# Final Project: Database-Driven Application

Due: Thursday, December 17 @ 11:59PM

Total Possible Points: 80

## How to Submit

* Moodle assignment (no email accepted)
* Submit the **final draft of your Requirements Specification document**, use cases, class diagram, and ER diagram electronically.
  + Either include the use cases, class diagram, and ER diagram in sections 4-6 of the Requirements Specification document,
  + OR, submit the aforementioned items in separate documents with clear references in sections 4-6 of the Requirements Specification document
* Submit your C++ code as a **Qt Creator project** in a ZIP/RAR file.

## Goals

* To develop clear, concise documents detailing all requirements gathered for an information system
* To analyze and design a 3-tier, database-driven application based on gathered requirements.
* To implement an object-oriented, GUI-based asset manager application based on your design documents.
* To make and document revisions to an application: iteratively clarifying requirements, redesigning models, and/or using Git to manage versions of your application code.

## Your Task

You and your partner(s) will finish analyzing, designing, implementing and documenting, your chosen database-driven application that you started in Assignment 3. The draft of the Requirements Specification you signed off is your guide. Use the assignments and labs to help you with the design and implementation of the various parts of your system.

## Requirements/Grading

Please submit the following documents in an electronic format that I will be able to open – Microsoft Word, Microsoft Visio.

### Requirements Specification with Use Cases and Models (35 points)

*Grading criteria follows assignments 3a and 3b, plus all requested revisions. Refer to the assignment details and ALL feedback from your instructor.*

1. Requirements Specification (**15 points total**)
   1. Includes updated cover page and table of contents
   2. Includes all seven (7) sections with complete requirements and all requested revisions made
   3. Includes Appendices A & B only with definitions and complete references
   4. *Formatting*: consistency throughout document (font, spacing, etc.)
   5. *Proofread*: checked for spelling and grammar
2. **Use Cases** for functional requirements: **(7 points total)**
   1. Minimum two (2) use cases
   2. Required use cases:
      1. *Course Registration App*: Add Course, Drop Course
      2. *Secure Chat App*: Create Chat, Message User in Chat
   3. Includes all requested revisions to use case content or styles:
      1. Adheres to the use case styles presented in class
      2. Identifies primary user, preconditions, postconditions, and one Alternative Flow
      3. Steps in Normal Course/Main Success Scenario are clear, complete & avoid technology-specific jargon
3. A complete **Class Diagram** depicting all classes you defined **(7 points total)**
   1. Identifies critical objects (classes) in domain
   2. Includes key attributes, visibilities, and multiplicities
   3. Includes critical methods per class
   4. Includes all requested corrections
   5. Class diagram and implemented C++ classes match
4. A complete **Entity-Relationship Diagram** depicting all data model entities, their attributes, and relationships. **(6 points)**
   1. Identifies critical entities in data model with sensible relationships
   2. Includes important fields (names and types)
   3. Includes well-chosen primary keys, foreign keys, and cardinalities.
   4. Includes all requested corrections
   5. ER diagram and implemented database tables match

### GUI Programming & Database (45 points)

1. Database (**10 points total**)
   1. Accessible via an Azure-based SQL Server or provided SQL script. (*3 points*)
   2. Accessible via a restricted, non-administrator account with CRUD privileges account (*2 points*)
   3. Includes all relevant tables (*5 points*)
2. GUI Layout (**10 points total**)
   1. *Widgets* (*5 points*) – Includes the following
      1. a (minimal) menu bar
      2. appropriate buttons to perform tasks
      3. large tabular or text display area to display key information
      4. a list box, or drop-down menu to select/filter specific items (Course Registration system only)
   2. *Usability* (*5 points*) – Layout is easy to understand; buttons use clear labels and/or icons.
3. Classes & Functionality (**25 points total**)
   1. Menu bar– at a minimum, includes a File menu with an Exit menu item. (*2 points*)
   2. File I/O (*3 points*)
      1. Database connection string stored in a text file in project folder
      2. Program reads in connection string from file.
   3. Implementation of functional requirements (*20 points sub-total*)
      1. Application asks for password; uses hash function to compare with input with stored, hashed password. (*2 points*)
      2. Displays all relevant application objects (courses, students, messages, users, chats, etc.) (*5 points*)
      3. Objects may be added, updated, and/or removed, according to requirements. (*5 points*)
      4. All buttons and menu items include event handlers (signals -> slots) to accomplish some sub-task: e.g., adding/editing a stock list. (*3 points*)
      5. Implements all requirements bug-free (*5 points*)