison between the generic distributed system and the most recent development of distributed computing system, e.g., cloud computing system, is also required.

END of option 2

Topic option 3

A research on:

TECHNOLOGIES APPLICABLE TO BUILD BOTH DISTRIBUTED DATABASES AND DISTRIBUTED SYTEMS

A *distributed database* (DD) is a collection of multiple logically interrelated databases distributed over a computer network. A *distributed database management system* is a software system that manages a distributed database while making the distribution transparent to the users. While a distributed system (DS) was sometimes viewed as *internet + distributed database,* the technologies applied in distributed systems are much more complicated than those of distributed databases. On the other hand, many IT professionals agreed that distributed database technologies, such as distributed transaction processing, data replication, fragmentation, etc., have contributed a lot and been used in today's distributed system development.

You are required to prepare a report on the technologies that were initially developed for distributed database systems, and are then adopted (or/and shared) by distributed systems. List a set of technologies that have been used in both areas of DD and DS. Then take two (or more) technologies that have been used in both areas of DD and DS as examples and address the main differences when they are employed in building a DD and DS.

You may start with the development history of distributed database and summarize the key technologies applicable to both DD and DS.

Other requirements:

- The events stated in your report should be important, relevant, and chronological;
    - The evidence and data used for supporting your arguments should be solid and referenced;
    - The conclusions should be coincident with your evidence, data, and analysis;
    - Figures/diagrams/tables should be with captions and numbered and cited in the text;

END of option 3

Topic option 4

A Research on

MIDDLEWARE FOR DISTRIBUTED SYSTEMS: ITS PAST, PRESENT, AND FUTURE

In the early days, people saw no need for middleware in distributed systems. With the evolution of distributed systems from handling relatively simple and specific tasks in a homogenous network to dealing with complex and various activities in a vast heterogeneous environment, it is middleware that makes modern distributed systems perform effectively and more efficiently. The evolution should continue with the rapid increase in networked activities in peoples' daily life, which will drive not only the further expansion of middleware in its current territory, but also the middleware technology to some new domains in the future.

You are required to prepare a report on the evolution of middleware in terms of its ***models*** and ***services*** to the development of distributed systems in the past, present, and future.

Other requirements:

- The events stated in your report should be important, relevant, and chronological;
- The evidence and data used for supporting your arguments should be solid and referenced;
- The predications should be based on logical induction and/or statistics;
- The conclusions should be coincident with your evidence, data, and analysis;
- Figures/diagrams/tables should be with captions and numbered and cited in the text;

END of option 4

Topic option 5 (for those who are good at programming):

A Mini RMI programming project

Background Information:

Many distributed systems have been based on explicit message exchange between processes. With **remote procedure call (RPC)**, a client application component can effectively send a request to another application component (at the server side) by doing a local procedural call, which results in the request being packaged as a message and sent to the callee (at the server side). Likewise, the result will be returned to the client application as the result of the procedure call.

As the popularity of object technology increased, techniques developed to allow calls to remote objects, leading to what is known as **remote method innovation (RMI)**. An RMI is essentially the same as an RPC, except that operates on objects instead of applications.

A client-server interaction can be principally implemented by using a RPC/RMI. If the interaction was between applications residing in two machines, it is conceptually named "two-tier" interaction. If the interaction involved applications residing in three or more machines, it is sometimes named "three-tier" or "multi-tier" interaction.

Tasks:

Note: *You may choose to do this task individually (i.e., to implement "two-tier" interaction); or as a team of two students (i.e., to implement "three-tier" interaction*)

Refer to the source code in the Appendix that defines a class *Average*. It is an application that runs on a local machine. It accepts a number of integers as marks to some units; then adds them up, and finally returns the average as double type. The result is displayed in a message box.

Individual Task (*Task-A*, to be completed by individual student):

Your task is to make a remote application, *Evaluator*, which is used to assess if a student is qualified for an Honours study based on the record of his/her Bachelor's course. The basic requirements of this application include:

1. Only authenticated users are allowed to use this application.
2. Client application should not only allow users to type in their ID and marks of individual units one by one through keyboard, but also provide functions to display the evaluation results received from the remote server.
3. The number of input integers (marks) should be at least 12 (meaning counted 3 semester advanced standing from somewhere else) and at most 30 including the failed mark and repeated mark of the same unit (meaning you are automatically disqualified if you failed more than 6 units during the course).
4. Server application should provide operations that process the input data, calculate the averages, assess the qualifications against the evaluation criteria, and inform the client the evaluation results.
5. The basic operations on the server should include displaying individual marks on the server in their input order, selecting the best 8 marks, calculating the course average, calculating the average of best 8 marks, evaluating the qualification, and sending the 'correct' evaluation result back to the client.
6. The evaluation criteria are described as follows:
   - If the course average is equal or greater than 70, return message "student ID, course average, QUALIFIED FOR HONOURS STUDY!";
   - If the course average is less than 70, but the average of best 8 marks is 80 or higher, return message "student ID, course average, best 8 average, may have a good chance! Need further assessment!";
   - If the course average is less than 70, but the average of best 8 marks is between 70 and 79, return message "student ID, course average, best 8 average, May have a chance! Must be carefully reassessed and get the coordinator's special permission!";
   - If the course average is less than 70, and the average of best 8 marks is also less than 70, return message "student ID, course average, best 8 average, does not qualified for honours study! Try Masters by course work."
7. The client and server-side applications should be able to run in deferent machines.

[Hints] You may need to use an array to store and select the input data on the server. You need to make the array empty when the results are returned to the client. (Why?)

Remember the steps to create a RMI application:

1. define the interface for the remote object
2. implement this remote interface
3. create stub and/or skeleton
4. implement the server software
5. implement the client software

Team-work Task (Task B: task for two students):

Your task consists of two parts:

(1) implement the *Task-A,* as descripted above (for simplicity, we name this server-side application *server-1*);
(2) create a *third-tiered* server, which holds/mimics a "tony" database server (called *server-2*). The server-1 and server-2 communicates by RMI as well. Additional functions are as below:
    a) Server-2 stores all data received from client application by way of server-1, e.g., whenever an users type in ID and marks of individual units (see step 2 in Task-A), server-1 also pass the data and save it on server 2;
    b) It allows a user check his/her unit results, say, by typing in ID; in such a case, server-1 validates data input and makes request from server-2, and then responses to the client request, etc.

Basic requirements

1. The applications should include all required components.
2. The observable behaviours of your application should be consistent with what is described.
3. The application provides necessary error handling mechanisms.
4. Provide a project report that is well structured and informative (no more than 15 pages)*, and separate Java code files of your project.

The project report format requirement

Your project report may take the similar format as that of the research report showing in page 3, except for the Main Body part, which should now include

i. (a brief) Introduction of RMI
ii. Application requirements
iii. Application design and implementation procedure
iv. Application setting-up and usage steps (screen shots)
v. An example showing snapshots of application running status and input/outputs
vi. Conclusion

Appendix: source code

The above Sections i) to vi) should be no longer than15 pages.

Appendix (for Option 5 only):

```
// Fig. 4.9: Average2.java from Deitel & Deitel <<Java How to Program>> (5th ed)
// Class average program with sentinel-controlled repetition.
import java.text.DecimalFormat;
import javax.swing.JOptionPane;
public class Average2 {
       public static void main( String args[] ) {
          int gradeCounter,  // number of grades entered
           gradeValue,    // grade value
           total;         // sum of grades
          double average;    // average of all grades
          String input;      // grade typed by user
          // Initialization phase
          total = 0;         // clear total
          gradeCounter = 0;  // prepare to loop
             // Processing phase
             // prompt for input and read grade from user
          input = JOptionPane.showInputDialog(
             "Enter Integer Grade, -1 to Quit:" );
          // convert grade from a String to an integer
          gradeValue = Integer.parseInt( input );
          while ( gradeValue != -1 ) {
             // add gradeValue to total
             total = total + gradeValue;
             // add 1 to gradeCounter
             gradeCounter = gradeCounter + 1;

             // prompt for input and read grade from user
```

```
                input = JOptionPane.showInputDialog(
                "Enter Integer Grade, -1 to Quit:" );
                // convert grade from a String to an integer
                gradeValue = Integer.parseInt( input );
                }

                // Termination phase
                DecimalFormat twoDigits = new DecimalFormat( "0.00" );
                if ( gradeCounter != 0 ) {
                average = (double) total / gradeCounter;
                // display average of exam grades
                JOptionPane.showMessageDialog( null,
                "Class average is " + twoDigits.format( average ),
                "Class Average", JOptionPane.INFORMATION_MESSAGE );
                }
                else
                JOptionPane.showMessageDialog( null,
                "No grades were entered", "Class Average",
                JOptionPane.INFORMATION_MESSAGE );
            System.exit( 0 );    // terminate application
        }  // end method main
}  // end class Average2
```

Reference:
Deitel and Deitel, Java How to Program (fifth ed), Prentice Hall, 2003.

END of option 5

END of Assignment Description