ISYS2120 Assignment 3 (sem1, 2022)

Due: Sunday 23 October, 11:59pm Sydney time
Value: 15% [10% for the group as a whole, and 5% individual component]
This assignment is done in **groups of up to 5 students** (aim for exactly 4 members, but it may happen that sometimes a group is smaller or larger, eg if there are not enough students in a lab). We ask that all students in a group be attending the same lab session, so you can work together more easily, and show progress to the lab demonstrator each week.

Procedure: In week 10 lab, you should form a group. It is expected that most people will stay with the group they had joined in Asst12 but it is not necessary. There will be separate Canvas groups for this assessment; we will initialize them as copies of what were formed for asst2, but membership can change as described below. If any student wishes to not be with their asst2 team, they can simply remove themselves from the initial group; this should also be done if you were in a group from a class that you will not be attending in future. The lab demonstrator will then work with anyone who is unassigned, to form groups properly. If necessary, the demonstrator or coordinator may also rearrange group membership.
If, during the course of the assignment work, there is a dispute among group members that you can't resolve, or that will impact your group's capacity to complete the task well, you need to inform the unit coordinator, alan.fekete@sydney.edu.au. Make sure that your email names the group, and is explicit about the difficulty; also make sure this email is copied to all the members of the group. We need to know about problems in time to help fix them, so set early deadlines for group members, and deal with non-performance promptly (don't wait till a few days before the work is due, to complain that someone is not doing their share). If necessary, the coordinator will split a group, and leave anyone who doesn't participate effectively in a group by themself.
How to submit your work: produce a PDF file called <groupName>_Asst3.pdf, containing the report whose structure is described in detail below. Also, each student should produce a file that is a compressed folder with your codebase. This file should be called <unikey>_Asst3_code.zip or similar. Each of these files should be uploaded at the corresponding submission link in Canvas; the report should be submitted by only one member of the group, while each member submits their own codebase.

Weekly progress steps
Week 9: In lab, group membership should be settled, (and communication should be arranged for the rest of the assessment). Also, the members should agree on which of the groups of functionalities will be done by each person. As well, the data owner needs to be chosen, and they should use pgadmin to give each member permissions to access the appropriate tables.

Week 10: each student should upload a file with the SQL commands they will use, at the heart of their code, for each of the required functionalities [Note that due to NTEU industrial action, there will be no labs in week 10]

Week 11: each student should show the lab demonstrator (or upload a file if they miss lab) the code which they have written in database.py, for each of the required functionalities

The task:

The starting point for this assessment is the data-backed web-app that was used in the week 8 lab, accessing a small artificial dataset about university study.

Each group member is asked to extend this web-app with some extra functionality (extra webpages in the website, each page supporting some reporting or modification of the data) as described below. The group as a whole needs to produce a report, with sections that describe each member's work, and also some jointly-written sections.

Each member can choose one of the following groups of functionalities to create (if the group has less than 5 members, some of these will not be done). Each group asks for 4 webpages to be created. Note that each member is expected to write the code to create these pages, in their own copy of the web-app (using their own login etc for the postgresql access) but all should access the same dataset, which is that of one group member who is chosen as the data owner [see below]

- A group of pages is for information about the classrooms. One page should list all the classrooms (showing the classroomid, number of seats and type of room). One page should allow the user to search for rooms with more than a given number of seats. One page should produce a report showing how many classrooms there are, of each type. One page should allow the user to add a new classroom to the dataset.

- A group of pages is for information about the academicstaff. One page should list all the academicstaff (showing the id, name, department, address [but not the password or salary!]). One page should allow the user to search for staff in a particular department. One page should produce a report showing how many staff there are, in each department. One page should allow the user to add a new academicstaff member to the dataset.

- A group of pages is for information about the prerequisites between units of study (from the Requires table). One page should list all the perquisite pair (showing the UoSCodes and names of the two units, and the date from which the requirement was enforced). One page should allow the user to search for all the units which are prerequisites of a given unit. One page should produce a report showing how many prerequisites there are, for each unit of study. One page should allow the user to add a new prerequities, unit pair to the dataset.

- A group of pages is for information about the lecture locations (from the Lecture table). One page should list all the locations (showing the

UoSCode, UoS name, semester, year, and the classtime and classroomid of the lecture for that unitoffering). One page should allow the user to search for units which have a class at a particular time (eg Mon12). One page should produce a report showing how many classes occur in each room. One page should allow the user to add a new lecture to the dataset.

- A group of pages is for information about the textbooks used (found in the textbook column of the UoSOffering table). One page should list all the units (by UoSCode, semester, year, and unit name) with the textbook for each. One page should allow the user to search for all unitofferings which use a particular textbook. One page should produce a report showing how many unitofferings use each textbook. One page should allow the user to change the textbook for a given unitoffering.

Extensions: Any member who wants to can also write some extra functionality, that extends the schema of data with some additional tables (that must be relevant to the domain), and then provides functionality involving the extra information. For example, someone might choose to include information about assessment tasks (such as their value, type, due date, and the unit offering they form part of), and then a page to list the assessment tasks in a given unit.

Data management. One member should be chosen as data owner for the group. This member will need to GRANT appropriate permissions for the other members, so they can access the tables they need to write their group of pages. Note that the queries each other member writes, will need to mention the schema name for the data owner (eg SELECT … FROM abcd1234.UoSOffering WHERE …) It is expected that each group member will tell the data owner what table access they need, and then the data owner can grant it. Similarly, if any member wants to alter the data schema (for an extension), they will tell the data owner what DDL commands to perform, but the data owner will execute those commands on the shared dataset, through pgadmin,

Testing. The group is expected to ensure that the work of each member is tested by another member of the group. As indicated in the marking scheme, testing needs to be done for the SQL query (as a query) by submitting it directly to the database via pgadmin, as well as testing the webpage itself by opening it a browser. Higher marks also require tests against different states of the database, not just the initial state that arises from loading the tables with the data we supplied.

Security discussion. The group is expected to discuss and jointly write about the security aspects of the system (for the situation where real data about students, classes, academicstaff, etc are in the data, rather than the artificial data we supplied). The report should indicate what security goals would be appropriate, and then discuss mechanisms in which the system tries to achieve security, and also it is desirable that the report evaluate the effectiveness of the mechanisms (for example, it could indicate some gaps between the security and what the goals are aiming for). The report should consider both the security provided by postgresql itself (eg the different database users and their logins), and the

security built in to the webapp (for example, the way the app manages logins by end-users who claim to be students of the system).

The report to be submitted.
The report should be written jointly, and contain the following sections.

For each member, one section which gives the member's name and sid, and then reproduces the code that member wrote in database.py. [There is no need to show parts of the code that we provided for lab8]. Any extensions done by the member should be mentioned explicitly here, as well in the "extensions" section below.

One section which describes the way the work was tested and checked for quality. The testing should be done by someone other than the author of the aspect being tested, and should check the SQL query as well as the webpage. Note that the goal of this section isn't to argue that the system has high quality, but rather to accurately show that thorough checks were done.

One section which discusses the security of the system, covering appropriate security goals, the mechanisms both in postgresql and in the webapp code. And for higher levels of mark, there should be an evaluation of how well the mechanisms achieve (or not) the security goals. Note that the goal of this section isn't to argue that the security is high; rather the section should accurately describe the strengths and weaknesses of the system security.

One section indicates any extensions done, with a statement of which student did the extension, and the DDL statements which were done to extend the schema of the shared data. The code with extra functionality will be in the earlier section corresponding to the member who wrote that.

Marking scheme:
Based mainly on information in the submitted report, but also the code can be considered by the marker

Group marks [all members get the same mark] /10

Quality assurance and testing (all members get the same mark)/2
2: Report describes a thorough process for checking quality, and clearly reports the outcomes, with good suggestions on how to improve any identified issues
1.5: report describes some testing on each webpage of each member as well as on each SQL query, by someone other than the author of that page/query, with a clear statement of the outcome (ie report indicates , for every page and SQL query, what test was run and by whom, and it shows the output produced, and indicates either that this was expected, or shows difference between actual output and what was expected)
1: report describes some testing that was done on every SQL query in the codebases, by someone other than the author of that query, with a clear statement of the outcome (ie report indicates , for every SQL query, what test

was run and by whom, and it shows the output produced, and indicates either that this was expected, or shows difference between actual output and what was expected)
0: Report describes some testing that was done

Security discussion (all members get the same mark)/4
4: A thorough discussion of security goals and mechanisms used both in the postgresql system and the webapp code, with an thorough evaluation of the fit (or gap) between what is achieved by the mechanisms and what is expected by the goals, and insightful suggestions of changes that would improve the security.
3: A clear statement of security goals, detailed description of the mechanisms used both in the postgresql system and the webapp code, with a sensible evaluation of the fit (or gap) between what is achieved by the mechanisms and what is expected by the goals.
2: A clear statement of sensible security goals (appropriate for the domain, when real data are stored) and description of some of the security mechanisms in the system.
1: Some aspects of security are described

Extensions (all members get the same mark)/4
4: Two extensions achieved, at least one of which which demonstrates some kind of functionality beyond what is shown in the standard tasks (which are listing data in a table, searching for data based on a value entered textually, inserting or modifying data, entered textually and producing a grouped summary report). For example, the extension might offer a drop-down of choices, or display the data in a more sophisticated way.
3: Two extensions achieved well (including well-chosen extensions of the schema of the shared data)
2: Some extension achieved (including extension of the schema of the shared data)
1: Some extension attempted

Individual marks / 5
Individual contribution based on the student's progress contributions in week 10 and week 11)/1
0.5 for showing progress in week 10
0.5 for showing progress in week 11

Individual contribution based on code done by the individual student and shown in the corresponding section of the group report/4
4: High-quality code, including an extension which demonstrates some kind of functionality beyond what is shown in the standard tasks (which are listing data in a table, searching for data based on a value entered textually, inserting or modifying data, entered textually and producing a grouped summary report). For example, the extension might offer a drop-down of choices, or display the data in a more sophisticated way.
3: Correct and sensible code for each of the functionalities expected in the appropriate group, including appropriate SQL commands and passing the parameters and results appropriately between the Python and the SQL

2: Correct SQL commands for each of the functionalities expected in the appropriate group
1: Correct SQL command for at least one functionality