**Final Project Guidelines - .NET II**

Build a data-driven application with the following features:

***Planning:***

* \*A Statement of Work
* \*An ER Diagram for your database   
  (11+ tables including those for users)
* \*Data Dictionary
* \*Set of UI Drawings
* \*List of User Stories

***Database***

* \*A SQL script and batch file for deploying your database
* \*Sample data in the script
* \*Stored procedures for all data mutation operations and queries

***Solution:***

* \*n-tier architecture   
  (presentation, logic and data access layers at a minimum,  
  with a data objects project)
* \*At least four projects in the solution
* \*References between projects set properly

***Application:***

* \*WPF front end is preferred
* \*Support at least two user types with login preferred
* \*Screens for basic data operations CRUD

Theme/Topic

* This is up to you.
* Pick something fun if you like
* Useful projects are often best

***You can make an application to do whatever you like***:

* manage a collection
* record transactions
* track schedules
* manage projects
* organize a club
* anything that uses data

**Base Points:**

***Required to avoid deductions (gets you to a B):***

* \*Four projects
* \*Reasonable division of classes
* \*Program builds and runs
* \*Results correct / database updates as expected
* \*Database script works
* \*Features work

***Quality Plusses (can take your grade to an A):***

* \*App: well-conceived and user tested application
* \*Unit tests for your logic layer
* \*Code to interface, not implementation
* Dependency Inversion
* \*Data: good data types in app, DTOs, normalized tables
* Code: Well-written, well commented, good method names
* \*UI: UX focus, looks good, inputs constrained

***Quality Deductions (can lower your grade):***

* Poor naming conventions used
* Copy/Paste code
* Un-validated inputs
* Unsafe code (try/catches missing)
* Unlabeled/confusing inputs
* Poor coding style

**Advice:**

* Use a use case or user story driven approach.
* Design your user interfaces on paper or in Pencil or Visio to make them easier to build.
* Design your database carefully, but don't obsess over it.
* Add one feature at a time.  That way, if you run out of time, you always have something working.
* Sort your user stories into three categories:  
  1) MVP (minimum viable product – the most needed features)  
  2) Desirable features  
  3) Nice to add extras--if time allows
* Write stored procedures as you need them.  Don't spend all your time on the database and forget to build the app. Your grade is based on the number and quality of deliverable features (that means those that are working full stack from the user interface).
* Complete each feature (user story) as you build it.  Don't expect to have time to go back and fix things like missing input validation, bad UI labels or broken code.
* Test as you go so you won’t have a lot of debugging.
* Ask questions and be there on project days.
* Take this project seriously.  This should be a portfolio piece you can link to your resume. Employers will want to see it.