# Project: Sprint 3 - Database design and implementation

## General instructions:

This assignment must be performed in assigned project teams. You must update your project report by adding the sections relevant to the tasks below, using this (updated) [**project report template**](https://docs.google.com/document/d/1FHyQMjngGzt9kDd1qTOheFvQYS7qtCqW-FL3XW4_pEA/edit?usp=sharing).

Do not jump to the implementation or even the logical design and then reverse engineer your conceptual design! Work in the order specified.

Ensure that all entities/attributes/tables/columns are named in a meaningful way, using a consistent convention.

### *Sprint activities:*

### Part 1: Refine requirements [*try to complete this within the first 2-3 days of the Sprint*]

* If you still have user stories left to complete, choose a small subset of them to be included in Sprint 3.
* Refine your chosen subset of user stories as before.

See [project report template](https://docs.google.com/document/d/1FHyQMjngGzt9kDd1qTOheFvQYS7qtCqW-FL3XW4_pEA/edit?usp=sharing) for format.

### Part 2: Perform conceptual design

Derive an updated conceptual design to accommodate the stories to be considered in this sprint and/or correct issues from previous sprints. You may change existing components of your original conceptual design or add new components. As part of this, do the following:

* Concretize the list of entities. For each entity, identify attributes and determine their types (i.e., simple / composite, single-valued / multi-valued, stored / derived). Some entities may have natural primary keys. Identify such primary keys.
* Concretize relationships among the entities identified above. For each relationship, identify cardinality and participation constraints. Only include relationships that will require some form of representation in your database.

See [project report template](https://docs.google.com/document/d/1FHyQMjngGzt9kDd1qTOheFvQYS7qtCqW-FL3XW4_pEA/edit?usp=sharing) for format.

### Part 3: Perform logical design, with normal form identification

Update your logical design (i.e., relational model) to match the updated conceptual design derived in [Part 2](#_6k50oc4b9q9g) above. As part of this, do the following:

* Map all the entities (both strong and weak), attributes and relationships that you have identified so far into relational tables.
* For each table
  + indicate the **primary key**;
  + indicate all **foreign keys** and mention what attribute of which table each foreign key refers to.
* In situations where you have multiple options for mapping (e.g., foreign key approach versus cross-reference approach), briefly **justify** your choice.
* For every new or updated table in your logical design, derive and specify the highest level of normalization. If the level is below fourth normal form (4NF), either normalize the table or justify why you choose to leave the table below 4NF. *You are not required to check for normalization levels higher than 4NF*. [**This**](http://www.rlvision.com/blog/method-for-determining-candidate-keys-and-highest-normal-form-of-a-relation-based-on-functional-dependencies/) resource may be useful to you when determining what the highest level of normalization is.

### Part 4: Indexes

For each table in your logical design, identify what indexes should be created (*material on indexes will be covered in the week of 11/19, so wait until then to work on this part*). For each index that you choose to create:

* Specify whether the index will be **clustered** or **nonclustered**.
* Briefly justify your choice. Since indexes are primarily used to speed up query performance, index design is done based on expected queries. So, in your justification, you must essentially explain what types of queries are common/most likely on a particular table/set of tables and justify your index choices in that context.

Add index details to your **logical design** using the format shown in the [project report template](https://docs.google.com/document/d/1FHyQMjngGzt9kDd1qTOheFvQYS7qtCqW-FL3XW4_pEA/edit?usp=sharing).

### Part 5: Stored programs and views

As you have already seen from [this discussion](http://www.mysqltutorial.org/introduction-sql-views.aspx) on views, views present several advantages (simplify complex queries, limit data access, provide security, etc.).

As you have also learned, it is possible to use stored programs (i.e., stored procedures, stored functions, triggers and events) to perform some repetitive / automatic / scheduled operations on your database.

As part of this sprint, you must identify and write **views** and **stored programs** that are relevant to your application. *Note: in Sprints 1 and 2, you were asked to write key SQL queries. Consider converting these queries into views or stored programs* ***when relevant***.

Demonstrate the working of your views/stored programs from your user interface (*a simple button to trigger the view/stored program is sufficient - there is no need to implement actual features*).

See [project report template](https://docs.google.com/document/d/1FHyQMjngGzt9kDd1qTOheFvQYS7qtCqW-FL3XW4_pEA/edit?usp=sharing) for format.

### Part 6: Implement and deploy database

Modify your database to reflect all changes made in this sprint. In other words:

* Create (or update) new (or existing) tables according to your updated, normalized logical design.
* In all tables (old and new):
  + ensure that appropriate data types are chosen for all columns;
  + identify and set up any NOT NULL and UNIQUE constraints;
  + set up its primary key;
  + set up all foreign keys;
  + create all the indexes identified.
* Populate your tables with relevant, meaningful data, updating old data as needed.
* Create all your views and stored programs within your database.

Note: As before, **do** **not** attempt to manually update your Sprint 2 database. Instead,

* export your Sprint 2 schema as a SQL script;
* make updates to the script to reflect the updated tables/data;
* create a new Sprint 3 schema;
* import your updated script into this new Sprint 3 schema;
* add views/stored programs as needed;
* add indexes as needed.

## Submission:

Create a **compressed** (.zip) file containing the following:

* your updated project report, adhering to the updated [project report template](https://docs.google.com/document/d/1FHyQMjngGzt9kDd1qTOheFvQYS7qtCqW-FL3XW4_pEA/edit?usp=sharing);
* a MySQL dump of your database.

**One** member of each project team must submit the zip file via **Canvas**.