SJSU CMPE 226 PROJECTS SPRING 2020

You and your teammates will demonstrate your mastery of the material taught in this course with two projects. The first project involves paper-study and in-class presentation. The second project consists of a multi-phased 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 phases for the second project 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. 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 projects.

Schedule

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

1. Team Formation: 11:59pm on 2/16
   - Email the list of team members and who the team lead is to kong.li@sjsu.edu with subject "CMPE226_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" on Canvas. Each team would need to complete several Project assignments, each of which is a group assignment in Canvas.
   - Set up Canvas notification if you haven't done so.
2. Project 2 Proposal: 11:59pm on 2/25
   - Submit the following to the Canvas assignment "Project 2: Proposal".
     - CMPE226_TEAMn_PROPOSAL (.pdf, .doc, or .docx)
   - You must re-submit the proposal until it is approved by the instructor. If a revision is not submitted within 5 days after the instructor gives you feedback, each team member receives 1 point deduction (of the project's report portion) and such deduction repeats for every 5 additional days. Failing to get the

final approval within 10 days from the deadline, each team member may receive further deduction. *Each resubmission must high-light the portions you modified.*
    - The proposal must be approved by the instructor (i.e., receive a "complete" grade on Canvas) in order to proceed.
3. Project 2 ER Diagram: 11:59pm on 3/8
    - Submit the ER diagram to the Canvas assignment "Project 2: ER Diagram".
        - CMPE226_TEAMn_ERD (.pdf, .doc, or .docx)
    - If the ERD is based on additional modifications to your approved proposal, include your latest proposal as part of the submission.
    - Your initial ERD submission earns up to 5% of the final class grade and the grade is final.
    - You must re-submit the ERD until it is approved by the instructor. If a revision is not submitted within 5 days after the instructor gives you feedback, each team member receives 1 point deduction (of the project's report portion) and such deduction repeats for every 5 additional days. Failing to get the final approval within 15 days from the deadline, each team member may receive further deduction. *Each resubmission must high-light the portions you modified.*
    - The ERD must be approved by the instructor in order to proceed.
4. Project 1 Report and presentation slides: 11:59pm on 4/21
    - Submit the followings to the Canvas assignment "Project 1 – Report/slides".
        - CMPE226_TEAMn_PAPER_REPORT (.pdf, .doc, or .docx)
        - CMPE226_TEAMn_PAPER_SLIDES (.pdf, .ppt, or .pptx)
5. Project 2 Report, source code, and presentation slides: 11:59pm on 5/3
    - Submit the followings to the Canvas assignment "Project 2 – Report/demo/slides".
        - CMPE226_TEAMn_REPORT (.pdf, .doc, or .docx)
        - CMPE226_TEAMn_SLIDES (.pdf, .ppt, or .pptx)
    - Include "SJSU CMPE 226 Fall 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

      `CMPE226_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 sample app log files
    - Zip the above hierarchy and submit the following to the Canvas assignment "Project 2 – Implementation/testing".
        - CMPE226_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. Refer to the section "Academic Integrity and Collaboration Policy" 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 1: paper-study and presentation

The instructor will assign a paper to each team to study. The team studies the paper as a group and *uses your own words* to write a *short* summary report in *less than three pages*. The summary report in paragraph format (not list format) includes

- Team #, team members, paper title
- the task to tackle or problem to resolve
- assumptions / operating environments / intended usage
- the design/implementation
- the result, design/implementation tradeoffs
- your own insight and/or your own criticism – use your own words; do not copy feature list from the paper
- references, if any

When you write the summary report, pay attention to the theme or the high-level picture of the paper and adopt a top down approach to summarize the paper. Do include the essential features but no need to dive into all tedious details – have a good balance between breadth and depth. You can assume readers of the summary report are your classmates who have not read the actual paper.

To avoid plagiarism, make sure you cite the reference [n] at proper places in the summary report.

Note we focus on quality instead of quantity – a lengthy summary does not translate to higher grade. A short summary with your own insight/criticism is far better one without.

Selected teams will present the paper similar to conducting a workshop with slides in class. The instructor will make presentation slides available on Canvas (under files/project 1 slides) to the entire class. The length of the presentation will be announced later.

Project 2 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 ER 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, views, stored procedure, and triggers (if any), 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.)

Additional requirements:

- The database application must utilize views and stored procedures, both of which are defined on the database server side. Instead of just DB operations, the body of stored procedure must include control flow (i.e., if, while, etc.) along with DB operations.

- The project must have multi-DBstatement transactions, that modify data, along with commit or rollback. A transaction can be initiated from either server side (e.g., stored procedure), or client side (i.e., DB application), and the initiation can be either implicit or explicit.

Database application

The database application provides user interface to access the database for administrative side and end user side. You must write your own database application which includes necessary SQL queries to perform various operations against the database you designed. You can**not** use any existing SQL clients (MySQL Workbench, MS SQL Management Studio, PHPMyAdmin, etc.) as your database application. During development, you can certainly leverage any existing SQL clients (MySQL Workbench, MS SQL Management Studio, PHPMyAdmin, etc.).

Your database application can be based on any languages (e.g., Java, C#, C++, Python, Perl, PHP, etc.) and any technologies (JDBC, ODBC, etc.) and any frameworks (Spring, Express.JS, Node.JS, etc.). Your database application must support "user login" operation and must store user password, in either hashed or encrypted form, to some database tables.

- Team size < 3: The database application can be as simple as a command line application or a GUI interface (Web-based or not).
- Team size >= 3: The database application must have a GUI interface (Web-based or not).

Except publicly-accessible portion, after a successful user login operation, the database application selectively enables user-specific (or role-specific) 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).

Any Object-to-Relational Mapping (ORM) tool is not allowed. See a latter section for details.

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

Any ORM tool is not allowed

For RDBMS portion of project 2, you are **not** allowed to use any Object-to-Relational Mapping (ORM) 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 (ActiveRecord, DataMapper, etc.) in Ruby and 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 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 to get you familiar with designing ERD, mapping ERD to tables, and formulating SQL queries.

Project 2 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
- Members working on proposal
- Miniworld (high level) description
- Purpose of this application/DB, and the intended users
- Objects/actors/roles in the miniworld in list format
- Planned functionalities and operations for each actor/role in list format. Be as specific as possible.
- Scenarios: a few stories illustrate exactly how each actor/role utilizes functionality/operations defined earlier to interacts with one another
- (optional) Data requirements for each object/role/actor in list format. You could even define entities, attributes, relationships, cardinality, participation, etc.

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. Need to be approved by the instructor in order to proceed.

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.

Project 2 Proposal Template

Title: <...>

Team <?>
<Member Name1>
<Member Name2>
...

Members working on proposal
<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 in list format>
<What are the objects? one descriptive sentence for each. 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.>
Object1: one-sentence description
Object2: one sentence description
Actor1: one sentence description

- Role11: one sentence description
- Role12: one sentence description

Actor2

- Role21: one sentence description

...

Planned functionality and operations

<List the planned functionality and planned operations for each actor/role in list format. Be as specific as possible.>
<How/when does an actor (under each role) and objects interact with one another? Which/where actor/role can perform which operations under which situation?>
Role1 (or actor1)

- ...
- ...

Role2 (or actor2)

- ...

...

Scenarios

<Real usage examples of your app. use a few stories to demonstrate how your intended user is using this apps, step by step, by leveraging one or more existing functionality and operations defined earlier. Must cover all roles/objects defined earlier. Do NOT invent any new functionality / 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", specify data requirements for each object/role/actor **in list format** in the Miniworld (see slides for example). You can further define entities, attributes, relationships, cardinalities, participations, etc.>

Project 2 ER Diagram

It contains the following mandatory sections:

- Project title
- Team #, team members

- Members working on ER diagram
- Data requirements in list format: based on functionality/operations in your proposal
- ER diagram: Design the preliminary ER diagram of your project by utilizing any existing tools. Hand-drawing is not acceptable. Export (not screenshot) the ERD as an image file and insert it as picture into CMPE138_TEAMn_ERD (.pdf, .doc, or .docx) so that ERD picture is viewable in the report.

Any ER Diagram without mandatory sections will be rejected. Need to be approved by the instructor in order to proceed.

Project 2 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 of each role/actor, **in list format**
  - If different from the planned list, status of each planned one
  - Any missing functionalities and the status of each

  For example,

  Role1

• Operation1	done
• Operation2	partially done – which part is not done
• Operation3	newly added, done

  Role2

• Operation1	done
• Operation2	not implemented

  …
- **Member's contributions: list project major areas/tasks, as detailed as possible, done by which team member(s)** and completion dates, **in list format by team member**. Areas/tasks include (but not limited to):
  - **Proposal, ERD** (explain the reasons if different from your proposal/ERD submissions)
  - **Report, slides**
  - **Implementation (break down to components), testing (break down to components)**

  For example,

  Member1
    - …
    - …

  Member 2
    - …
    - …

  …
- Final design of database portion
  - ER diagram: annotate & highlight modifications, if any, from the approved ERD
  - Schema diagram
  - Specification of each DB object (table, column, view, stored procedure, etc.) and its meaning/purpose
  - Functional dependencies of each table and normalization

        - The normal form (3NF, BCNF, etc.) of tables. Justify the reasons if any of them is below 3NF
        - Denormalization, if any (which ones and why)
    - User login password: hashed or encrypted? How is this done? In which column of which table? Include a screenshot showing the result set of the table(s) with password column
    - Any explicit multi-DBstatement DB transactions that modify data and are initiated from DB server side (i.e., initiated from stored procedure)
        - If so, show code snippet of multi-DBstatement DB transactions
    - Stored procedures and views: describe functionality for each and show code snippet
        - Screenshots – show code snippet that invokes the stored procedure(s) you defined
        - Screenshots – show code snippet that retries result set from the view(s) you defined
    - Any additional DB objects/concepts utilized (trigger, index, isolation, concurrency control, 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-DBstatement DB transaction that modify data and are initiated from DB apps side is used to implement such functionality
            - If so, show code snippet of multi-DBstatement 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 proposal, ERD and why
- Any unique designs you are proud of
- Functionality test cases and test plan execution
- Project postmortem
    - issues uncovered
    - implement something differently
    - potential improvements
    - etc.

Project 2 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.

Project 2 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://docs.microsoft.com/en-us/sql/ssms
    - SQL Server on Linux: https://docs.microsoft.com/en-us/sql/linux
    - Azure Data Studio (Linux/Mac/Windows): https://docs.microsoft.com/en-us/sql/azure-data-studio/
- 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.)

Project 2 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 – student, grade, class, section, department, professor, etc.
- Hospital – doctor, patient, appointment, room, nurse, etc.
- Banking – accounts/loan, customers, employees, ATM, deposit/withdraw, etc.
- Airport, plane, airport - flight, flight leg, customers, pilots, arrival/depart, seat, 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.)
- Any topic with ERD similar to ones from class or homework

Project 2 Extra Credit

In addition to relational database, your project also utilizes one or more NoSQL database(s), such as Cassandra, MongoDB, CouchDB, Couchbase, Riak, HBase, etc. You should leverage NoSQL under the situation where NoSQL is a better fit than RDBMS in the context of your project, and justify your usage of NoSQL (i.e., do not use NoSQL just for the sake of using NoSQL).

• Cassandra	http://cassandra.apache.org/
• MongoDB	https://www.mongodb.org/
• CouchDB	http://couchdb.apache.org/
• Couchbase	http://www.couchbase.com/
• Riak	http://basho.com/products/#riak
• HBase	http://hbase.apache.org/
• DynamoDB	https://aws.amazon.com/dynamodb
• Other NoSQL services from cloud providers	
• Etc.	

You are allowed to use Hibernate for NoSQL portion.

Project 1: List of papers {available on Canvas under files/papers}

F. Chang, J. Dean, S. Ghemawat, et al, “Bigtable: a distributed storage system for structured data.” Proc. Conf. OS Design and Implementation (OSDI06), pp. 205–218, 2006.
    http://research.google.com/archive/bigtable-osdi06.pdf

J. Corbett, J. Dean, M. Epstein, et al, “Spanner: Google’s Globally-Distributed Database.” Proceedings of OSDI 2012.
    http://research.google.com/archive/spanner-osdi2012.pdf
    white paper: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45855.pdf

G. DeCandia, D. Hastorun, M. Jampani, et al, “Dynamo: Amazon’s Highly Available Key-value Store.” Proceeding SOSP '07 Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles, Pages 205-220.
    http://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf

Doug Terry, “Replicated Data Consistency Explained Through Baseball.” Communications of the ACM, December 2013, Vol 56(12): pp. 82-89.
    https://www.microsoft.com/en-us/research/publication/replicated-data-consistency-explained-through-baseball-2/