Project Details

Database Systems Course (2017-2018) – Final Project

The goal of the project is to design and implement a useful and interesting web application related to **music: e.g. songs, artists, albums, music videos, lyrics etc.** and should Include data imported from at least two, semantically different, external data sources imported into a MySQL database. (See examples below).

Project requirements:

- ★ Design and create a web-application, with an underlying mysql database schema for storing, querying, updating information from two sources, at least one should be via an API as stated below
- ★ Populate your database with **at least** 20K unique records containing music-related data from at least one of the following API services. See a full list here (any other API should be approved by the course staff):
    - ★ Facebook API
    - ★ Youtube API
    - ★ SoundCloud API
    - ★ Vemeo API
    - ★ Last.FM
    - ★ MusicMatch
- ★ Your database should contains at least 6 different tables
- ★ The application should contain at least 7 *different* **Complex SELECT queries.** That includes:
    - ★ At least one full-text search query (data should be indexed correctly)
    - ★ All others should be non-simple (i.e, a plain SPJ query). For example, queries that contain: nested queries, conditions, GROUP BY + HAVING, aggregations, EXIST, etc.
- ★ The web-application must support at least 3 additional queries that use INSERT or UPDATE for updating the database via the API.
- ★ Important: Your app should not rely on “continuous usage” (for example, a social) since we need to be able to quickly check if it works as expected or not.
- ★ The web-application should be deployed and run on the university web servers and use its MySQL server as a database server. Users and passwords to the DB will be sent after you submit the group’s details.

Examples:

★ "Song.ly": The perfect app for finding the right song for the right situation. Want to dedicate a song to your loved one? Song.ly's database contains thousands of lyrics and their statistics, and let you search for pop songs that mention the word "baby" more than 5 times.

★ "Musify.Us": Do you have a hard time on deciding what to listen to, as a group of friends? type in the musical preferences of each memeber, and the situation you're in (i.e studying, road trip etc.) and get your playlist for the day!

★ "MusicGeek": All your "cool" friends listen to "cool" music? the MusicGeek allows you to smartly choose your music by performing complex searches such as "Top 10 songs liked by people living in Berlin having less than 100K view"

Working Teams:

★ Work should be done in groups of 4-5 students. All groups should be approved by email.

★ Send the TA an email by **December 19th** with the details (students names and ids) of your team (one mail per group). Otherwise you will be randomly assigned.

Coding guidelines:

★ Project will be written in Python or PHP.

★ UI will be in HTML (any Javascript library/platform is allowed to be used)

★ You can use any external libraries as long it:

1) can be installed (by you) and work on the university servers properly. It is your responsibility to ask the system dept. to install external libraries.

2) Does not automatically create a schema.

3) Does not perform database optimisations.

4) Does not generate SQL queries.

★ Your code should be readable.

★ Errors should be handled

DB design

Should be done according to the principles taught in class, and in particular:

★ Give meaningful names to tables, fields, indexes, keys, etc.

★ Use keys and foreign keys.

★ Use indexes where needed to optimize your queries.

★ Explain why did you choose this specific DB design, and discuss alternatives.

★ Avoid redundant data as much as possible (e.g., by the decomposition principles of lesson 9).

Documentation

★ User manual:

- What does your application do? – an overview
- Administrator username and password (if exist)
- The screens of the applications, how to get to them and what are their features
- How to perform DB Update via the Web UI

★ Software documentation

- DB scheme structure (also explain your choices while designing the DB)
- DB optimizations performed (i.e, indexes, storage engines etc.)
- Description of each of the 7 complex queries. Please explain how you optimized the queries and db design to support each query.
- Code structure
- Description of your API use
- External packages/libraries that you used
- General flow of the application

Submission and Grading System

★ As mentioned earlier, the we application will be uploaded to the school's web servers and use the MySQL server as its database server.

★ You will submit your source code and your SQL file that creates the DB as well as the documentations , in a single ZIP file, in the following structure (Do not submit external libraries source code) :

- /SRC
    - /API-DATA-RETRIVAL
    - /APPLICAITON-SOURCE-CODE
    - /CREATE-DB-SCRIPT.sql
- /DOCUMENTATION
    - /URL-TO-THE-APP.txt
    - /NAMES-AND-IDS.txt
    - /USER-MANUAL.pdf
    - /SOFTWARE-DOCS.pdf
    - /MYSQL-USER-AND-PASSWORD.txt

Bonus (Up to 10 points)

★ Originality in design, features, "thinking out of the box"

★ Interesting algorithms implemented

★ Exceptionally convenient and aesthetic UI

Tips and Advices

★ Make sure you invest efforts in the right places. This is a Database project, and the focus should be on the database design, optimizations, and queries rather then the UI.

★ The “complex queries” constraint is not well defined. We hope you understand by now which queries are simple and which are complex. You can consult the course staff if you are not sure.

★ All APIs have a daily/hourly usage limits. Make sure you retrieve data efficiently so you’re not exceeding the limit. Make sure to start this process in advance as it might take several days.

★ As your application must be deployed on the university servers, make sure ***in advance*** that it is running successfully (e.g. does not require any external library and is able to communicate with the sql server). For any system issue please contact system@cs.tau.ac.il and address the course staff if your issues are not resolved.

★ Come to DB day (it is optional, but recommended..)

★ And of course, start early! Similar to Rome, your entire project can not be built in a day..