Course Project

CS 348 - Spring 2021

(Total weight 30%)

Project Description

Important Dates

- Project information: Mon, May 17 (week 2)
- Milestone 0 (preparation): Mon, May 24 (week 3) [Not Graded, but important]
- Milestone 1 (proposal): Mon, June 14 (week 6) [30%]
- Milestone 2 (mid-term report): Mon, July 5 (week 9) [30%]
- Demo slots: Aug 09-13 or video demo due Aug 13, 11pm
- Final report and code submission: due by the scheduled demo slot; updatable until Aug 13, 11pm. [40% including the demo]

Overview

The project is to build a database-driven application from the ground up. There will be some examples and instructions to help you get started. No prior experience in developing such applications is assumed. This document also describes a number of possible ideas.

Submission and Grading

There will be three milestones and a final project demo; see Important Dates above for due dates. You will find the details of what to submit for each checkpoint later in this document under respective sections. Only one member needs to submit through MarkUs – "CS 348 Spring 2021" at (`https://www.student.cs.uwaterloo.ca/~markus/`) on behalf of the entire project team; make sure all members are added to the submission. Because of the open-ended nature of course projects, certain instructions may not apply to your particular project; when in doubt, consult the instructor. One TA will be assigned to each group, for the purposes of grading and helping direct the projects.
Each project will be graded on a scale of 0-100 points. A breakdown is as follows:

1) 30 points for submitting the required work at three checkpoints (milestone 1, milestone 2, and demo). (milestone 0 will not be graded);
2) 30 points for completing the proposed work;
3) 40 points for the quality of the work.

Out of the 70 points (2 & 3) for completeness/quality, 10 points are reserved for impressive and/or innovative work beyond what is expected. In other words, meeting the expectation will ensure a project grade of at least A-, but A and A+ will require exceptional work. Each team will be able to talk to their assigned TA about what will qualify as "beyond expectations" during the project development process. What the "required work" means may evolve over the course of the project. Milestone 0 ensures that each student finds a team and get familiar with the environment setup. Milestone 1 proposal helps you get a feel for the amount of work involved, and work with you to ensure that it meets the minimum requirements of depth and scope for a course project.

Teamwork

The project should be completed in (4 or 5)-person teams. Any other team size requires explicit approval from the instructor (please make your request through Piazza private questions); team sizes below 4 or above 5 are strongly discouraged. Regardless of the team size, an equal amount of work is expected and the same grading scale will be applied. All members in a team will receive identical grades for the project. You are required to report each team member's effort and progress in the milestone and final reports. **If there is any problem working with your team members that you cannot resolve by yourself, bring it to the instructor's attention as soon as possible.** Last-minute complaints of the form "my partner did nothing" will not be entertained. You can use Piazza to pitch your project ideas by presenting a few lines of your ideas (project domains or types you are interested in) and the number of additional team members you are looking for.

Platform Issues

To develop the project, there are two main options for platforms: (i) DB2 on the school servers (e.g. `linux.student.cs.uwaterloo.ca`) from Assignment 1; (ii) MySQL on local machines. We provide some sample codes and instructions for you to start with one of two options. If you wish, you may use other languages, tools, or application development frameworks. Setting up the whole application/database stack is non-trivial and can be a rewarding experience. However, the course staff can only support the technologies used by the provided examples.

Tasks Overview

The project is to build a database-driven application. Specifically, you will need to complete the following tasks over the course of this semester. Note that different members of a team can work on some of these tasks concurrently.

1. Pick your favorite data management application. It should be relatively substantial, but not too enormous. Several project ideas are described at the end of this document, but you are encouraged to come up with your own. When picking an application, keep the following questions in mind:

    a. How do you plan to acquire the data to populate your database? Use of real datasets is highly recommended. You may use program-generated "fake" datasets if real ones are too difficult to obtain.

    b. How are you going to use the data? What kind of queries do you want to ask? How is the data updated? Your application should support both *queries* and *updates*.

2. Design the database schema. Start with an E/R diagram and convert it to a relational schema. Identify any constraints that hold in your application domain, and code them as database constraints. If you plan to work with real datasets, it is important to go over some samples of real data to validate your design (in fact, you should start Task 7 below as early as possible, in parallel to Tasks 3-6). Do not forget to apply database design theory and check for redundancies.

3. Create a sample database using a small dataset. You may generate this small dataset by hand. You will find this sample database very useful in testing, because large datasets make debugging difficult. It is a good idea to write some scripts to create/load/destroy the sample database automatically; they will save you lots of typing when debugging.

4. Design a user interface for your application (e.g. command line-based on school serve, or web-based). Think about how a typical user would interact with the database. Optionally, it might be useful to

build a “canned” demo version of the interface first (i.e., with hard-coded rather than dynamically generated responses), while you brush up your user interface design skills at the same time. Do not spend too much time on refining the look of your interface; you just need to understand the basic “flow” in order to figure out what database operations are needed in each step of the user interaction.

5. Write SQL queries that will supply dynamic contents for the interface you designed for Task 4. Also write SQL code that modifies the database on behalf of the user. You may hard-code the query and update parameters. Test these SQL statements in the sample database.

6. Choose an appropriate platform for your application. JDBC? Python or PHP? JavaScript or plain HTML? Start by implementing a “hello world” type of simple database-driven application, deploy it in your development environment, and make sure that all parts are working together correctly. The course website will provide pointers to working examples.

7. Acquire the large “production” dataset, either by downloading it from a real data source or by generating it using a program. Make sure the dataset fits your schema. For real datasets, you might need to write programs/scripts to transform them into a form that is appropriate for loading into a database. For program-generated datasets, make sure they contain enough interesting “links” across rows of different tables, or else all your join queries may return empty results. Keep in mind that the school severs has limited capacity: for larger databases, you may need to consider cloud-based platforms.

8. Test the SQL statements you developed for Task 5 in the large database. Do you run into any performance problems? Try creating some additional indexes to improve performance.

9. Implement and debug the application and the user interface. Test your application with the smaller sample database first. You may need to iterate the design and implementation several times in order to correct any unforeseen problems.

10. Test your application with the production dataset. Resolve any performance problems.

11. Make your application fancier! You may add additional features such as well designed/user-friendly interface; efficient query answering; support multi-user access control; well-tested applications (corner cases, potential security issues).

Milestone 0: Project Preparation

You should have formed a team and started thinking about task 1-6. In particular, your team should get familiar with one of the platforms and be able to run "hello world" type of simple database-drive applications (task 6). Some examples (ProjectNote-PhP-MySQL.pdf, ProjectNote-JDBC-DB2.pdf) are provided on Learn under Course Project to help you start. The goal of milestone 0 is to ensure that you have found a team, and your team has already tried out the chosen programming platform and environment (as switching platforms later will be painful and time consuming).

Submit the following electronically under "Project Milestone 0" via MarkUs by **Mon, May 24, 11pm** (week 3)

1. A progress report **report.pdf** should at least contain:
    - A brief description (1-2 paragraphs) of your application: which dataset do you plan to use, who are the users of the application, who are the administrators of the database systems etc, what functionalities you may like to support.
    - A brief description (1-2 paragraphs) of your choice of platform (school servers, local machines) and user interface (e.g., how users interact with your application? command line or web or GUI, etc?)
    - A section on **members**, that list the members of your team, and for **each member**, a description of effort and progress made by this member to date.
2. A .zip or .tar.gz archive of your source code **code.zip**. The source code directory should at least contain:
    - README.txt to describe how to create and load your sample database to your chosen platform. YOu don't have to use the datasets described in the report. Toy datasets (e.g. a single table) can be used.
    - Source code for your "hello world" type of simple database-drive applications with your sample database (e.g. the application allows user to connect to the database, to select some rows from a table, etc.) This just ensures you have (installed and) tested the database systems and simple application code on school servers or your own machines for your interested applications.

Milestone 1: Project Proposal

You should have completed Tasks 1-6 and have started thinking about 7. If you plan to work with real data, you should also have made significant progress on Task 7 (you should at least ensure that it is feasible to obtain the real dataset, transform it, and load it into your database). **Submit** the following electronically under "Project Milestone 1" via MarkUs by **Mon, June 14, 11pm** (week 6):
[30 points in total]

1. A progress report **report.pdf** should at least contain:
   - A brief description of your application (Task 1, 3, 4, 5, 6). Important points:
     - Who are the users of the application? How the user interacts with your application and how the application respond to the users' requests? You can write a brief English description of how users interact with the interface (e.g., "the user selects a car model from a pull-down menu, clicks on the 'go' button, and a new page will display all cars of this model that are available for sale". This is an example of one feature/functionality). Or, instead, you can submit a canned demo version of the website.
       Try to list all the **key** features/functionalities of the application. Feel free to list additional features that your group are interested in implementing if possible. We assume that different features require different underlying query templates. For example, 'SELECT a FROM r WHERE b = 0' is considered the same as 'SELECT b FROM r WHERE a <1'. Try to list possible features. We expect at least 5 of them.
       Note that this is a proposal. You don't need to implement all of them, but we do expect you have written down the related SQL queries in the report and tested them.
       [10 points. Guideline: 1 point for one unique feature description; and 1 point for its corresponding SQL query/update and its expected output with sample database (not full dataset). These SQL query description should map to the code implemented for test-sample.sql and test-sample.out]
     - A plan for getting the data to populate your database, as well as some sample data. **[2 points. Guideline: 1 point on the description of the data; 1 point for how the data is generated/cleaned/imported for your application.]**
     - Describe the system support: e.g. programming framework chosen for the application (e.g. OS that run for your application and DBMS or school server), database support for the backend (e.g. MySQL, DB2, etc) **[2 points. Guideline: 1 point for framework, 1 point for db description.]**
   - Design database schema (Task 2)
     - A list of assumptions that you are making about the data being modeled. **[2 points, e.g. "user id is unique".]**
     - An E/R diagram for your database design. **[4 points for a well-thought E/R diagram to realize the application and assumptions described in the report. Guideline: -1 point for any unclear explanation of the terms.]**
     - A list of database tables with keys declared (relational data model). **[4 points for proper translation of your E/R diagram into relational data model. Guideline: -1 point for any incorrect translation.]**
   - A section on **members**, that list the members of your team, and for **each member**, a description of effort and progress made by this member to date. **[1 point]**
2. A .zip or .tar.gz archive of your source code **code.zip**. Note that we won't test your code, but the code will be scanned by the TA. Make sure what you describe in the report reflects the code (e.g. test-sample.sql, test-sample.out, application platform). The source code directory should at least contain:

- A README file describing how to create and load your sample database; and how to run your working database-driven application, and what feature it currently supports. **[1 point]**
- Files containing the SQL code used for creating tables, constraints, stored procedures and triggers (if any). **[1 point]**
- A file test-sample.sql (or other corresponding files) containing the SQL statements you wrote for Task 5. **[1 point]**
- A file test-sample.out showing the results of running test-sample.sql over your sample database (not the full dataset). **[1 point]**
- Code implementing a simple but working database-driven application on your chosen platform (1-2 simple features/functionalities can be run), which can serve as a starting point for completing your project. **[2 points for 1 feature]**
- If applicable, any code for downloading/scraping/transforming real data that you have written for Task 7 so far.

Milestone 2: Project Mid-term Report

You should have completed Tasks 1-9 and have made good progress on 10. **Submit** the following electronically under "Project Milestone 2" via MarkUs by **Mon, July 5, 11pm** (week 9):
[30 points in total + 2 bonus points]

1. A progress report **report.pdf** should at least contain:
   - Add details to all the requirements from milestone 1 with more details. Highlight (e.g. use a different color) the changes and new descriptions you made if there are (e.g. new changes, E/R diagram, and a list of tables). If the TA thinks that the report.pdf from milestone 1 has a lot of details already, you can keep the same details and get full points. Examples for details: you can add snapshot of the features, elaborate the features, or add new features.
     - Detailed application description on its purpose, use, etc **[1 points in total: 0.5 point for stating the purpose and use, 0.5 point for detailed motivation and description.]**
     - Detailed feature description. We expect at least 6 of them. **[9 points in total: 0.5 point for listing a feature and its underlying SQL query, 1 point for the detail (e.g. feature snapshot, test cases with production dataset and expected output) of each feature.]**
     - Detailed plan for getting data **[1 point in total: 0.5 point for a plan, 0.5 point for details.]**
     - Detailed description of system support **[1 point in total: 0.5 point for a plan, 0.5 point for details.]**
     - Database schema (a list of assumptions, E/R diagram, a list of tables with keys **[6 points in total: 2 points for having a basic design; 2 points for clarity (easy to understand); 2 points for explanation on the choice of the design.]**
   - Changes you made to the database during performance tuning in Task 8, e.g., additional indexes created **[2 points in total: 1 point for each change to improve performance.]**
   - A section on **members**,listing the members of your team, and for each member, a description of effort and progress made by this member since the last milestone. **[1 point for detailed effort and progress.]**
2. A .zip or .tar.gz archive of your source code **code.zip**. Note that we won't test your code, but the code will be scanned by the TA. Make sure what you describe in the report reflects the code (e.g. test-sample.sql, test-sample.out, application platform). The source code directory should at least contain:
   - A README file describing
     - How to generate the "production" dataset and load it into your database. Do not submit the production dataset itself through if it is too big; instead, submit the URL where you download/scrape the raw data (if applicable), and the code that extracts and transforms (or generates) the production dataset. **[1 point in total with detailed instruction how to generate the "production" dataset.]**
     - State which features have been implemented and state which files that contains the implementation. We expect at least 3 features. **[6 points in total: 2 points for each implemented feature.] [0.5 bonus point for each additional implemented feature, capped by 2 points.]**
   - Code implementing a simple but working database-driven application on your chosen platform. This code includes 1-2 more features than the version in milestone 1.
   - A file test-production.sql containing the SQL statements you wrote for Task 5. You may wish to modify some queries to return only the top 10 result rows instead of all result rows (there might be lots for large datasets). **[1 point]**
   - A file test-production.out showing the results of running test-production.sql over the production dataset. **[1 point]**

Project Demo & Final Report and Code

At the end of the semester, you will need to present a working demo of your system. **[40 points in total]**

Instructions for demo:
You have 2 options for the demo. Please inform the assigned TA the choice of your team **before July 30, 11pm**.

- Option 1: Online demo with your assigned TA in **Aug 9-13**.
  Preparation work:
  - Email your assigned TA the choice of your team and set a suitable demo slot with the TA **before July 30, 11pm**. **[2 points]**
  - Inform the TA the meeting tool and all the necessary links or documents for your demo (e.g. zoom, skype, `https://meet.jit.si/`) before the demo.
  - **Note:** To ensure a smooth online demo presentation, the team are recommended to **rehearse** before the demo session. Feel free to use slides, or video, or any tools that enable a smooth presentation. We will not grade the tools or demo formats, but the contents and clarity of the demo will affect the grades.
  - Every member should be present and contribute to the online demo.
  - Consider 5-8 minutes presentation with the TA.
- Option 2: Submit a video of your demo (no more than 10 mins) on Learn (Dropbox) before **Aug 13, 11pm**.
  Preparation work:
  - Email your assigned TA the choice of your team **before July 30**. **[2 points]**
  - You may use powerpoint or quicktime to record your computer screen. (More help: `www.digitaltrends.com/computing/how-to-record-your-computer-screen/`). Feel free to choose any tools to make video, even your phone camera. You can also submit ppt files with video/audio embedded.
  - Each member should contribute to some part of the video production.

Regardless which option you take, the demo will be graded based on the following items.

- Presentation contents & clarity. Your presentation should clearly illustrate the following items for your application. **[18 points in total]**
  - Application overview **[2 point]**
  - How to use your application? (demonstrating functionalities/features) **[6 points]**
  - System support for your application and briefly explain the back-end queries for each feature **[6 points]**
  - The contribution of each member **[2 points]**
  - Summary **[2 points]**
- Application quality: **[20 points in total]**
  - 6 promised features **[2 points for each completed and well tested functionality, capped by 12 points]**
  - Impressive/innovative features (Task 11). Examples can be

- Well designed/user-friendly interface
- Efforts on improving query performance for 'millions' of records or complex queries involving many joins or subqueries
- Efforts on studying multi-user access control
- Efforts on breaking your applications with corner cases or security issues
- ....

[For each of the case above, 2 points for basic effort, additional 2 points for impressive work; capped by 8 points.]

Submit your finalized members.txt, report.pdf and code.zip electronically under "Project Final" before **Thur Aug 13, 11pm**. Please highlight the changes made by you since the last milestone.

Project Ideas

Below is a list of possible project ideas for which high-quality datasets exist. Of course, you are welcome to come up with your own.

Entertainment, sports, or financial websites

Examples include those that allow visitors to explore information about movies, music, sports, games, stocks, etc. There are already many commercial offerings for such purposes. While there is less room for innovation, there are plenty of examples of what a good website would look like, as well as high-quality, well-formatted datasets. For example, IMDb makes their movie database available `http://www.imdb.com/interfaces`; historical stock quote can be downloaded and scraped from many sites such as Yahoo! and Google Finance. This project is well-suited for those who just want to learn how to build database-backed websites as beginners. You can always spice things up by adding features that you wish those websites had (e.g., different ways for summarizing, exploring, and visualizing the data).

Websites providing access to datasets of public interest

If you are interested in doing some good to society while learning databases, this project is for you. There are many interesting datasets "available" to the public, but better ways for accessing and analyzing them are still sorely needed. Here are some examples:

- Data.gov (`http://www.data.gov/`) has a huge compilation of data sets produced by the US government.
- The Supreme Court Database (`http://scdb.wustl.edu/data.php`) tracks all cases decided by the US Supreme Court.
- US government spending data (`https://www.usaspending.gov/`) has information about and database downloads of government contracts and awards.
- Federal Election Commission (`https://www.fec.gov/data/`) has campaign finance data to download as well as nice interfaces for exploring the data.
- GovTrack.us (`http://www.govtrack.us/developers`) tracks all bills through the Congress and all votes casted by its members. The Washington Post has a nice (albeit outdated) website (`http://projects.washingtonpost.com/congress/113/`) for exploring this type of data (in predefined ways), but you can be creative with additional and/or more flexible exploration and analysis options. Vote Smart (`https://votesmart.org/`) has a wealth of additional, useful information on votes, such as issue tags, synopses and highlights.
- Each state legislature maintains its own voting records. For example, you can find North Carolina's here: `http://www.ncleg.net/Legislation/voteHistory/voteHistory.html`. Some states provide records in already structured formats, but for others, you may need to scrape their websites.
- The Washington Post maintains a list of datasets (`http://www.washingtonpost.com/wp-srv/metro/data/datapost.html`) that have been used to generate investigative news pieces. Most of these datasets hide behind some interface and may need to be scraped. Use this list for examples of what datasets are "interesting" and how to present data to the public effectively.
- Stanford Journalism Program maintains a list of curated transportation-related datasets (`http://www.datadrivenstanford.org/`).

- National Institute for Computer-Assisted Reporting maintains a list of datasets of public interest (`http://www.ire.org/nicar/database-library/`). Use this list for examples of what datasets are "interesting"—they are generally not available to the public, but there may be alternative ways to obtain them.

Your task would be to take one of such datasets, design a good relational schema, clean up/restructure the data, and build an application for the public to explore the dataset. You are welcome to come up with your own.