Database Design Project: SV [Something] Registration System using LLM

This is a semester long project to give students the experience of developing database system and database application using a commercial database management system, developing tools and Google Cloud Platform (GCP). Table 1 shows the executive summary of this project by phases.

Table 1. Summary of database design term project by phases

Phase	Description	Output	Due
Phase I (15%)	**Project Initiation** ○ Problem Statement ○ Plan your project	Documentation	Sep 17
Phase II (25%)	**Conceptual/Logical Database Design** ○ ER, EER modeling ○ Relational modeling ○ Relational Algebra	Documentation	Oct 1
Phase III (15%)	**System Setup using GCP** ○ Install MySQL using Google Compute Engine ○ Install Web server using Google App Engine (if needed)	Implementation	Oct 15
Phase IV (25%)	**Physical Database Design/Implementation** ○ Create Table and other objects ○ Data Insert ○ Formulate the queries	SQL statements & Documentation	Oct 29
Phase V (20%)	**Application Develop and Demonstration** ○ Application implementation ○ Final Documentation ○ Demonstration	System & Documentation	Nov 19

Stages of the Projects

Phase I [15%]: Project Initiation

Step 1: Form Project Team

This is a group project (with 3 members) using Google Cloud Platform (GCP). This semester, each group will develop "SV **[something]** Registration System using LLM" based on RDBMS, specifically MySQL on GCP and Large Language Model (LLM) such as ChatGPT.

First, you need to find a partner for your group, then self-sign in "Project Group" on Canvas, before you submit Phase 1.

Here are some important requirements of your project:

- You should use RDBMS running on GCP Compute Engine for your back-end storage (e.g., MySQL, Postgres, or SQL Server). A local RDBMS, i.e., running on your computer or laptop, is NOT accepted for your final project.

- You can design your presentation layer (i.e., interface layer) based on your preference. Your system can be web-based system (i.e., based on Web Server) using Web Application server, or stand along system (i.e., directly connect to RDBMS using GUI). Another option can be App using smart phone. You need to decide and specify what system architecture you are using in this project, i.e., Web Application server and program language (e.g., Django for Python)
- In this semester, you are going to use LLM to implement Open SQL programing. The examples of LLM include ChatGPT, LLaMA, BARD, or Claude.
- Programming Language: Any, such as Python, C, C++, Java, ASP .Net, PHP, Perl, etc.

Step 2: Problem Statement

First, make the name of your application. For example, "SV Course Registration" or "SV Event Registration App", etc. You can select any topics that may be happened at SV campus. Please, think about creative and fun ideas of **[Something]**. Your creativity will be considered for Phase 1 grade.

Second, you identify the problems of your system. For examples:

- "SV **[something]** Registration" can support web-based **[something]** registration at SV campus,
- "SV **[something]** Registration" can be managed by registration department, such as registration, modify the event and room assignment
- Each group can sign-up the system and modify the profile information
- And more problems

Third, here is the list of requirements of your system:

- There are two types of users: "Admin" and "Normal" users.
    - Admin user: can manage entire your system, so can access any data. In addition, admin user can access daily/weekly reports.
    - Normal user can be "students", "faculty" or "staffs"
- Normal user can register the system, and then sign-in the system
- Normal user can modify profile information
- Admin can add/modify/remove any information in the system
- Admin can see the statistics of data in the system
- Admin user can send a message to a normal user regarding to any issue on the registration. The message should be pop-up when a user log-in
- And more …

In addition to the minimum requirements, each system should add at least 15 more requirements. It is also good idea to add more specific characteristics of your registration system, e.g., "Event registration", "Course Registration", and more…

Last, you should present the defined problems using a model tool, such as UML model or Use Case model.

Step 3: Formulate in English at least +15 realistic queries

The queries would be useful to somebody using the data. The queries must be realistic and you should have more than few complex queries. Note, the queries that the database system

must be able to answer ultimately determine what information needs to be maintained in the database. For example, you need not include the information about best classmate of students if no user of your database will be asking for that. On the other hand, some additional information, may be included in the database if you feel that somebody may be interested in it.

Step 4: Specify the assumption about the database in English

Here you talk about attributes, keys, the nature of relationships between entities, etc. Do not discuss something that is obvious (e.g., that a user can post several items). In addition, don't make too many simplifying assumptions.

[Submission guideline for Phase I]

You should submit the full and legible report with a cover page that includes your name, the title of the project, and abstraction (brief description of the project). The body of Phase I documentation should include sufficient detail to describe all steps of Phase I:

- (Step1) Project Information: name, member, and so on
- (Step2) Problem Statement: description of your "SV **[something]** Registration System" with designed software model (e.g., UML or Use Case model)
- (Step3) List at least 15 queries in English
- (Step4) Any assumptions

In conclusion, describe what you have learned from the project, and what you need for the better project output. Instructor may recommend modification or additions to the document submitted in Phase I. After revision phase I as recommended, in this phase, you can move to the next phase.

The name of your submitted documentation file should be your "Group#_phase1". You can use your Group number for #. For example, "Group1_phase1.doc". Then, you will submit your file to Canvas.

Phase II [25%]: Conceptual / Logical Database Design

Step 5: EER modeling

Based on the Step 1 ~ 4 in the previous phase, in phase II draw an Enhanced Entity Relationship (EER) diagram for the problem you choose. This EER modeling should adhere to the principles taught in the course. It should also use the notation taught in the lecture and exercise lab. Entity should have keys clearly identified and relationships should have the cardinality information. The resulting EER diagram should contain:

- At least 10 entities
- At least 10 relationships (1:1, 1:N, and N:M)
- Some attributes on relationships
- At least 1 Weak entity (you can add a weak entity on purpose if not exist)
- At least 1 Generalization/Specialization

Step 6: Creating Relations

Based on the EER diagram, you should create relations using the relational model in the lecture. Make sure that you need to identify the primary key and foreign keys in all the relations.

- Identify other keys (candidate keys) in your relations
- Identify functional dependencies in your application and list them

Step 7: Relational Algebra (Chapter 8)

Formulate the queries in Step 3 using relational algebra. If, at this point, you are unable to state the queries in relational algebra, this is probably because your database is not powerful enough. Then, you must go back to step 2 and see how the database can be improved.

[Submission guideline for Phase II]

You should submit the full and legible report with a cover page that includes Team name, names of all members, the title of the project, and abstraction (brief description of the project). The body of Phase I documentation should include sufficient detail to describe all steps of Phase I:

- (Step5) ER diagram of the database
- (Step6) Relational database schema (including diagram)
- (Step7) Give the relational algebra expressions for the queries

In conclusion, describe what you have learned from the project, and what you need for the better project output.

The name of your submitted documentation file should be your "Project#_phase2". You can use your Group number for #. For example, "Group1_phase2.doc". Then, you will submit your file to Canvas.

Phase III [15%]: System Setup using GCP

In this phase, students should setup database server and web server using GCP. You are going to create Google Compute Engine for the database server where your selected RDBMS should be installed. On the other hand, you are going to create Google App Engine for the web application server for the selected web application server.

Step 8: Creating Database Server

First, create a project on GCP, named "DB-GROUP#", e.g., "DB-GROUP1". Also, assign a billing account which is based on a coupon in the exercise lab.

For the database server, you can select any RDBMS including MySQL, Postgres, SQL Server. However, your RDBMS should be installed on GCP Compute Engine.

Step 9: Creating Web Application Server

For the Web Application Server, you can use Google App Engine. You can select any app frameworks such as Django, Flask, Spring and webapp2.

- If you are using Django, please refer Exercise Lab 9. You can conduct Lab 9 in advance (if not covered in the classroom yet).

- If you are using any others, you can install and setup your selected server or software as a testing.

[Submission guideline for Phase III]

After you finish Phase III, you can just share your project with TAs. And submit several screenshots with description for each step to prove your work.

The name of your submitted documentation file should be your "Project#_phase3". You can use your Group number for #. For example, "Group1_phase3.doc". Then, you will submit your file to Canvas.

Phase IV [25%]: Physical Database Design/Implementation

In this phase, students should implement the database tables from the normalized set of relations created in the previous phase. Sample data should be supplied for each table. This phase is called as Physical database design.

Step 10: Creating DDL Script **`Filename: dbDDL.sql`**

Make a file containing the SQL statements that create your entire database schema, named dbDDL.sql. This includes the tables with their constraints, view, indexes, triggers, and all other database objects if you have them.
To keep the project consistent, make sure you have at least 8 tables. Make sure you have the following database objects:

- At least 1 trigger,
- At least 1 function or procedure, and
- At least 1 view.

Step 11: Creating DML Script **`Filename: dbDML.sql`**

Make a file containing INSERT statements which populate the table created in Step 9, named dbDML.sql. This script will contain SQL commands to fill data in your data. Each table should have around 7 ~ 10 sample data. If needed, other DML statement, such UPDATE, and DELETE can be included here

Step 12: Creating Drop Script **`Filename: dbDROP.sql`**

Create a script that will drop all the objects you have created for your project including table, trigger, index, and etc.. This will be used to start from a clean state after some inserts and deletes have been added to your application to check the correctness of your queries. You should be able to clean everything through this script and re-create the database instance.

Step 13: Creating SQL Script **`Filename: dbSQL.sql`**

Create a script with queries from the relation algebra in Step 7, named dbSQL.sql. This script should contain at least 5 queries on your database. Use the comment facility in mysql script (starting a line with -- , or /* */) to write the English version of your query, followed by the SQL version of the query. Also show the expected output in the file. These queries need to satisfy the following:

- Should be at least join queries (some involving more than 2 relations)

- At least two of them should be aggregate queries including GROUP BY and HAVING clauses with ORDER BY clause as well
- At least one of them should have subquery (either inline view, nested or with)

The purpose of having you write these is to make you think about slightly complex scenarios on your database schema and have you write queries involving join, aggregation and nesting that you have learned in the class.

[Submission guideline for Phase IV]

Before you submit, you should test all four SQL script files using mysql.

Make a single zip file. The name of your submitted file should be your "Group#_phase4.zip". You can use your group number for #. For example, "Group1_phase4.zip". Then, you will submit your file to Canvas.

Phase V [20%]: Application Develop and Demonstration

Step 14: Application Development

In this phase, you will develop a front-end application that will interface with your DBMS at the backend. The user will interact with the DBMS only through this interface. The interface will have a menu-driven input through which all interaction with the DBMS is accomplished. The results will be displayed to the user as well.

Specifically, the application will include dynamic SQL and open SQL programming capabilities, allowing users to interact with the database using both predefined queries and natural language input. The application should be capable of converting user queries in plain English into SQL using a Large Language Model (LLM) and returning the appropriate results from the database.

Dynamic SQL Implementation (at least 50% of queries)

- **Description:** Implement dynamic SQL to allow the application to generate SQL queries on the fly based on user input or application logic.
- **Example Use Case:** A user selects multiple filter options (e.g., date range, category, status) from the UI, and the application generates a corresponding SQL query dynamically to retrieve the data.

Open SQL Programming (at least 30% of queries)

- **Description:** Integrate an LLM (e.g., GPT-4 or similar) to interpret user input in plain English and convert it into a valid SQL query. The application should process the user's natural language query, use the LLM to translate it into SQL, and then execute the SQL query against the database.
- **Example Use Case:** A user inputs a query like, "Show me all courses that I have taken in 2024," and the LLM converts this into an SQL query that retrieves the relevant data from the database.

The interface should also have an interactive chat-based assistance bot for each user category (Admin and Normal Users). This bot should be designed to assist logged-in users by translating natural language queries into SQL commands using the Large Language Model.

Chat bot should not only retrieve and display query results but should also facilitate real-time database updates when needed.

If you want to do it with as a web-based interface which can be accessed from anywhere as long as connected to internet.

Step 15: Project Demonstration

Sign up for a demo (A sign-up sheet will be made available at the appropriate time by instructor). Be prepared for the demo about 5 to 6 minutes. Your demo may include

- Introduction:
- Executive summary of the project
- Demonstration. If your application needs setting up on our machine, make sure it can be done in as less time as possible by automating it using compilation scripts etc. If you are using some specific web servers or tools, you need to demo on your own laptop.
- Conclusion

Step 16: Final Report and Project delivery

Prepare the final project report including:

- Cover page (Same as Phase I)
- Phase I documentation. You need to revise it if needed
- Phase II documentation. You need to revise it if needed
- Sources of SQL scripts created in Step 9 ~ 13
- System configuration of the project (usually system diagram)
- Sources of Applications generate in Step 14
- In conclusion, describe what you have learned from the project, and what you need for the better project output.

Put everything including documents, SQL file, and source files into one compressed file such as zip, tar, jar or rar. Then, submit it to Canvas.

Make a single zip file. The name of your submitted file should be your “Group#_Final.zip”. You can use your group number for #. For example, “Group1_Final.zip”. Then, you will submit your file to Canvas.