**CSCI 5333.4 DBMS HW #1 Grading notes**

**Some issues and observations:**

* Missing multiplicity.
* Incorrect multiplicity.
* Good to use a CASE tool, such as Visual Paradigm but the watermarks make it difficult to read. Why not just use Rationale Rose?
* Some students copied the question and generic descriptions of what class diagrams and use case diagrams are into the answer. They may think that longer answers may look better. This is a big misconception. It is much more difficult to provide good short answer than copying and pasting contents that do not belong. Worse, copying and pasting without proper attribution is considered cheating because it gives the impression that the content is your own work which in fact it is not.
* For data modeling for database, focus should be paid on persistent data. It is not necessary to have a class for a report and a page, for example. That will be for the view part in the MVC model, for example. There may be many additional reports and pages. You only need to ensure that the captured data are sufficient to support these views. In later phases of modeling, such as when you need to actually implement these reports, pages, input forms, etc., you may then need to have implementation classes.
* Do not have too many use cases in a top level use case diagram.
* Many actors can share the same use cases, such as login, check work history, etc. (They may have different privileges which may not need to be captured in a top level diagram.) Another example: two use cases: login and logout; three use cases: create user account, create technician account, create administrator account.
* Document at least your actors, use cases and classes.
* Not all functions need to be captured in a top level use case diagram, only major ones. The focus should be on data, not operations. Operations are not as important as data in this phase of data model.
* Use cases are actions (sentences). Examples of inappropriate use case names:
  + Remaining technicians
  + User information
  + User already exists
  + Problem description
  + Problem comment
  + Problem specification
  + Request submission
  + Technician\_Expertise
  + Pending\_Work
* Avoid generic description. E.g.
  + Store information
  + performOperation() – Various operations are preformed assigned to the Requester.
  + Store inf: this will store the information.
  + Associated with
  + Is Associated
* Association names or role names are needed when the nature of the association is not obvious. (e.g. supergroup and subgroup; head and secretary, etc.)
* When you model implementation classes (such as Java or C#), the focus is on the operations (as most data members should be private and hidden. On the other hand, when you perform data modeling for databases, the focus is on the data members required. Operations are still useful, but they are used to support data requirement.
* Many modeled implementation classes: classes that use database access to get data to implement certain applications (such as providing a view page or a report, displaying a form, assigning a technician to a work, etc.) They are needed in the later phases of modeling. Focusing too much on them obscure data requirements.

**Examples of problematic modeling:**

* No class for work request. No class for computer. No class for user.
* Missing associations, eg. Between requests and report,, between requests and problem, etc.
* Include classes that are not needed, e.g. Request History, Request Page, Output Page., New User, New Technician, Request Information, Request Session, Support Page.
* Include associations that are not parts of the systems: submit request between users and technicans; administrator ‘checks’ requests, administrator “assigns work” to technician, technician (0..\*) send work order to Computer (1), etc.
* Firstname, lastname, SS#, phone stored in the general user class as well as requester, technician and administrator.
* Model software as an attribute of technician.
* Model “new user” and “new technician” as subclasses of “administrator”
* Model expertise as a class and include an associate ‘stores” between the administrator and expertise.
* One to one association between “user” and “technician” with the name “send request order”.
* Groupid and grouplocatoin as attributes of the computer class.
* Uses of generic and ambiguous terms: e.g. info, uniq\_id, System details class, attributes computer details and user details.
* Add <<PK>> stereotype to SSN.