Project 2017

Graph Theory

Due: 23/04/2017

The following document contains the instructions for Project 2017 for Graph Theory. You are required to design and prototype a Neo4j [2] database for use in a timetabling system for a third level institute like GMIT. The database should store information about student groups, classrooms, lecturers, and work hours – just like the currently used timetabling system at GMIT [1].

Minimum Viable Project

The minimum standard for this project is a GitHub repository containing a document detailing the design of the database and a prototype Neo4j database following that design and using data from GMIT.

The document should clearly state what data needs to be stored by a timetabling system, and how that data is to be represented in the database. This means that you must clearly specify the design choices you have made as to what data are represented as nodes, relationships, labels, relationship types or properties. It should also specify, and give examples of, Cypher queries for creating, retrieving, updating and deleting data.

The prototype database must contain data from GMIT's own timetabling system, in the structure set out in your design document. The data can be copied or scraped from GMIT's timetabling website, or otherwise obtained. You should include, as a separate section in your design document, how you obtained the data in your prototype database.

A better project will be well organised and contain detailed explanations. The architecture of the system will be well conceived, and example queries for various tasks, such as optimising timetables, will be provided.

Submissions

GitHub must be used to manage the development of the software. Your GitHub repository will form the main submission of the project. You must

submit the URL of your GitHub repository using the link on the course Moodle page before the deadline. You can do this at any time, as the last commit before the deadline will be used as your submission for this project.

Any submission that does not have a full and incremental git history with informative commit messages over the course of the project timeline will be accorded a proportionate mark. It is expected that your repository will have at least tens of commits, with each commit relating to a reasonably small unit of work. In the last week of term, or at any other time, you may be asked by the lecturer to explain the contents of your git repository. While it is encouraged that students will engage in peer learning, all documentation and software that is contained in your submission must have been written by you. You can show this by having a long incremental commit history and by being able to explain your code.

Marking scheme

This project will be worth 50% of your mark for this module. The following marking scheme will be used to mark the project out of 100%. Students should note, however, that in certain circumstances the examiner's overall impression of the project may influence marks in each individual component.

40%	**Documentation**	Clear explanation of the database design and usage.
40%	**Database**	Clean and intuitive database design, with data from GMIT.
20%	**Management**	Incremental work with informative commit messages.

Advice for students

- Your git log history should be extensive. A reasonable unit of work for a single commit is a small function, or a handful of comments, or a small change that fixes a bug. If you are well organised you will find it easier to determine the size of a reasonable commit, and it will show in your git history.
- Using information, code and data from outside sources is sometimes acceptable – so long as it is licensed to permit this, you clearly reference the source, and the overall project is substantially your own work. Using a source that does not meet these three conditions could jeopardise your mark.

- You must be able to explain your project during it, and after it. Bear this in mind when you are writing your README. If you had trouble understanding something in the first place, you will likely have trouble explaining it a couple of weeks later. Write a short explanation of it in your README, so that you can jog your memory later.

- Everyone is susceptible to procrastination and disorganisation. You are expected to be aware of this and take reasonable measures to avoid them. The best way to do this is to draw up an initial straight-forward project plan and keep it updated. You can show the examiner that you have done this in several ways. The easiest is to summarise the project plan in your README. Another way is to use a to-do list like GitHub Issues.

- Students have problems with projects from time to time. Some of these are unavoidable, such as external factors relating to family issues or illness. In such cases allowances can sometimes be made. Other problems are preventable, such as missing the submission deadline because you are having internet connectivity issues five minutes before it. Students should be able to show that up until an issue arose they had completed a reasonable and proportionate amount of work, and took reasonable steps to avoid preventable issues.

- Go easy on yourself – this is one project in one module. It will not define you or your life. A higher overall course mark should not be determined by a single project, but rather your performance in all your work in all your modules. Here you are just trying to demonstrate to yourself, to the examiners, and to prospective future employers, that you can take a reasonably straight-forward problem and solve it within a few weeks.

References

[1] GMIT. Galway mayo institute of technology - web timetables. http://timetable.gmit.ie/.

[2] Neo Technology Inc. Neo4j: The world's leading graph database. https://neo4j.com/.