Course Project

CS 348 - Fall 2025

Project Description

Important Dates (all deadlines 11:59pm)

- Milestone 0 (preparation): Thu, Sep 25 [Not Graded, but important]
- Milestone 1 (proposal): Tue, Oct 21 [30%]
- Milestone 2 (progress report): Fri, Nov 14 [30%]
- Milestone 3 (final report, demo video, and code): Thu Nov 27 [40%]

Overview

The goal of this project is to build a database-driven application from the ground up. Your application will have a set of "basic features" and a set of "advanced features". 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. You will be working in a group of 4-5 students – minimum of 4 and a maximum of 5 students. There is the possibility to perform consultations with TAs/IAs regarding your idea and development of the project throughout the term.

Submission and Grading

There will be three milestones; 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 of the group needs to submit the deliverables by the deadline on Learn – "CS 348 Fall 2025" at Submit > Dropbox on behalf of the entire project team; make sure all members' names 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 TAs/IAs.

One TA will be assigned to each group, for the purpose of grading and helping direct the projects. The project constitutes **a maximum of** 30% towards your final marks in the course.
Milestone 0 ensures that each student finds a team and gets familiar with the environment setup. Milestone 1's proposal helps you get a feel for the amount of work involved and if it meets the minimum requirements of the depth and scope for a course project while ensuring that you are planning a doable project. Detailed grading rubrics can be found on later pages.

Late submission policy: For each milestone, you have 2 extra days after the deadline for submissions and the entire team loses 25% worth of marks for each additional day, i.e., we subtract that many marks from your actual score for that milestone.

Enrolling in Groups

You will indicate the groups you are with by enrolling into a group on Learn by Milestone 0's deadline. To enroll in a group on Learn, please follow the steps described in this documentation: `https://uwaterloo.atlassian.net/wiki/spaces/ISTKB/pages/1548779704/Groups+-+Students`.
Choose one group number and stick to this group until the end of the project and milestones cycle. Students should make all their submissions for each milestone through the same group link. One student uploads the files on behalf of the entire group.

Teamwork

The project should be completed in 4 or 5-person teams. Regardless of the team size, an equal amount of work is expected and the same grading scale will be applied. All team members will receive the same score for project. However, if certain team members are not contributing equally to the project, the instructor will consult the relevant TA and will allocate scores proportional to the contribution. 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 the 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, you must use 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 will be a rewarding experience. However, the course staff can only support the technologies used in the provided examples.

Tasks Overview

The goal of this 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 that uses relational data. 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 "synthetic" 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* (i.e., reading data) and *updates* (i.e., writing data).

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 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 on your user interface design skills at the same time. Do not spend too much time 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 the 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 servers have limited capacity.

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 for 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 tasks 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[1]) 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" on Learn (Dropbox) **Thu, Sep 25 11:59pm**.

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 will users interact with your application? command line or web or GUI, etc?)
    - A section on **members**, that lists the members of your team, and for **each member**, a description of effort and progress made by this member to date.
    - A link to the **GitHub repo** you will use throughout the project to store your source code. We recommend having 4 directories at the top: milestone-0, milestone-1, milestone-2, milestone-3. For each milestone, place the required documents and data under the corresponding folder. When needed use a separate README file for each directory. Share the Github repo with your TA using their Github username. If you want to organize your repo differently, this is fine, as long as you explain your organization clearly in the README of the repo.

2. The source code directory in the **GitHub repo** should at least contain:

    - README.md 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 system and simple application code on school servers or your own machines for your interested applications.

[1] This Java app is just a sample from a previous term. This term, we are not using DB2 but you can build a similar Java app with MySQL's JDBC driver.

Grading Rubrics for Milestones 1,2, and 3

We will use the rubric in Table 1 for grading milestones 1, 2, and final.

R1-R16 will be graded based on the **report.pdf** submitted for each milestone. Note that not every milestone needs to have completed each item we list in R1-R16. See Table 2 to see which items are required for each milestone.

- **R1. Application high-level description**: A high-level 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, and what functionalities you may like to support.
- **R2. System support description**: A description (1-2 paragraphs) of your choice of programming framework chosen for the application (e.g. locally deployed or on school servers or on the cloud).
- **R3. Database with sample dataset**: A description of the sample data, how the data is generated/cleaned/imported for your application, and how to use the data to populate your database.
- **R4. Database with production dataset**: A description of the production data, how the data is generated/cleaned/imported for your application, and how to use the data to populate your database.
- **R5. Schema design**
  - R5a. Assumptions: A list of assumptions that you are making about the data being modelled. For example, "user id is unique".
  - R5b. An E/R diagram for your database design: Full points for a well-thought E/R diagram to realize the application and assumptions described in the report. -1 point for any unclear explanation of terms.
  - R5c. Relational data model: A list of database tables with keys declared (relational data model). Full points for proper translation of your E/R diagram into a relational data model. -1 point for any incorrect translation.
- **R6. Basic Feature/Functionality 1**: We expect at least 5 distinct basic features for your applications (R6-R10). We consider 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'.
  - R-a. Feature interface design: A description of the interface of this feature. Who are the users of this feature? How will a user interact with your application and how does 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 Basic Feature/Functionality).
  - R-b. SQL query, testing with sample data: We expect you to have written down the related SQL queries in the report for this feature and tested them on the sample database. Please include the related SQL queries and their expected output in the report. You must submit your test database records in some format (e.g., csv), and a test-sample.sql and test-sample.out files that respectively show what the SQL query is and what the query's output is on the test database.
  - R-c. SQL query, testing with production data: Please describe the changes you made to the database during performance tuning for larger databases, e.g., additional indexes created. Please also indicate how to evaluate the performance difference with/without the performance tuning.

You must submit a "production" database (your intended real database) records in some format (e.g., csv), and a test-production.sql and test-production.out files that respectively show what the SQL query is and what the query's output is on the production database.

- R-d. Implementation, snapshot, testing: We expect a fully implemented and tested feature. The report can include the feature snapshot, the test case description (how to test the interface of this feature).

- **R7-R10. Basic Feature/Functionality(s) 2-5** is the same as R6.
- **R11. Advanced feature 1**: You will also implement a set of more advanced features than the basic features in terms of its usage of the database. Make sure you propose to your TAs what these advanced features are in advance. *You need to ask for TA's approval of these advanced features beforehand* (after M1 but before the final milestone). Specifically, your feature can be more advanced in one of 2 possible ways:
  - *Query complexity:* They can use more advanced queries than the basic features; For example: Full-text search with relevance, rankings, not exist, geo queries etc.
  - *Use of advanced MySQL features:* They can use more advanced features of MySQL, such as indices, views, transactions, constraints, join order hints, authorization (you can propose features that use other advanced aspects of MySQL as well);
- **R12-R15. Advanced feature(s) 2-5** is the same as R12.
- **R16. Member**: A description 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.

C1-C5 will be graded based on the source code posted on your GitHub repo. For milestones 0-2, we will not test your code, but the code will be scanned by the TA. For your final milestone, you will do a live demo followed by a testing session, during which the TAs will test your applications. 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:

- **C1. README file**: 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.
- **C2. SQL code (create tables, etc)**: Files containing the SQL code used for creating tables, constraints, stored procedures and triggers (if any).
- **C3. SQL queries for features, test-sample.sql, test-sample.out** A file test-sample.sql (or other corresponding files) containing the SQL statements, and a file test-sample.out showing the results of running test-sample.sql over your sample database.
- **C4. SQL queries for features, test-production.sql, test-production.out** A file test-production.sql (or other corresponding files) containing the SQL statements, and a file test-production.out showing the results of running test-production.sql over your production database.
- **C5. Application code**: The end-to-end application code that implements all the claimed features for the corresponding milestone.
- If applicable, any code for downloading/scraping/transforming real data that you have written.

D1-D6 will be graded based on the **final demo**. Your presentation should clearly illustrate the following items for your application.

- D1. Application overview
- D2. How to use your application? (demonstrating functionalities/features)
- D3. System support for your application and briefly explain the back-end queries for each feature
- D4. The contribution of each member
- D5. Summary

In addition, you will be marked on the overall quality of the presentation and demo (i.e., how polished is the demo presenation). This constitutes D6.

Task No.	Items	No attempt	Needs work	Acceptable	Excellent	Score
1	**R1. Application high-level description**	0	0.5	1.0	1.0	
6	**R2. System support description**	0	0.5	1.0	1.0	
3	**R3. Database with sample dataset**	0	1.0	1.5	2.0	
7	**R4. Database with production dataset**	0	2.0	3.0	4.0	
2	**R5. Schema design**					
	R5a. Assumptions	0	1	1.5	2.0	
	R5b. E/R diagram	0	2	3	4	
	R5c. Relational data model	0	2	3	4	
	R6. Basic Feature/Functionality 1					
4	R-a. Interface design	0	0.5	1.0	1.0	
5	R-b. SQL query, testing with sample data	0	1	1.5	2.0	
8	R-c. SQL query, testing with production data	0	0	0.5	0.5	
9/10	R-d. Implementation, snapshot, testing	0	1	1.5	2.0	
	R7. Basic Feature/Functionality 2					
4	R-a.	0	0.5	1.0	1.0	
5	R-b.	0	1	1.5	2.0	
8	R-c.	0	0	0.5	0.5	
9/10	R-d.	0	1	1.5	2.0	
	R8. Basic Feature/Functionality 3					
4	R-a.	0	0.5	1.0	1.0	
5	R-b.	0	1	1.5	2.0	
8	R-c.	0	0	0.5	0.5	
9/10	R-d.	0	1	1.5	2.0	
	R9. Basic Feature/Functionality 4					
4	R-a.	0	0.5	1.0	1.0	
5	R-b.	0	1	1.5	2.0	
8	R-c.	0	0	0.5	0.5	
9/10	R-d.	0	1	1.5	2.0	
	R10. Basic Feature/Functionality 5					
4	R-a.	0	0.5	1.0	1.0	
5	R-b.	0	1	1.5	2.0	
8	R-c.	0	0	0.5	0.5	
9/10	R-d.	0	1	1.5	2.0	
11	**R11. Advanced feature 1**	0	1	1.5	2.0	
11	**R12. Advanced feature 2**	0	1	1.5	2.0	
11	**R13. Advanced feature 3**	0	1	1.5	2.0	
11	**R14. Advanced feature 4**	0	1	1.5	2.0	
11	**R15. Advanced feature 5**	0	1	1.5	2.0	
	R16. Member.txt	0	1	2	3	
	C1. ReadMe.txt	0	1	1.5	2	
	C2. SQL code (create tables, etc)	0	1	1.5	2	
	C3. SQL queries for features, test-sample.sql, test-sample.out	0	1	1.5	2	
	C4. SQL queries for features, test-production.sql, test-production.out	0	1	1.5	2	
	C5. Application code	0	6	8	10	
	D1. Demo-app overview	0	1	1.5	2	
	D2. Demo-use app	0	2	4	6	
	D3. Demo-system support	0	2	4	6	
	D4. Demo-user contr.	0	1	1.5	2	
	D5. Demo-summary	0	1	1.5	2	
	D6. Demo presentation	0	2	4	5.5	

Table 1: Grading Rubrics for Milestones 1, 2, and final

Items	Milestone 1	Milestone 2	Final	Max Point
R1. Application high-level description	*	*	*	1.0
R2. System support description	*	*	*	1.0
R3. Database with sample dataset	*	*	*	2.0
R4. Database with production dataset	-	*	*	4.0
R5. Schema design				
R5a. Assumptions	*	*	*	2.0
R5b. E/R diagram	*	*	*	4.0
R5c. Relational data model	*	*	*	4.0
R6. Basic Feature/Functionality 1				
R-a. Interface design	*	*	*	1.0
R-b. SQL query, testing with sample data	*	*	*	2.0
R-c. SQL query, testing with production data	-	*	*	0.5
R-d. Implementation, snapshot, testing	-	*	*	2.0
R7. Basic Feature/Functionality 2				
R-a.	*	*	*	1.0
R-b.	*	*	*	2.0
R-c.	-	*	*	0.5
R-d.	-	*	*	2.0
R8. Basic Feature/Functionality 3				
R-a.	*	*	*	1.0
R-b.	*	*	*	2.0
R-c.	-	*	*	0.5
R-d.	-	*	*	2.0
R9. Basic Feature/Functionality 4				
R-a.	*	*	*	1.0
R-b.	*	*	*	2.0
R-c.	-	*	*	0.5
R-d.	-	-	*	2.0
R10. Basic Feature/Functionality 5				
R-a.	-	*	*	1.0
R-b.	-	*	*	2.0
R-c.	-	*	*	0.5
R-d.	-	-	*	2.0
R11. Advanced feature 1	-	-	*	2.0
R12. Advanced feature 2	-	-	*	2.0
R13. Advanced feature 3	-	-	*	2.0
R14. Advanced feature 4	-	-	*	2.0
R15. Advanced feature 5	-	-	*	2.0
R16. Member.txt	*	*	*	3
C1. ReadMe	*	*	*	2
C2. SQL code (create tables, etc)	*	*	*	2
C3. SQL queries for features, test-sample.sql, test-sample.out	*	*	*	2
C4. SQL queries for features, test-production.sql, test-production.out	-	*	*	2
C5. Application code	*	*	*	10
D1. Demo-app overview	-	-	*	2
D2. Demo-use app	-	-	*	6
D3. Demo-system support	-	-	*	6
D4. Demo-user contr.	-	-	*	2
D5. Demo-summary	-	-	*	2
D6. Demo-presentation-quality	-	-	*	5.5
Maximum scores per milestone	45	62.5	100	

Table 2: Basic Requirements for Milestones 1, 2, and final; * means “required”, and - means “not required"

How to compute the overall score of your project? Your final score will be a weighted average of your score for each milestone as follows:

- Milestones 1, 2, and 3 comprise respectively 30%, 30%, and 40% of your final score.
- If you obtain s_1, s_2, and s_3 for milestones 1, 2, and 3. Then your overall project score will be

$$\frac{s_1}{45} * 30 + \frac{s_2}{62.5} * 30 + \frac{s_3}{100} * 40 \tag{1}$$

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" on Learn - Dropbox by **Tue, Oct 21 11:59pm**:

1. A progress report **report.pdf** should at least contain:
    - R1. A high-level 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.
    - R2. Describe the system support (1-2 paragraphs) (Task 6): e.g. programming framework chosen for the application (e.g. OS that runs for your application and DBMS or school server), database support for the backend (e.g. MySQL, DB2, etc).
    - R3. A plan for getting the sample (toy) data to populate your database, as well as some sample data (Task 3).
    - R5. Design database schema (Task 2)
        - A list of assumptions that you are making about the data being modelled.
        - An E/R diagram for your database design.
        - A list of database tables with keys declared (relational data model).
    - R6a, R6b, R7a, R7b, R8a, R8b, R9a, R9b. We expect at least **4 features' description, their corresponding query template, sampled queries, and expected output.**
    - R16. 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.
    - A link to the GitHub repo.
2. The source code directory should at least contain:
    - C1. 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.
    - C2. Files containing the SQL code used for creating tables, constraints, stored procedures and triggers (if any).
    - C3. A file test-sample.sql (or other corresponding files) containing the SQL statements you wrote for Task 5, and a file test-sample.out showing the results of running test-sample.sql over your sample database (not the full dataset).
    - C5. 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.

Only the deliverables necessary for Milestone 1 will be graded in this milestone. If a group has achieved more, that will be graded in the future milestones.

Milestone 2: Project Progress Report

You should have completed Tasks 1-9 and have made good progress on 10. **Submit** the following electronically under "Project Milestone 2" on Learn Dropbox by **Fri, Nov 14, 11:59pm**:

1. 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 a snapshot of the features, elaborate the features, or add new features.

   A progress report **report.pdf** should at least contain:

   - R1. A high-level description (1-2 paragraphs) of your application.
   - R2. Describe the system support (1-2 paragraphs) (Task 6).
   - R3. A plan for getting the sample (toy) data to populate your database (Task 3).
   - R4. A plan for getting the production data to populate your database (Task 7).
   - R5. Design database schema (Task 2).
   - R6-R10. A description of your application features. Try to list all the **key** features/functionalities of the application. Feel free to list additional features that your group is interested in implementing if possible. We expect all **5 features' description, their corresponding query template, sampled queries, and expected output**, and at least **3 features implemented and tested.**
   - R16. A section on **members**.
   - A link to the GitHub repo.

2. The source code directory should at least contain:

   - C1. 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.
     - 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. State which features have been implemented and state which files contain the implementation. We expect at least 3 features.
   - C2. Files containing the SQL code used for creating tables, constraints, stored procedures and triggers (if any) for sample database and production database.
   - C3. A file test-sample.sql (or other corresponding files) containing the SQL statements you wrote for Task 5 and a file test-sample.out showing the results of running test-sample.sql over your sample database (not the full dataset).
   - C4. A file test-production.sql containing the SQL statements you wrote for Task 8. 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) and a file test-production.out showing the results of running test-production.sql over the production dataset.
   - C5. Code implementing a simple but working database-driven application on your chosen platform. This code includes at least 2 more features than the version in milestone 1.

Milestone 3: Project Demo & Final with Report and Code

At the end of the semester, between Nov 24 (Mon) - Nov 27 (Thu) you will need to present an **in person** demo of your system to your TA followed by a testing session where the TA will test your features (ultimately testing the correctness of the queries that underlie the features of your application).

Instructions for demo:

- By Fri Nov 21: Email your assigned TA to find a day/time for 20 minute in person time. **[You lose 2 marks if fail to do so.]**
- **Note:** To ensure a smooth demo presentation, the team should **rehearse** before the demo session. Feel free to use slides, video, or any tools that enable a smooth presentation. We will not grade the tools or demo formats, but the overall quality of the contents and clarity of the demo constitutes a small part of your marks (D6).
- Every member should be present and contribute to the demo.
- Your demo should be maximum 10 minutes. The remaining 10 minutes will be for live Q&A.

The demo presentation will be graded based on the following items.

- Presentation contents. Your presentation should clearly illustrate the following items for your application.
  - D1. Application overview
  - D2. How to use your application? (demonstrating functionalities/features)
  - D3. System support for your application and briefly explain the back-end queries for each feature
  - D4. The contribution of each member
  - D5. Summary
  - D6. Overall quality clarify of the presentation. Try to make your demo as engaging and compelling as you can.

During the testing session, the TAs can take off some marks of your milestone 3 mark for the features that are not working properly. The demo/report/code should include all the items listed in Table 1.
After your demo presentation, complete your report by Nov 27th. Submit your finalized report.pdf (including the GitHub repo) electronically under "Project Final" on Learn Dropbox before **Nov 27th, 11:59pm**.
Please **highlight** the changes made by you in report.pdf since the last milestone (M2). The grading of the final submission will be mainly based on your demo and the final report.

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 of accessing and analyzing them are still sorely needed. Here are some examples:

- Open Canada (`https://search.open.canada.ca/data/`) has a huge compilation of data sets produced by the Canadian government. My research group has done some work in this space. You can look in the introduction sections of these works to get a better feel for the overall goals and motivation of the Canadian government for publishing these datasets: [1]`https://www.jeffjianzhao.com/papers/governor.pdf`, [2]`https://www.youtube.com/watch?v=7c0SifFA5v4`,[3]`https://openproceedings.org/2024/conf/edbt/paper-44.pdf`.
- 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.