University of Glasgow | School of Computing Science

Assessed Coursework

Course Name	IS3			
Coursework Number	Assessed exercise part 2			
Deadline	Time: 4:30pm		Date:	5/12/13
% Contribution to final course mark	15%		This should take this many hours:	22.5 hours
Solo or Group ✓	Solo		Group	✓
Submission Instructions	Submit through Moodle.			
Marking Criteria	See below			
Please Note: This Coursework cannot be Re-Done				

IS3 2013-2014

Assessed Exercise Part 2 (AX2)

The aim of AX2 is to give you experience of creating a software tool that lets you find and convey interesting features in the same data as explored in AX1. You should use your AX1 exploratory data analysis, prototype visualisation system, and evaluation findings as a starting point as you move on to implement and then use your system.

You should continue working in the same groups as AX1. You can use any Java or Javascript toolkit that you wish (e.g. D3, Prefuse or Swing). You may also use use code components, widgets or similar elements written by other people. All this is perfectly acceptable as long as you give proper attribution, so that the source or authorship of each part of your system is clear. Note however that you must use either Java or Javascript in a 'pure' way, in that you cannot use unusual versions, or other languages or infrastructures (e.g. Python, MongoDB, MySQL, etc.)—so that you, in effect, demand that those marking your work install new software to make it run. (Note that Javascript + HTML/CSS is fine: they are standard enough so that markers need not install new software to make them run.)

Stage 1. Establish Your Design Requirements

AX1 asked you to do some basic exploratory data analysis, to consider data and design questions such as the following:

- What questions will you ask of the data?
- What charts and interactions will you use to provide insight?
- What significant patterns/trends exist in the data?
- What are you going to roll up?
- What are you going to drill down into?
- What aggregates and similar statistical combinations are you going to make, i.e. new measures?

In addition, you were asked to use Mackinlay's ranking of encodings as an aid as you made choices with regard to encodings and interactions for your prototype. The paper prototype, and its evaluation, should have given you a clearer idea of what to implement, and what analysis you will be able to perform as a result.

Document these ideas as *software design requirements*. As noted below, these requirements will form part of the report for AX2—which is worth marks. Note that you are not constrained to make your software prototype *exactly* match your paper prototype. For example, you may have decided upon useful design changes as a result of your evaluation. Also, as you begin to code, you may find that the language/toolkit you use may restrict what you can implement in the time available… or perhaps open up some new powerful idea that you can demonstrate as valuable. Lastly, you should note that marks will be given for quality and ingenuity of implementation, e.g. for skilful integration of analysis features, for functional implementation of sophisticated or novel algorithms, and for creative use of tools and techniques in order to support analysis. (A warning, though: please do not simply 'wedge in' all the complex functionality you can! Unless that functionality directly supports the free flow and analytic power of interaction, this approach will degrade the quality of your system… and so lead to lower marks.)

Stage 2. Implement Your System

Use iterative design and the requirements of stage 1 to *implement your system*. The details of how you do this are up to you, as a group, but… apply good software design practices! Decide who will make and do what early on, but be willing to adapt. Use version control (even if it's informal) and make frequent backups, integrate the parts created by different team members incrementally, test as you go, and document your design process (e.g. who did what, when he/she did it, and why he/she did it). Note that not everyone will be coding all the time.

In this way you can avoid some of the problems that caused major problems for students in IS3 (and IM2) before. For example, do not have all team members code independently and then try to assemble it all the night before the demo and handin. (Nothing worked. Very low marks.) Another example: do not have all the code on one person's laptop, in case a disc crash happens on the day of the demo, and all you have to show is some broken code from the week before. (No pity. Very low marks.)

Stage 3. Carry out your analysis

With your system implemented in accord with your requirements, you should now be able to *do your analysis and establish your findings*. You should, in effect, be developing your answers to your questions. This may include answers that you can include in your report, but also answers conveyed through dynamic interaction (that can be explained in your report). You may also use your analysis to expose new questions that are, in themselves, useful features to know about. Remember that the challenge here is to show interesting or significant features in the data, and conclusions from that data, as much as showing interesting or significant features of your system.

Marking Scheme, Submission and Demos

Submission will be done via the IS3 Moodle page. You should submit two files: your code and data, and your report.

The report (25 pages max) should summarise your design requirements (including how it stems from your paper prototype), a description of the submitted version of your system (and how it matches your requirements), and a description of your analytic findings (and how you derived them). Also include a one page description that introduces your system to a new user (i.e. to the marker), to help him/her run your code and try out analysis. Your report should make clear which parts of AX2 were done by which person— the requirements, system, analysis and write-up. Also note which features (such as code libraries you imported) were developed by people outside the group.

If your implementation language is Java then submit a project file for either Eclipse or NetBeans, so that a tutor can quickly load your work into an IDE, run it and inspect the code. Note that raw Java files for command line compilation/execution are not acceptable, and that it must be runnable in the standard SoCS environment. Your marker will assume a standard university environment, and will not install new compiler versions and the like in order to get your code to run.

If your implementation language is Javascript then submit a zip file with all your code and data, , so that a tutor can quickly load your work into a web browser, run it and inspect the code. You can assume that the tutor will run a basic local web server for this purpose, and will use the web browser and/or an editor such as Sublime Text to explore your code. Your marker will assume a standard university environment, and will not install new libraries and the like in order to get your code to run.

The maximum mark for this exercise is 30, to be scaled down later to match the 15% contribution of this exercise to your IS3 course mark. Assessment will involve six features, worth 5 marks each:

- Design requirements
- Code functioning according to requirements
- Code quality and ingenuity
- Analytic findings
- Report writing quality
- Demo quality

Demos will be arranged for the last sessions of IS3, in the week of December 2nd. More details of these (e.g. signup sheets for demo slots, the length of each slot) will be announced nearer the time.