CS 422/522 Software Methodologies
Project Evaluation Criteria

1. Project Delivery

A designated team member must deliver the project; this person will deposit a single .zip file containing all project documents and files into Canvas before the project's due date and time.

A project may receive lower grades than it should because it is not clear where to find some of the team's work products, work products are missing, or it is not clear exactly how to run and use the application. Please ensure that every team member reviews the final .zip before submitting it. Although the following checklist is not complete, it may help avoid some of the more apparent errors:

1. **Completeness**: Make sure the team completes every piece of the project. Verify there is a final sign-off on each work product by the writer and a second person acting as a reviewer.
2. **Consistency**: Where documents depend on one another, have the documents reviewed for consistency. For example, the User's Guide must be consistent with the actual code delivered and the requirements.
3. **Application**: You must provide all the instructions and links necessary for anyone to set up (if needed) and run your application. I suggest testing this on users who are outsiders to your project. The teaching staff cannot award points for code that they cannot run and will not have time to try to debug code or installation instructions.
4. **Code**: Your final code submission must provide instructions on how to run the code. Provide tutorial-like examples contained in scripts or notebooks. Provide a README or equivalent document describing the code structure; document your code. In particular, be sure that any code you have used from other sources (not entirely written by yourself) is **clearly marked** to avoid confusion over plagiarism issues.

2. Grading Overview

The course project will be graded based on 100 points, distributed to the project components as described in Section 3. Note that the total number of points in Section 3 is 105.

A problem (e.g., inadequate documentation or a program bug) can cost points in multiple categories. For example, suppose the documentation is not adequate for using a feature. In that case, you will not get credit for implementing or documenting a feature that does not work (you may instead lose points for inaccurate documentation). Likewise, if you implemented a feature, but the grader cannot use it because the program crashes, you will not get credit for implementing it.

3. [15] - Initial Project Plan (SRS & SDS)

When evaluating these initial deliverable products, the grader will assign 0 to 1 point to each of the following fifteen bullets based on how well each was completed.

Project Plan

- A management plan. How is your team organized? How is the work divided among team members? How does your team make decisions? How will your team meet, and how will it communicate?
- Work breakdown schedule (with > 10 milestones) and project schedule (who will do what).
- Monitoring and reporting: You must monitor individual and project progress to track who did what and when they did it.
- A building plan. What is the sequence of steps you will take to build the system? When will each "build" of the system take place?
- A rationale for the plan. Why have you broken the system into these parts, and why did you choose those steps? What are your risks and your risk reduction strategies?

SRS

- Problem statement – Clear and concisely, what problem should be solved? The problem, including the task requirements, must be described independently of the solution (the software you will build).
- Description of users – Who are the users? What do they expect? What do they know already? What are their current technology usage patterns?
- Scenarios or Use Cases – The document must include three specific, different, realistic scenarios.
- Detailed description of requirements – Expand the problem statement and generalize from the use cases. Include at least 10 specific and measurable requirements, of which at least 6 must be non-functional.
- Split the functional and non-functional requirements between *essential* and *non-essential* ones. Identify a reasonable number of *essential* requirements that are well-specified and attainable.

SDS

- A description of the product you intend to build. This document must describe your product's externally visible behavior as precisely as possible but should be concise and clear.
- An overall design description. What are the significant parts of your system, and how do they fit together? Provide an architectural diagram (this is a static diagram). The system structure must be clear.
- Explain each major subsystem using a separate static model and dynamic model. All diagrams must be clear and understandable.

- Design an underlying Database where the system will store information. Specify the design using an ER diagram. The database must be in at least the third normal form (see https://en.wikipedia.org/wiki/Database_normalization).
- Design Rationale. Why did you choose this particular design? What main organizing principles did you use to break your system into parts?

4. [40] - Application Quality

Application quality includes features used by all user categories (perhaps even administrative users, such as sysadmins and end-users). Functionality must also provide performance and scalability.

Robustness: 10 points

This aspect evaluates the extent to which the system executes without bugs or errors, such as failing to install and compile, behaving differently than specified, or crashing.

Feature Set: 10 points

This aspect evaluates the extent to which the system exceeded, met, or fell short of requirements. The feature set considers requirements that were assigned and specified by the team. We evaluate whether essential features were missing or provided.

Ease of Use: 10 points

This category includes setup (if any) and usability for end-users and administrators. It is typically distinct from documentation quality but may include the presence and usefulness of online help. Usability considerations for administrative users may differ from usability considerations for end-users; this aspect considers both. For a prototype, this aspect addresses both existing and planned features.

Ease of use is subjective. Remember that new users (i.e., clients or us) expect to encounter a system that parallels the current world by establishing a clear metaphor. Those concepts must be evident in the user interface.

User documentation, including installation and setup: 10 points

This aspect evaluates the extent to which documentation clearly and accurately describes, for an end-user, how to accomplish real-world tasks using the software, including how to install and set up the system. The documentation must be clear, accurate, well-written, well-organized, and complete.

5. [50] - Organization, Planning, and Technical Documentation

Technical documentation and system organization are evaluated together. Good documentation can make a good design evident, and poor documentation can doom an excellent design to degradation over time, but good documentation cannot compensate for poor design.

Project Plan: 10 points

This section evaluates how effectively the team performs project planning, how well it followed the plan, how well the team adapts it to inevitable hiccups, and whether risks are effectively considered, mitigated, and re-evaluated when plans change.

A report must indicate who did what, when they did it, and how long everyone spent on each of their tasks (assigned date and completed date, as well as time on task). Teams must keep a clear record of meetings and outcomes (i.e., when meetings occurred and what was accomplished and agreed upon at each meeting.) A series of continually updated project plans show the status of significant project milestones and deliverables at regular intervals during the project lifecycle.

Systems Requirements (SRS) - 10 points

This part must include all elements described in the initial SRS submission, updated to reflect your implementation. This document must consist of a clear, complete, and well-organized description of requirements, a rationale for the project includes, defers for future versions, and (as appropriate) excludes. The document must provide a user-centric specification and a precise technical specification. Prospective developers should be able to use the report for system creation or maintenance.

System Architecture and Design documentation - 10 points

The software design documentation must include the following parts:

- Software Architecture: This section describes the software structure by answering the following questions for a reader:
    - How is the software decomposed into components?
    - How do the components work together to implement the essential application features?
- Why was this particular design chosen? What is the rationale for any critical design decisions?

- Module Interface Specifications: defines each component’s interface in the design, including the services provided, parameters, and the effects of calling each service.

This section must provide an easily understandable system architecture specification, including critical architectural design decisions and their rationale. The design must satisfy the stated design goals.

Technical documentation - 10 points

This aspect evaluates the technical documentation, that is, the human-readable text that a programmer or human installer would use to understand (a) how to install the software, (b) software dependencies, (c) required versions of components, (d) how source code files relate to each other, (e) the purpose and inner workings of each source code file. This part must include documents that are separate from the code, as well as comments within the code.

Class Presentation - 2.5 points

The presentation should clearly and accurately convey lessons learned and the application of lessons from class (see Presentations.pdf).

Video Highlights - 2.5 points

Each team must produce a short video (5-15 minutes) showcasing the system's main properties seen from the Software Engineering point of view.