Project Grading

Final Project Delivery

Unfortunately, some projects receive lower grades than they should simply 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 make sure that you have one or more team members check the status of all deliverables well before the final date and again before delivery. The following checklist is not complete but may help avoid some of the more obvious errors:

1. **Completeness**: Check that each part of the *assembla* work space has been filled in. Make sure that there is a final sign-off on each work product by both the writer and a second person acting as reviewer.
2. **Consistency**: Where documents depend on one another, have the documents reviewed for consistency. For example, make sure that the User's Guide is consistent with the actual code delivered and with the requirements.
3. **Home Page**: Make sure that the home page has pointers too all of the work products including any web pages, server sites, or repositories. Double check that all of the links operate as expected.
4. **Application**: it is critical that you provide all of the instructions and links necessary for any idiot to set up (if needed) and run your application. I suggest testing this on real, live idiots. I cannot award points for code that I cannot run and will not have time to try to figure out how to run each team's application.
5. **Code**: Your final code must be uploaded to the *assembla* team site and you must provide instructions on how to access it so the code can be reviewed. Make sure that you have provided a README or equivalent describing the code structure and that the code is documented. In particular, be absolutely that any code that you have used from other sources (not written entirely yourself) is **clearly marked** to avoid any confusion over plagiarism issues.

Warning: you cannot afford to assume that external sites like *assembla* will be up on any specific day. Make sure you start your checks early and leave time to complete them before the delivery date.

Grading Overview:

A total of 100 points is possible, but don't expect to get anywhere the total possible points --- the point guidelines include a lot of "headroom" for exceptional projects, with typical scores being 50-75% of the maximum in each category. Note that a single problem (e.g., inadequate documentation, or a program bug) can cost points in more than one category. For example, if the documentation is not adequate for using a feature, you won't get credit for implementing that feature, nor will you get credit for documenting a feature that doesn't actually work (you may instead lose points for inaccurate documentation). Likewise, if a feature is implemented but cannot be used because the program crashes, you won't get credit for implementing that feature.

40 - Application Quality

Application quality includes features used by all categories of user. Often this includes administrative users (sysadmins, etc.) in addition to end users. Functionality also includes performance and scalability.

Robustness: 10

10	Absolutely bulletproof, could not find an unhandled exception
8	Robust under reasonable use, had to work at causing an error
5	Minor bugs, works well enough to be usable
2	Major bugs interfere with normal use

Feature Set: 10

10	WOW! Exceptional productivity, well beyond expectations.
8	All needed features and some pleasant surprises
5	Adequate for the intended purpose
2	Missing features interfere with normal use

Ease of Use: 10

This category includes setup (if any) as well as usability for end users and for administrators. It is typically distinct from quality of documentation, but may include presence and usefulness of online help. Usability considerations for administrative users may be quite different than usability considerations for end users; both are considered here. For a prototype, this addresses both existing and planned features

10	Couldn't ask for more
8	Quite usable, but could still be improved
6	Adequate usability, won't discourage normal use
3	Usability problems interfere with normal use
0	Unusable

User Documentation Including Installation and Setup: 10 points

10	Excellent guide to all implemented features. Clear guide to installing or setting up application as needed
8	Quite usable, but could still be improved
6	Useful documentation for some aspects of the software but significant omissions or errors

3	Not very useful, due to flaws, omissions, or poor organization
0	No user documentation, or useless documentation

35 – 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 a good design to degradation over time, but good documentation cannot compensate for poor design.

Project Plan: 10 points

This section is evaluated on how effectively the team preforms project planning, how well it follows the plan, and how well the team adapts the plan to inevitable hiccups. For example if risk are effectively considered, mitigated, and re-evaluated when plans change.

10	Report clearly indicates 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). There is clear record of when meetings were held, and what was accomplished and agreed upon at each meeting. A series of continually updated project plans continues to show all of the major project milestones and deliverables.
7	A plan was in place and followed. Some elements of the planning documents are missing or unclear. There is a record of tasks but it is not always clear exactly who did what and when or where the plan had to be changed and why.
4	A plan was in place and followed. Tasks and roles were assigned, but it is not clear who did what, when they did it, and how long everyone spent on each task (from assigned date to completed date, as well as time on task).
0	No final updated report of the project management is provided.

Systems Requirements (SRS & ConOps) –10 points

10	Clear, complete, and well-organized description of requirements, including rationale for what is included, what has been deferred to the future, and (as appropriate) what has been excluded. ConOps provides clear user-centric specification. SRS provides well defined, precise develop and test-to specification.
8	Good description of requirements. Some minor problems, but completely adequate as a guide to future developers. Shows clear understanding of the roles of the ConOps and SRS such that the right kinds of information appear in each. Reasonable expectation that others could use the document for maintenance or system creation.

5	Fair specification of requirements. Some kinds of requirements are missing or are put in the wrong document. Shows uneven understanding of how to write effective requirements or the roles of the requirements documents. Useful to maintainers and others but would probably require some support.
3	Requirements are barely adequate for project use. Missing significant parts or shows significant misunderstanding of how to write effective requirements.
0	Requirements statement is missing or inconsistent with the product; not useful to future developers.

System Architecture and Design Documentation - 5 points

The software design is informally communicated as follows:

- Software Architecture: This section should describe 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 most important application features?
- Why was this particular design chosen? What is the rationale for any key design decisions?

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

5	"Architecture" section effectively communicates how the components of the system work together to implement system functionality and the rationale for the major design decisions. Interface section specifies the services provided by each module, parameters, and service semantics.
3	Sufficient description of the structure of the system to determine primary functional structures. Gives some rationale for the design used.
1	Some material but inadequately explained. Key design decisions are not described or are unclear.

Detailed Technical Documentation - 5 points

This can be any combination of external and internal documentation (e.g. JavaDocs), and applies to both conventional source code (e.g., Java) and to other artifacts such as Makefiles, page templates, etc. This is primarily an evaluation of usefulness of the documentation, not raw volume, and necessarily includes factors such as coding style.

5	Exceptionally well documented and readable... easy to read and modify.
3	Well documented, but could be better in places.

1	Inadequate documentation or serious impediments to understanding.
0	Seriously inadequate or misleading documentation.

QA Planning – 5 points

For Project 1 the QA plan should include:

1. Planned reviews: specify when each artifact (e.g., the Requirements spec.) will be reviewed and record the results of the review.
2. Testing: specify when each increment will be tested. Provide links to all test cases used and record the results of the tests.

5	Documents which artifacts will be reviewed, by whom, and when. Results of reviews are documented. Fully documented test cases and results.
3	Well documented, but could be better in places.
1	Inadequate documentation or serious impediments to understanding.
0	Seriously inadequate or misleading documentation.

10 - Class Materials

Developer Logs - 5

5	Easy to determine who has done what
2	Present but missing some important information.
0	Missing or unusable

Presentation - 5 points

5	Presentation show key lessons learned and the application of lessons from class
3	OK presentation missing some essential information.
1	Omits major parts of requested information

15 - Overall SE and Project Control

Overall project control is the sum of the Managerial Control and the Intellectual Control. Good managerial control will show the ongoing management of resources in which the team meeting, Project Plan, milestones and Developer logs show that tasks are being effectively managed over time. Good intellectual control shows control of the development of the software functionality and qualities from definition of Requirements, through Design, Coding and Testing.

15	Excellent control demonstrating full understanding of the SE concepts covered in the course. Team is working as a unit to plan.
10	Overall good control demonstrating good understanding of SE concepts. Some parts may be uneven or minor difficulties executing the plan
5	Weak control with team members proceeding in an ad hoc fashion.
0	Crashed and burned.

E-mail: Email instructor
Based on the open source Sinorca design
with content from Prof. Michal Young and Prof. Anthony Hornoff