* Entity-Relationship (E-R) Diagram of the complete database scheme
* Lucid description of the relational database scheme of the social networking site database, including a discussion of the reasoning behind your design decisions. Make clear how your design supports efficient query processing.
* A list of all functional dependencies in the relational database scheme
* Description of integrity constraints, including referential integrity

You will also be required to submit a *Users Guide* that carefully explains how to use all aspects of the system. It should be understandable by non-computer experts. Be sure that the user interface (screen design, menu structure, etc.) is clearly explained.

**E-R Diagram**

The preferences attribute set is tied to the user, not to the account. Here, the account means the credit card account. A user can have several associated credit card accounts, with each account having its own transaction history. However, a physical person is not supposed to register with more than one user name, because each person will have a unique SSN. We did not build any table that maps from one SSN to several usernames. Further, the same user name will be associated with the same person regardless of the role they play.

**Relational Database Scheme/Design Process for Efficient Query**

Even though the demo data for the account section included many more attributes, we parsed it out to only have two columns. Similar things was done with the User table. The reason was the follow reduce redundant functional dependency to avoid update anomaly and deletion anomaly. We know that if we make irrelevant attributes such as preferences attributes sets as part of the User primary key, we run into problems when any of the preferences field is null. Further, every time we want to update any user information, all of the entries in the User table with a different preferences has to be update as well. If repetitive functional dependency was not avoided, then it would lead to significant query inefficiency.

It’s also worthy to note that Stony Brook’s SQL host connection only allows up to about 20 connections simultaneously. Therefore, our project must be especially sensitive to the load it puts on query transaction.

As for the security aspects, URL parsing has been one of the hardest attack vector to defend against. Suppose our main URL is [www.royalhuangs.com/](http://www.royalhuangs.com/), then to support custom URL with username and circle name appended to the domain part, the string length and character combination can be infinitely long and varied. We decided to now allow custom URL for different users, circle, or posts, so our server is not burdened with distinguishing arbitrarily long, random URL versus malicious path. Placing slight limit on functionality end up increasing our security significantly.

**Functional Dependency**

**Referential Integrity Constraints**

Primary key of Person is actually personId which has default increment. Since User may not want to enter his or her SSN right away when they register, and User is created before all of the necessary information for being a Person. Thus Person actually uses foreign key references from User table. Both PersonId and SSN will be unique in the Person table.

Do we need to name the specific functional dependency types?

**User Guide**

**Registration**

At the top level, all information associated with the Manager will be entered by the database administrator directly through SQL. Employee will register through the same interface as other facebook users. If they are given with employee privilege by their manager, then they will also have an extra tab that allows them to switch between Facebook user and Employee interface within the same login.