Final project details

Brief Summary

The final project will be to create a GUI program that works in tandem with a database (an embedded system DB) to perform some meaningful work. This application will have a front end application (which is assumed to be written in Java), a database backend (which must be written in MySQL), the database must be populated with a significant and reasonable amount of meaningful data (junk or garbage data like “uh$iog88” is not acceptable). I have provided you with a number of resources to find real data sets from a number of fields that you are free to use but you are also free to use your own data should you wish.

The application will query the database (and the data contained therein), perform some meaningful work on the data, display some results to the user, and store some permanent data back into the database.

Details

The project is to be written in either Java or Python for the front end GUI and must use MySQL database and server as a backend. This is to mimic the work that we have done in class. The use of MySQL is non-negotiable but I am willing to consider other languages for the front end but each student must contact me individually with a short proposal as to which language they wish to use and why they wish to use the other language. Even if one student is approved to use a language (such as C#) this does not mean that every student is approved as approval is given on an individual basis. Approval for non-Java or non-Python languages is not guaranteed. No approval is needed for Java or Python.

Conceptually the project can be broken into five major components and will be graded based on these:

- The application (GUI)
  - Hereafter referred to as ‘the GUI’
- The database (the tables)
  - Hereafter referred to as ‘the Database’
- The SQL used to query the database
  - Hereafter referred to as ‘the SQL’
- The Data Processing performed by your application
  - Hereafter referred to as the ‘data processing’
- Some self-generated, permanent data
  - Hereafter referred to as ‘the Permanent Data’

The GUI

The GUI is the front end of your application and is the only part of your application that the end user should be expected to interact with. You may **freely use either Java or Python** as the base language for this part of the project. You are **encouraged to seek out a graphics library** to aid you in writing a GUI that is both more clean looking and work quick to write than the Java SWING library that we studied in class. While you are free to write you GUI using SWING and you will lose no points for doing so, and you will gain no points for using a higher level library, using a higher level library will allow you to create a sharper look in a shorter period of time, which should be a better use of your time.

The GUI should provide the end user with the ability to interact with the Database without the need to write any sql. This interaction can be in the form of buttons, or textboxes where the user enters search parameters or similar terms, or similar sorts of GUI interfaces; however, **the user should not be expected to write any code (and should not be able to) in order to retrieve any data** that they require. This means that all functionality that your program might provide will have to be preprogrammed by you and given in the form of the user interface.

The **user interface itself should be usable and understandable** but the goal of this project is to understand database, not GUI design so functionality is the primary concern. You will not be graded over the looks of your GUI but you will be graded over the functionality of your GUI (is your GUI functional, does it provide all the needed functionality, do all the buttons actually work). You will also be graded over whether the layout is understandable; randomly placed buttons such as a search bar in the upper left but the 'go' button to initiate the search being in the bottom right is not acceptable. The layout of the GUI should be logical and ordered. But precise layout, following a certain look-and-feel, the colors of your GUI, and such aesthetics are not graded in this assignment.

The Database

This is where your data is stored. The database **must be written using MySQL** just as we have used for this entire class. This is where all of your tables are created and where your raw data is stored. This is also the location where you must send your sql queries to retrieve data.

The size of the database must be relatively large. It is difficult to quantify the size of a database but I can give some examples: The classicModels and Sakila databases are perfectly fine examples of large databases (Too large to be expected for this class with sakila having 18 tables and hundreds of records each). However, we can consider the world database. It has three tables each of which has a couple of thousand records each. This is a good size of database for the kind of work that we performed on that particular database and is about the size of database that we are looking to create for this assignment. **You may not use any of the datasets that I used as examples in class**

When considering the database that you are constructing, it must be large enough to support the goal of the application; if it is a stock market analyzer then it must have sufficient amount of market data to provide a reasonable analysis, if it is a population census analyzer then it should have all the census data, if it is analyzing credit card fraud then it should have as much data on said fraud as you can get. **The expectation is that you should have two to three tables in your database** that may or may not be related by a primary-key/foreign-key relationship (depending on if that relationship is appropriate or not), **each of which contain several hundred to several thousand rows.**

Do note that many of the datasets that you find online either lack a primary key or the primary key that they use is that the same as the primary key that you will want to use in your second table. If you need to relate two tables together then you will have to load the data into your table and then ALTER your data to make the tables match by either adding new columns to act as a primary key or changing the existing primary keys to be similar. Along this line thought, the reason that you will need to relate two table together **is because all tables that you use in your final project must be in Third Normal Form (3NF)**

The SQL

This is the SQL code that is embedded in your application code (Java or Python) and is used to retrieve or modify data that is stored in your database. It is not enough to have a good looking GUI and have good data in your database, you must also be able to interact with it. That is performed via **the embedded SQL that you write and is a graded portion of this assignment**. **Your program must have a sufficient amount of intractability with your database to perform the primary function of your application**. Example, if your database stores stock market data, then your application must have a sufficient level of functionality preprogrammed into it to be useful to the end user. Functionality might include buttons such as: display market info for a certain company, display market info for the top stock market index trackers (Dow Jones, S&P 500, etc), calculate growth or loss for a given company over the past 6 months, etc. Your application must have these (or similar) functionality built into and readily accessible to the end user. These functionalities are supported via embedded SQL written into things like buttons or similar components.

Therefor you will be graded on the presence, sufficiency, and functionality (does it actually work) of said SQL.

The Data Processing

After your data has been created in the database and loaded into your application via some embedded SQL commanded, your application must then do something with the data that has been retrieved. What this means is that **your application must do more than simply display the data that is stored in your database, it must provide the user with some form of usability, functionality, or processing**.

Some examples of what is not appropriate would be a database of movies and TV shows and your application displays two tables; one table of movies and one table of TV shows. There is no functionality here and no processing that has been performed. Instead, you might do something like allow the user to search based on a set of parameters like genre or movie length or text in a description. That provides a functionality and would satisfy the Data Processing requirement of this project.

Another example might be a stock market analyzer. Imagine that you have a list of stock data for a range of companies over the past year and you want to create a custom formula to determine if a particular company is good to invest in right now. So you take the data that you have for company X (no relation to the company formerly known as Twitter) and put its stock market data through your custom formula and you get a value. Now say that this value ranges between -1 and 1 indicating whether now is a good time to buy or sell stock in company X. This is an example of data processing because you have taken data that already exists and performed some kind of mathematical calculation on it and then displayed it to the user.

So your application must provide the user with the ability to search your dataset (usability), it must provide extra information in the form of custom information that is not original in your dataset such as dynamic attributes that are calculated and then displayed (functionality), or your application must generate its own custom values (like the stock market example above) and display those to the user (processing).

The Permanent Data

Your application must write its own set of data to the database. You will not only need to use pre-generated data that is initially stored in the database, but you will also need to retrieved that data from the database, do some processing of that data, and then store something into the database based on that processed data. The exact nature of what you store into the database will vary from project to project but some examples might include: In the stock market example above, after analyzing certain companies you may identify certain ones as worth investing in so write those companies into a table to invest into. Or in a database that stores geological data about potential underground oil reserves in the US (which is available to you guys from the US geological survey), do some processing to identify more likely sites and then write those sites into another table.

Then, after this data is written, it must be retrievable again. So in the stock market example, after you have identified many companies and written them, your program should be able to write a list of all these companies to your GUI (such as in a table). **This data that is stored into a table does not count to the two to three table minimum needed by the Database requirement**. So in the end your database may have something like three tables of initial data plus this table to store your data for a total of four tables.

So your program must process data, store some self-generated data, and then be able to retrieve and display and/or process that self-generated data.

Rubric

The entire project is worth 100 points. Each section of the project is worth 20 points. So having a good looking and fulling functional front end is great, but that is only worth at most 20 points. Likewise, having a great database is good, but also only worth 20 points. To get full credit, you need to bring everything we have learned throughout the semester together.

The Rubric below shows the five categories (Application, Database, SQL, Data Processing, and Permanent Data) and what you need to do in order to maximize the points earned in each of these categories. For example, if you create an application that starts but has crashes then that is only worth somewhere between 5.2 and 10 points (26% to 50% of 20 points, depending on exactly where in the quality range the program falls). Another example is you have a fully functioning database, but it is small and contains ‘fake’ data (examples such as first name = fhxb and last name = ghjk, obviously these are garbage data), so while this database might be fully functional, this would be worth between 51% to 75% of 20 points.

Percentage	0-25%	26-50%	51-75%	76-100%	Notes
Application	The application is non-existent or non - functioning	The application contains only the minimum to work and/or crashes	The application works but is not user friendly	The application works and is user friendly	Remember that this is not a programming class, looks are secondary to function.
Database	The database is non-existent	The database contains errors	The database is too small or contains garbage data	The database is appropriate	Garbage data means ‘not real’ data like “h7892t34”. You should use a real data set.
SQL	You application does not actually connect to the database or does not use SQL commands to query the database	The SQL commands used contains errors and returns incorrect results	The SQL commands used returns incorrect results only in certain instances such as edge cases	The SQL always works correctly	Edge cases might include certain types of obscure searches that a user typed into a search bar that you did not account for in your testing or other similar things.
Data Processing	Your application only displays data but does not add any additional processing or functionality	Your application is severely lacking in the amount of data processing needed to support the intended goal of your project or there are major errors in your data processing.	Your application has Data Processing but not enough to fully support the intended goal of your project or there are minor errors in your data processing	Your application has an acceptable amount of processing to perform the primary function of your project	The data processing that is performed should be enough to support the stated goal of your project and it should work without errors.
Permanent Data	Your application does not store any permanent data	Your application stores permanent data but with errors	Your application stores permanent data, but does not reuse the data ever again	Your application stores and reuses its own data	Remember that you have to create your own data and then be able to reuse that data again somewhere in your program.

Submission

Submitting your assignment involves the following:

1. Submitting your source code
    a. You must submit all the source code used to create your GUI and your database
    b. The code needed to create your database must be submitted in the form of .sql files
    c. This will include and datasets that you downloaded, and .sql needed to create tables, and any other support sql that was needed to set up the MySQL server.
2. Documentation about the project
    a. You must submit references to any datasets that you used that are not yours
        i. Remember you have to cite your sources
    b. You must submit references to any external libraries that you used (such as graphics libraries)
        i. Remember you have to cite your sources
    c. A set of complete installation instructions
        i. If I can’t run your code, then I can’t grade your code.
        ii. Be sure your instructions are complete
    d. A set of complete usage instructions
        i. This should include every feature that your application has (if you don’t list it then how am I going to know it exists?)
        ii. As well as how to use said feature
    e. (optional) Any additional comments or instructions that I should know such as:
        i. If you took two (seemingly) unrelated datasets and added primary keys to them, include a description of how you added the keys, how you related them, and a justification for this (ie, how are you certain that you did this correctly?)
        ii. Anything else that you want me to know about your project
3. A video about 2-5 minutes long demoing your project
    a. This video should show only the function parts of your application. No technical details.
    b. The video should be narrated by you describing the features as necessary
        i. Don’t assume that I know what is happening just because you point at a button and click it.
4. (optional) An executable version of your application
    a. Always a good idea to include but not strictly necessary in this class because this is a solo, database course not a software engineering course.
5. Submission type
    a. Submit all your files (documentation, source code, etc) in a single archive (.zip)