# SJSU CMPE 138 & CMPE 180B TERM PROJECT SPRING 2019

You and your teammates will demonstrate your mastery of the material taught in this course with a multi-phased term project. The project consists of a non-trivial relational database to support a particular activity, event, or organization, and a database application as front-end to access the database. The initial project phases will concentrate on the design and construction of the relational database; the final phases will add an application with sufficient power to update, retrieve, and display information from the database, and to allow new data to be added to the database.

## Objectives

- To reinforce and practice database queries and design concepts learned in class
- To follow good software engineering practices in design, implementation, testing and documentation
- To learn how to write good technical report and do good presentation
- To collaborate effectively in a team environment

## Team

Each team consists of **?** to **?** students – to be announced after the drop-class deadline. There is no restriction on who can team up with whom; graduate students can team up with undergraduates. Any deviation from the specified team size, such as a one-person team, is not allowed.

Each team should select a team lead to coordinate various team activities and submit team assignments and reports. All assignments should be submitted to Canvas on time.

Each team member should collaborate with each other and be a good team player to complete the term project.

## Schedule

Just like any homework assignment, you must be submitted each of the followings **on time**.

1. Team Formation: 11:59pm on 2/13
    - Email the list of team members and who the team lead is to kong.li@sjsu.edu with subject "CMPE138_TEAM", and CC to the rest of the team

      | lastName | firstName |
      |---|---|
      | Doe | John |
      | Jane | Smith |
      | … | … |

      Team lead: …
    - A unique team number n will then be assigned.
    - Each team becomes a "user group" in Canvas. Each team would need to complete several Term Project assignments, each of which is a group assignment in Canvas.
    - Set up Canvas notification if you haven't done so.
2. Project Proposal: 11:59pm on 2/25
    - Submit the following to the Canvas assignment "Project: Proposal".
        - CMPE138_TEAMn_PROPOSAL (.pdf, .doc, or .docx)
    - You have to re-submit the proposal until it is approved by the instructor, and must get the final approval no later than one week from the deadline. *Each resubmission must high-light the portions you modified.*
    - The proposal needs to be approved by the instructor in order to proceed.
3. ER or EER Diagram: 11:59pm on 3/11

- Submit the ERD or EERD to the Canvas assignment "Project: ER Diagram".
  - CMPE138_TEAMn_ERD (.pdf, .doc, or .docx)
- Upon rejection, you must re-submit the ER Diagram within one week until it is approved by the instructor. *Each resubmission must high-light and indicate the portions you modified either within ERD or by using comment on Canvas.*
- ER/EER Diagram needs to be approved by the instructor in order to proceed.

4. Final Report, source code, and presentation slides: 11:59pm on 5/1
   - Submit the following to the Canvas assignment "Project – Report/demo/slides".
     - CMPE138_TEAMn_REPORT (.pdf, .doc, or .docx)
     - CMPE138_TEAMn_SLIDES (.pdf, .ppt, or .pptx)
   - Include "SJSU CMPE 138 Spring 2019 TEAMn" as comment near the top (1st line, if possible) of each source file (.sql, .java, .py, .c, .cpp, .h, .js, .html, etc.) written by the team.
   - Organize all source code (DB app, SQL scripts, sample data, app log files) into the following hierarchy

     `CMPE138_TEAMn_SOURCES`

     `<DB-Application directory>`: a directory which includes DB app source tree

     `SQL`: a directory which includes SQL related scripts (table, view, stored procedure, index, sample data, etc.)

     `Log`: a directory which includes all app log files
   - Zip the above hierarchy and submit the following to the Canvas assignment "Project – Implementation/testing".
     - CMPE138_TEAMn_SOURCES.zip

**Note**

Each project must be original – *the project should be created by the team from scratch specifically for this course **ONLY***. Any copying/cheating activity is not allowed. Please refer to the section "POLICY ON CHEATING" in syllabus.

Failing to turn in project related assignments (report, slides, etc.) would result in penalty for the entire team.

If a team member does not participate in any team activities, other team members should notify the instructor immediately without delay, and this team member will receive 0 point for the entire project.

Each project-related assignment is a group assignment, though project grade for each member is individual.

Each team member **must** participate in coding (which is part of implementation) and **must** participate in presentation/report writing. No coding implies a very poor mark for "implementation/test".

**Project Details**

*The focus should be on the relational database side.* After all, this is a database course.

**Database**

Design the database based on your choice of relational database engine. You should include E/R diagram, tables, views, queries, DML, stored procedures, sample data, etc.

You should describe the purpose of each table/view and the meaning of each column with each table. You also should describe purposes of each query, stored procedure, etc.

The operational side of the database, done by queries/DML/stored procedures, should include

- administrative portion: by the database application on behalf of the site owner to add/update/delete records, load sample data, etc.
- end-user portion: invoked by the database application on behalf of end-user to interact with the database server.

For obvious security reasons, make sure you encrypt or hash any password before storing the data into tables. Any plain-text password is not allowed.

You also need to load sample data into your database. The loading can be done by INSERT or stored procedure.

Various database objects (tables, views, stored procedures, etc.) should be created manually via various SQL client tools before your database application connects to the database.

The database script (source code) should include creating various database objects, as well as loading sample data. (Creating database itself is optional.)

**Database application**

The database application provides user interface to access the database for administrative side and end user side. It can be based on any languages (e.g., Java, C#, C++, Python, Perl, PHP, etc) and/or any technologies (JDBC, ODBC, etc) and/or any frameworks. Note that you can**not** use any existing SQL client tool (MySQL Workbench, MS SQL Management Studio, PHPMyAdmin, etc.) as your database application; you could certainly use them during development.

The DB application can be as simple as a command line application or a GUI interface (Web-based or not).

Based on the user of the database application, the application (or database) should selectively enable certain functionalities including

- administrative portion: for the site owner to add/update/delete records, load sample data, etc.
- end-user portion: for the end-user to interact with the database server.

In addition to presenting data from DB (i.e., SELECT), the DB application must be able to perform modifications against the DB (i.e., INSERT, UPDATE, or DELETE).

If the application is multi-threaded, explain why, what, and how.

The DB application should have logging facility to record various operations (request, response, error, etc.) into text files. You could leverage from existing library such as log4j, log4c, etc.

**Project Proposal**

In less than 2 pages, describe the problem you try to resolve and propose a unique project topic – no duplicates. The proposal must include (but not limited to) the following sections:

- Project title
- Team #, team members
- Miniworld (high level) description, the purpose of this application/DB, and the intended users
- Objects and actors within the miniworld. Which actor can play which role(s) and how an actor interacts with others
- Specify the planned functionalities and operations for each actor. Please be as specific as possible.
- Include a few scenarios to explain exactly how each actor interacts with one another

- (optional) Entities within the miniworld, and attributes and associated data requirements (e.g., cardinality) for each entity

Please use the project proposal template as the starting point. If needed, you can then add any additional sections. Any proposal without mandatory sections will be rejected.

The instructor may reject the idea if it is too broad or too narrow. Upon rejection, you have to re-submit your proposal ASAP until it is approved, and you must get the final approval within one week from the original proposal deadline. Need to be approved by the instructor in order to proceed.

**Project Proposal Template**

**Title**: <…>

**Team** <?>

<Member Name1>
<Member Name2>
…

**Miniworld**

<High-level Description of the Miniworld in a few sentences.>

**Purpose of Application/database and Intended Users**

<What are the purpose or goal of this application/database? Who are the intended users of this application/database.>

**Objects/Actors/Roles**

<List all objects, actors, and roles played by each actor in the Miniworld.>
<What are the objects? e.g., department, project, etc.>
<Who are the actors? E.g., employee, dependents, etc.>
<Which role(s) are played by each actor? Employee can play as department manager, employee can play as supervisor and supervisee, department can play as project controlling department, etc.>
<optional: specify data requirements, make sure your data can support your planned functionality and operations>

**Planned functionality and operations**

<How does actor (under each role) and objects interact with one another? Which actor/role can do what?>
<List the planned functionality and planned operations for each role in list format. Please be as specific as possible.>

**Scenarios**

<real usage examples of your app. at least one example per role. use a few stories to demonstrate **how** your intended user is using this apps, **step by step**. In each step, the intended user leverages from one or more of planned functionality and operations. For example, John Doe browses through the web site and finds out the info is interesting. John then registers with the web site. Once he logins, he is now able to get more detailed info. He then specifies search criteria and finds out database book is in stock. He can then place an order, etc. Note this is the realization of your planned functionality and operations.>

**(optional) Data requirements**

<Based on the section "Planned Functionality and Operations", what are data requirements in the Miniworld? Entities? What are the attributes of each entity? Who must or may do what, corresponding data requirements such as cardinality, participation, etc.>

**ER or EER Diagram**

Design the preliminary ER or EER diagram of your project by utilizing any existing tools. Hand-drawing is not acceptable. Export the ERD/EERD from the software as an image (e.g., JPG, BMP, etc.) and then embed the image to CMPE138_TEAMn_ERD (.pdf, .doc, or .docx). Do NOT enclose ERD screenshot as it is not readable.
Need to be approved by the instructor in order to proceed.

**Final Report**

The final report in paragraph format (not list format) must include (but not limited to) the followings:

- Project title
- Team #, team members
- The choice of database project
- The choice of database engine, DB application technologies, frameworks, languages, DB access technology, etc.
- Final overall architecture: block/component diagram
- Final list of functionalities/operations
    - If different from the planned list, status of each planned one
    - Any missing functionalities and the status of each
- **Final major areas/components/tasks - done by which team member(s)** and completion date
- Final design of database portion
    - ER (or EER) diagram: annotate & highlight modifications, if any, from the approved ERD
    - Specification of each DB object (tables, columns) and its meaning/purpose
    - Functional dependencies of each table and normalization
    - Denormalization, if any (which ones and why)
    - The normal form (3NF, BCNF, etc.) of tables/relationship. Justify the reasons if any of them is below 3NF
    - Any explicit multi-statement DB transactions initiated from DB server side (e.g., initiated from stored procedure)
        - If so, show code snippet of multi-statement DB transactions
    - Any additional DB objects/concepts utilized (view, stored procedure, trigger, index, isolation, CC, etc.)
    - Source script (create DB objects, sample data, etc.) – separate file
    - Screenshots - Sample manual execution of SQL queries and/or stored procedures with result sets for important operations
- Final design of DB apps portion
    - Any specific functionality involving accessing more than one table (e.g., read t1 and then update t2)
        - any explicit multi-statement DB transaction initiated from DB apps side is used to implement such functionality
            - If so, show code snippet of multi-statement DB transactions
    - Source code – separate files
    - Screenshots - Sample App execution
    - Sample (text-based) application log files (not database transaction log file) – separate files
- Any major design decisions, trade-offs (and why)
- Any major modifications from the proposal, ERD, EERD and why
- Any unique designs you are proud of
- Test cases, and test plan execution

- Project postmortem
    - issues uncovered
    - implement something differently
    - potential improvements
    - etc.

**Final Presentation and Demo**

You should have slides (e.g., Power Point, PDF, etc.) to highlight your final report. Each team member should present his/her own areas/tasks. Keep in mind the focus should be on database server side.

After presentation, you should have *live demo* of your database and applications. Show your DB objects. Use DB application to run through important functionalities/operations (both administrative side and end user side).

The length of presentation/demo per team will be announced later.

**RDBMS**

You can choose any relational database system, such as

- MySql, GPL version (Windows, Mac, Linux)
    - Community Server: https://dev.mysql.com/downloads/mysql
        - MySql Installer for Windows includes everything in one package
    - Workbench: https://dev.mysql.com/downloads/workbench/
    - Connectors (e.g., JDBC, etc.): https://dev.mysql.com/downloads/connector/
- Microsoft SQL Server Express (Windows)
    - SQL Server Express: https://www.microsoft.com/en-us/sql-server/sql-server-editions-express
    - SQL Server Management Studio: https://msdn.microsoft.com/en-us/library/mt238290.aspx
    - SQL Server on Linux: https://docs.microsoft.com/en-us/sql/linux
- Oracle Database Express: http://www.oracle.com/technetwork/products/express-edition/downloads/index.html
- PostgreSQL: https://www.postgresql.org/download/
- Etc.

(All of the above are free.)

**Any ORM tool is not allowed**

You are **not** allowed to use any ORM (Object-to-Relational Mapping) tool nor tools generating SQL queries automatically, nor any other tools with similar functionality, such as (but not limited to)

- ORM and HQL in Hibernate
- ORM (i.e., ActiveRecord) in Ruby on Rail
- ORM in Django
- ORM and Core in SQLAlchemy
- ORM in peewee, PonyORM, SQLObject (for Python)

The bottom line is that you must manually map ERD/EERD constructs to DB tables, and must specify the original SQL queries (SELECT, INSERT, UPDATE, DELETE) directly in your DB applications, without relying on certain APIs that construct SQL queries behind the scene for you.

After all, the goal of this course is get one familiar with designing ERD, mapping ERD to tables, and formulating SQL queries.

**Project Ideas**

These are just some examples. You are highly encouraged to create your own topic.

- Ride share (Uber-like)
- Real Estate Business
    - Search properties
    - Renovate the properties (items to be fixed, contractors used, estimation, etc)
    - List the properties
        - Recent/similar sales in nearby areas
        - Features of the property (# of bedrooms, # of bathrooms, fireplace, pool, etc)
    - Open House, submit offers, accept offers
- Online Education
- Stock Exchange
    - Stock quote, event, news, e.g., http://www.nasdaq.com/
- Stock Brokerage Account
    - Market update, customers, place order/option/short, market update, e.g., www.schwab.com

The followings are a few sample projects from the past -- You cannot choose any of these topics nor anything close/related to these topics:

- Library system
- Online shopping/mall/clothing/car/etc.
- Movie/book/video/music/DVD/CD/tool/etc. rental/selling/exchange/store
- University/school system - student/grade/class/department/professor/etc.
- Hospital system – doctor/patient/appointment/room/nurse/etc.
- Restaurant order/management
- Job search
- Flight routing and reservation
- Event management (ticketmaster-like, calendar, etc.)
- Professional Network (LinkedIn-like)
- Social network (Facebook-like)
- Apartment/house leasing/rental (landlords/tenants/etc.)