Project Track 1

Section 1: Project Introduction

This project aims to provide an opportunity for students to apply relational database techniques to a domain of the student's choice. Students will be asked to create a relational (SQL) database-centric web-based application.

Students form teams of 4; the team formation process is outlined in the next section. Students are required to complete this group work in stages. We clarify the requirements for each stage in section 3. Project policies that are not listed in the course syllabus are listed in section 4. Please read the following sections carefully. Rubrics are also provided in each section. In addition, to clarify some questions you might have, we will keep an updated list of Q&A (grouped by stages) in section 4.

It is **strongly recommended** that you read the **entire** document so you have an idea of what is coming up.

Section 2: Team formation

Students should form teams of exactly four members (no more, no less). Only teams formed by the teaching staff may have 3 members. No team should have more than 4 members.

Students should register their teams by the stage 0 deadline by editing **the team signup sheet (https://docs.google.com/spreadsheets/d/11XM9GVC0w7ji1bIn77AsALGQjP0ZNwTow3JpM-Uzw14/edit#gid=0&range=A1)** . When putting your information into the boxes, please be **respectful of other students' submissions**. If students did not join a team by the stage 0 deadline, the course staff would randomly assign them to groups. The following are some team formation policies that students need to be aware of.

1. Each team must have 4 members, no more, no less. Groups must fill up spaces if a student drops the course before the team formation time elapses. Otherwise, the course staff will randomly assign students without a team to your group.
2. If you did not join a team by the Stage 0 deadline, do not worry, the course staff will randomly assign you to a team. You will receive your team information via email.
3. All teams must have a team captain. The team captain acts as the main point of contact between the team and the course staff.
4. Please feel free to reach out to course staff if you have any questions.

Section 3: Stages and Rubrics (Full mark: 100%)

In this project, you will develop a relational database-centric **web application**. We will use the terms "database" and "relational database" interchangeably in the remainder of the document.

Each project stage is due at 23:59 pm (CT) on the specified due date. **ALL SUBMISSIONS ARE COMPLETED ON GITHUB.** Deadlines are enforced strictly and identified by the commit time or when the project TA visits the GitHub repository after the deadline, whichever is earlier. Other deadline-related policies follow the syllabus. Detailed project submission requirements are listed for each stage. Please follow the submission instructions carefully.

Here is a high-level summary of what the stages would be:

Stage 0: Team Formation (2%)

Stage 1: Project description (8%)

Stage 2: Conceptual and Logical Database Design (15%)

Stage 3: Database design and Indexing (22%)

Stage 4: Midterm Demo (20%)

Stage 5: Final Demo (23%)

Stage 6: Report and video (10%)

Stage 0: Team Formation (2%)

For this stage, you will **form teams of four** and set up the **project repository on Github**.

1. The team captain must submit your team information by editing **the team signup sheet (https://docs.google.com/spreadsheets/d/11XM9GVC0w7ji1bIn77AsALGQjP0ZNwTow3JpM-Uzw14/edit#gid=0&range=A1)** before the deadline.
2. Check if you have the privilege to use the UIUC CS Github. Given the prerequisite of CS225, most students should have access to the **UIUC CS GitHub (https://github-dev.cs.illinois.edu/)**:
    1. [If you do not] you should login using your Illinois netID and password.
    2. Visit **CS411 Spring 2022 (https://github-dev.cs.illinois.edu/sp22-cs411)** Organization and verify that you **can** create new repositories. (The NEW button to the right of the image below)

1. Create a team repository that follows the sample. You MUST follow the naming conventions for TAs to clone your repository. The repository must be: **sp22-cs411-teamXXX-sampleName** where **teamXXX** is the assigned team ID with three digits and **sampleName** as your teamName. Please be wary that special characters might not be permitted as the URL.
2. Your repository should contain two markdowns: **Readme.md** and **TeamInfo.md**.You can find samples of these files in the **Sample repository (https://github-dev.cs.illinois.edu/sp22-cs411/sp22-cs411-team000-sample)**.
    1. Please follow the instructions and the format of both files. Reach out to your project TA for help if you need to.
3. When you are done, you should package your commit as a release. Please tag your releases correctly so project TAs can locate them. You can find instructions about publishing your release in the **Sample repository (https://github-dev.cs.illinois.edu/sp22-cs411/sp22-cs411-team000-sample)**. This information can also be found in **the dedicated Github Docs (https://docs.github.com/en/repositories/releasing-projects-on-github/about-releases)**. You can confirm your submission by looking at the **team signup sheet (https://docs.google.com/spreadsheets/d/11XM9GVC0w7ji1bIn77AsALGQjP0ZNwTow3JpM-**

Uzw14/edit#gid=0&range=A1) . If your release was correctly done, you should be able to access that exact submission next to your team ID on the **"stage submission" tab (https://docs.google.com/spreadsheets/d/11XM9GVC0w7ji1bIn77AsALGQjP0ZNwTow3JpM-Uzw14/edit#gid=569326282&range=A2)** , under the **Submission Link** column.

1. The team captain is responsible for keeping track of the progress of all members. The team captain will also coordinate work among different members. It is a great experience that teaches you how to manage a team.
2. Teams that are assigned by TAs will need to complete this stage within 72 hours of the original deadline. Team assignments will be sent out within 12 hours of the team formation deadline.

Rubric: This stage is worth 2%. You are graded by completeness. To be complete, you should have all the components listed below. Each missing component will result in a deduction of 1% with a minimum grade of 0. There are a total of 6 components in stage 0.

1. Project team signup (unless assigned by TAs), including team name
2. Create your project repository with the correct naming and repository URL
3. Create your project repository, in the right location, with the correct format
4. Create the **Readme.md** file
5. Create the **TeamInfo.md** file and make appropriate changes.
6. Create a release with the correct tag for your submission

Stage 1: Detailed Project Description (8%)

For this stage, your team is expected to develop the project idea and provide a detailed project description.

1. Each group should create a **doc** folder. You can submit your project description with a PDF or markdown file.
2. You should update your **TeamInfo.md** with the teamName and project summary.
3. You should create a project proposal with the following information:
    1. Project Title
    2. Project Summary: It should be a 1-2 paragraph description of what your project is.
    3. **Description** of an application of your choice. State as clearly as possible what you want to do. What problem do you want to solve, etc.?
    4. **Usefulness**. Explain as clearly as possible why your chosen application is useful. Make sure to answer the following questions: Are there any similar websites/applications out there? If so, what are they, and how is yours different?
    5. **Realness**. Describe what your data is and where you will get it.
    6. Description of the **functionality** that your website offers. This is where you talk about what the website delivers. Talk about how a user would interact with the application (i.e. things that one could create, delete, update, or search for). Read the requirements for stages 4 and 5 to see what other functionalities you want to provide to the users. You should include:
        1. Describe what data is stored in the database. (Where is the data from, what attributes and information would be stored?)
        2. What are the basic functions of your web application? (What can users of this website do? Which simple and complex features are there?)
        3. What would be a good creative component (function) that can improve the functionality of your application? (What is something cool that you want to include? How are you planning to achieve it?)

7. **A low fidelity UI mockup**: What do you imagine your final application's interface might look like? A PowerPoint slide or a pencil sketch on a piece of paper works!
8. **Project work distribution**: Who would be responsible for each of the tasks or subtasks?
   List of the person responsible for which exact functionalities in section f. Explain how backend systems will be distributed across members. Be as specific as possible as this could be part of the final peer evaluation metrics.

Rubric: This part is worth 8%. You are graded by effort and completeness. You should write as much detail as you can. Your Project TA might ask you to add additional information even after submitting your project proposal. Usually, **two or more paragraphs for each component** are recommended to achieve the number of details a TA needs to assess your project.

> If there are any remaining issues related to the repository creation and requirements from stage 0, there will be a 2% penalty to this stage. Otherwise, each missing component listed in this stage will result in a deduction of 1%. There are a total of 10 components in stage 1. The minimal grade you get for this stage is also 0.

1. Updates in **TeamInfo.md** for teamName
2. Updates in **TeamInfo.md** for project summary
3. Creation of the project proposal placed in the **doc** folder.
4. The project proposal includes the title and project summary
5. The project proposal includes a detailed description of your application
6. The project proposal includes a detailed usefulness description of your application:
7. The project proposal includes a detailed realness description of your application
8. The project proposal includes a detailed functionality description of your application (2%)
9. The project proposal includes a detailed low fidelity UI mockup of your application
10. The project proposal includes a detailed project work distribution across the team
11. Create a release with the correct tag for your submission

Stage 2: Conceptual and Logical Database Design (15%)

For this stage, your team is expected to provide the conceptual design (Entity-Relationship Diagram *or* UML Diagram) of the project database.

1. Create a single file, either using markdown or PDF, to submit your ER Diagram in the doc folder
2. You should have the ER/UML diagram for your application. You also need to provide a description of the assumptions you make for each entity and relationship. For example, "we think that there must be only 1 advisor for each student".
3. Your ER/UML diagram should satisfy the following requirements:
   1. Your project must involve at least 5 entities with **at most one** entity regarding user login information.
   2. The 5 entities listed above does not include any relationship tables
   3. It should involve at least two types of relationships (i.e., 1-1, 1-many, and many-many).
4. Convert your conceptual database design (ER/UML) to the logical design (relational schema)

Your relational schema should be formatted as follows:
Table-Name(Column1:Domain [PK], Column2:Domain [FK to table.column], Column3:Domain,...)

PK: Indicates that the column is a primary key for the table

FK: Indicates that the column is a foreign key referencing the primary key of table.column.

Domain: INT, Decimal, VARCHAR(X),....

Rubric: This stage is worth 15%. You are graded by completeness and correctness. There are several ways one can represent the same application. We will evaluate the correctness of your ER/UML design by the following metrics: 1) Have you represented your application entities and relationships correctly? And 2) does your ER/UML design match your application requirements and the assumptions you listed in the ER/UML description.

1. Does not have a submission located within the doc folder (-0.5%)
2. Having a complete and correct ER/UML diagram (+6%)
3. The ER/UML diagram has 5 or more entities (+2%)
4. Having a complete and correctly translated relational schema (+7%, -1% for each incorrect translation)
5. Assumptions of the ER/UML diagram (-0.5% for each missing description for each entity)
6. A description of each relationship and its cardinality (-1% for each missing description)
7. Create a release with the correct tag for your submission (-2% for incorrect release)
8. Fix all suggestions for the previous stages (-2% for missing requirements from previous stages)

Stage 3: Database Implementation and Indexing (22%)

Google Cloud Platform (GCP):

- This semester, we recommend using MySQL cloud database hosted on the **Google Cloud Platform** (MySQL@GCP), but that is not required.
- **We will email you your GCP account information before this stage.**
- **We will also host project workshops to help you set up your GCP account and your MySQL@GCP database.**

For this stage, your team is expected to implement the relational database, develop advanced SQL queries, and apply indexing techniques to optimize your queries.

1. Create a markdown or pdf called “Database Design” in the doc folder
2. When creating your MySQL instance, make sure you use MySQL ver.8 or above to use Explain Analysis
3. Implement the database locally or on **MySQL@GCP**. If on GCP, the database should be created in one of the team members' GCP accounts.

   Together, the team should:

   1. Implement at least four main tables (i.e., tables that include core application information, not auxiliary information, such as user profiles and login information).
   2. In the Database Design markdown or pdf, provide the Data Definition Language (DDL) commands you all used to create each of these tables in the database. Here’s the syntax of the CREATE TABLE DDL command:
      CREATE TABLE table_name (column1 datatype, column2 datatype, column3 datatype,...);
   3. Insert data to these tables. You should insert at least 1000 rows **each in** three of the tables. Try to use real data, but if you cannot find a good dataset for a particular table, you may use auto-generated data.

4. As a group, develop two advanced SQL queries related to the project that are different from one another. The two advance queries are expected to be part of your final application. The queries should **each** involve at least two of the following SQL concepts:
    - Join of multiple relations
    - Set operations
    - Aggregation via GROUP BY
    - Subqueries

5. Execute your advanced SQL queries and provide a screenshot of the top 15 rows of each query result (you can use the LIMIT clause to select the top 15 rows).

Indexing: As a team, for each advanced query:

1. Use the EXPLAIN ANALYZE command to measure your advanced query performance before adding indexes.
2. Explore adding different indices to different attributes on the advanced query. For each indexing design you try, use the EXPLAIN ANALYZE command to measure the query performance after adding the indices.
3. Report on the index design you all select and explain why you chose it, referencing the analysis you performed in (b).
4. Note that if you did not find any difference in your results, report that as well. Explain why you think this change in indexing did not bring a better effect to your query.

Rubric: This stage is worth 22%. You are graded by completeness and correctness. For completeness, each justification should be **at least one paragraph long**. The rubric is as follows:

1. Does not have a submission located within the doc folder (-0.5%)
2. Database implementation is worth 7% and is graded (as a group) as follows:
    1. +3% for implementing the database tables locally or on GCP, you should provide a screenshot of the connection (i.e. showing your terminal information)
    2. +2.5% for providing the DDL commands for your tables. (-0.5% for each mistake)
    3. +1.5% for inserting at least 1000 rows in the tables. (You should do a count query to show this, 1% for each table)
3. Advanced Queries are worth 7% and are graded (as a group) as follows:
    1. +5% for developing two advanced queries (see point 4 for this stage, 2.5% each)
    2. +2% for providing screenshots with the top 15 rows of the query results (1% each)
4. Indexing Analysis is worth 8% and is graded (as a group) as follows:
    1. +3% on trying at least three different indexing designs (excluding the default index) for each advanced query.
    2. +4% on the indexing analysis reports.
    3. +1% on the accuracy and thoroughness of the analyses.

Stage 4: Midterm Demo (20%)

In the previous stages, we designed all the database components for our application. Now it is time to connect your database to the front-end application. For this stage, your team is expected to build a simple user interface that interacts with the database you developed in Stage 3. Please note that we do not grade based on how fancy your user interface is. We grade based on the functionality of your interface and how it connects to the database.

As a team, you should create a simple user interface that performs CRUD (Create, Read, Update, Delete) and search operations on the table(s) you created in stage 3. You must implement each operation at least once (5+ operations: CRUD & Search), and they can each be on different tables. Try to make the interfaces and operations relevant to your application! The expected functions, along with their rubric, are listed below.

1. Insert new records (rows) to the database (3%):
    1. 0% if no interface for this operation is present
    2. +0.5% for having the code connecting the interface with the database
    3. + 1.5% for having the interface for insertion
    4. +1% for successfully inserting to the database
2. Search the database using a keyword search. Your application should allow the user to input their search keyword and return the result to the interface (5%):
    1. 0% if no interface for this operation is present
    2. +0.5% for having the code connecting the interface with the database
    3. +1.5 % for having a textbox for entering the search keyword
    4. +3% for the successful execution of the query and the returning the result to the interface
3. Update records on the database (3%):
    1. 0% if no interface for this operation is present
    2. +0.5% for having the code connecting the interface with the database
    3. + 1% for having the interface for update
    4. +1.5% for successfully updating the database
4. Delete rows from the database (3%):
    1. 0% if no interface for this operation is present
    2. +0.5% for having the code connecting the interface with the database
    3. + 1% for having the interface for deletion
    4. +1.5% for successfully deleting row(s) from the database
5. Integrate into your application both of the advanced SQL queries you developed in stage 3. (6%):
    1. 0% if no interface for this operation is present
    2. +0.5% for having the code connecting the interface with the database
    3. + 4% for having the interface for the advanced queries
    4. +1.5% for successfully executing the advanced queries

Each team will demo to their project TA by signing up on the midterm demo signup sheet. More details will be posted as the demos approaches. Only one member is required to be presented at the midterm demo. Please keep track of your demo time. The project TA can cut you off when time is up and not give you grades for components not presented during your demo. Missing your demo slot will result in a grade of 0 for the entire team. Other accommodations follow the course syllabus.

You must submit your code to the repository in order to receive the grade. You should also tag your release. Failure to tag your release will result in a 3% deduction.

Stage 5: Final Demo: Putting everything together (23%)

For this stage, your team is expected to develop advanced database functions and add a creative component to your application. **You are expected to continue working on the same (or an improved version of the)**

database you developed in Stage 3. You will also use this time to complete your application interface.

Your web application should be designed for your users, not for the teaching staff. So, make sure that your application functionality matches your intended application description (usefulness, realness, uniqueness,...).

1. A complete application with basic CRUD operations.
2. In addition to the previous requirements:
    1. Implement an advanced database program that is related to your application. An advanced database program should include transaction + trigger OR a stored procedure + trigger.
        1. Transaction requirements: A complete and functioning transaction with the correct and justified isolation level, involves at least two advanced queries, involves looping and control (e.g., IF statements) structures, and provides useful functionality to the application.
        2. Stored procedure requirements: A complete and functioning stored procedure, involves at least two advanced queries, uses cursors, involves looping and control (e.g., IF statements) structures, provides useful functionality to the application.
        3. Trigger requirements: A complete and functioning trigger, involving event, condition (IF statement), and action (Update, Insert or Delete), provides useful functionality to the application.
        4. Transactions, stored procedures, and triggers do not need to be related, as long as they are relevant to the application.
        5. Transactions, stored procedures, and triggers need to be implemented in SQL, not through Object-relational Mappers (ORMs).
        6. The conditions (i.e., the If statements) and the advanced database program needs to make sense to your application.
    2. Integrate your advanced database program with the application.
    3. This advanced database application must be triggered by some front-end interface/interaction.
3. For 1% of extra credit, the entire application should be hosted on GCP and connected to a MySQL instance on GCP.
4. **Extra Credit (1% of the entire project grade):** We would like your team to develop a **creative component (function)** that complements your project. This **creative function** can use existing libraries and/or APIs. It could be as simple as an interactive visualization tool, or as complex as a machine learning recommendation system.
5. A complete application demo. Think about what is the objective of your application? How would the user use the application? Discuss your system design choices (index selection, transaction, advance query, creative component)? Discuss future improvements or extensions.

Rubric: This stage is worth 23%. The entire stage would be graded during the demo. You are graded by completeness and correctness. The rubric is as follows:

1. Clear demo presentation. Arriving on time and completing the presentation within the given time frame (details will come at a later time on campuswire). (+1%)
2. A clear demo featuring a user's end-to-end process interacting with the system that involves presenting the CRUD operations, the advanced database program, and the (**optional**) creative function (10%)
3. The advanced database program contains at least a sophisticated transaction+trigger or a stored procedure+trigger (+12%)

    1. Project TAs may deduct up to 50% of this component if the transaction and/or trigger does not make sense to the application.
    2. If you implemented transaction+trigger (8%):
        1. Transaction requirements: A complete and functioning transaction with the correct and justified isolation level (2%), involves at least two advanced queries (2%, 1% each), involves looping and control (e.g., IF statements) structures (2%), and provides useful functionality to the application (2%).
        2. Trigger requirements (4%): A complete and functioning trigger (1.5%), involves event, condition (IF statement), and action (Update, Insert or Delete) (1.5%), provides useful functionality to the application (1%).
    3. If you implemented stored procedure+trigger (8%):
        1. Stored procedure requirements: A complete and functioning stored procedure (2%), involves at least two advanced queries (2%, 1% each), uses cursors, involves looping and control (e.g., IF statements) structures (2%), provides useful functionality to the application (2%).
        2. Trigger requirements: A complete and functioning trigger (1.5%), involves event, condition (IF statement), and action (Update, Insert or Delete) (1.5%), provides useful functionality to the application (1%).
4. **Extra credit:** The entire application is hosted on GCP AND connected to a MySQL database hosted on GCP. (+1% for hosting both application and database on GCP. Hosting both the application and database on other platforms, including but not limited to Azure and AWS, will receive this extra credit upon the project TA's approval.)
5. **Extra Credit:** A functioning and interesting creative component that is relevant to your application (up to +2% of the entire project grade)
6. During the demo, the team should also discuss the following points: (-1% for any missing point)
    1. Explain your choice for the advanced database program and how it is suitable for your application. For example, if you chose a stored procedure+trigger, explain how this choice is suitable for your application.
    2. How did the creative element add extra value to your application?
    3. How would you want to further improve your application? In terms of database design and system optimization?
    4. What were the challenges you faced when implementing and designing the application? How was it the same/different from the original design?
    5. If you were to include a NoSQL database, how would you incorporate it into your application?
7. You must submit your code to the repository in order to receive the grade. You should also tag your release. Failure to tag your release will result in a 3% deduction.
8. All members must be present at the final project demo. Members who did not attend the entire duration of the demo will receive a 0.

Stage 6: Project Report and Demo Video (10%)

For this stage, your team is expected to submit the final project deliverables, including a project reflection report and a video demo.

1. Your project reflection report should contain the following topics. The report would be graded by completeness and correctness.

1. Please list out changes in directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).
2. Discuss what you think your application achieved or failed to achieve regarding its usefulness.
3. Discuss if you changed the schema or source of the data for your application
4. Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?
5. Discuss what functionalities you added or removed. Why?
6. Explain how you think your advanced database programs complement your application.
7. Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.
8. Are there other things that changed comparing the final application with the original proposal?
9. Describe future work that you think, other than the interface, that the application can improve on
10. Describe the final division of labor and how well you managed teamwork.

2. The video should be between 2-5 minutes long (7 minutes is the absolute max). Make it fun. Think of it as your chance to advertise your project to investors who find it promising.

Rubric: This stage is worth 10%. You are graded by completeness and correctness. The rubric is as follows:

1. Each missing section of the report (-1%), if there are no changes or nothing to discuss, state it.
    1. The report is not well-written or does not provide enough details (-4%)
2. Missing video (-2%)
    1. Too short, too long video (-1%)
    2. The video does not provide a comprehensive overview of the application (-2%)
3. The report should be placed in the docs folder and a readme file containing the link to the video. (-1% if missing either component in the folder)
4. You should tag your release. Failure to tag your release will result in a 1% deduction.
5. Failure to submit your catme survey will result in a 1% deduction from the stage.

Section 4: Additional Project Policies

1. Fairness, contribution, and grading:

    The teaching staff cares a lot about fairness. By default, we assume all team members contribute equally. Staff will try to intervene in teams when they need help. Please reach out when you identify issues with your group. Yet, there are limitations to what we can do, which is why we take the following procedure to assure fairness in PT1 grading:

    I. All teams will submit a catme survey by the end of the semester that includes various questions regarding peer contribution.
    II. Based on catme responses, teams will be assessed by their project TA to identify if there were any issues within the team.
    III. All members would receive the same grade if the team had no issues.
    IV. If there are issues within the team, the project TA might reach out for more information. On a case-by-case basis, project TAs will look at **code commit records** on Github with other materials to determine if a penalty

would be given to students who contributed significantly less. However, TAs cannot raise grades for any individual members.

2. Github code commit:

   Github code commits are used as a last resort to understand team dynamics and contributions. The teaching staff encourages members to stay active on github. We also believe that this is a good chance for teams to get used to code management systems. Teams are also encouraged to use issues, wikis, teams, projects, and other git-supported tools to facilitate the project development process.