The University of North Carolina at Greensboro — Handout 3
CSC 490: Senior Project — January 26, 2010
Prof. Stephen R. Tate

Project Report Guidelines

This handout describes what is expected in each stage of your final project. Each major project stage includes a presentation and a report from each team. I care about concise, precise, and complete communication, with an emphasis on logical organization. Your decisions in one stage of the project should be easily understood and unambiguous, so that when you move into the next stage you have a clear agenda. I do care about proper grammar and spelling, but don't care about wordy, "pretty" language — be utilitarian and clear.

When creating your presentation, you should aim for approximately 20 minutes. Make sure you have a good, logical structure. The best presentations are "modular" in that they have several small, self-contained pieces that are related to each other in clear ways. The worst presentations are stream-of-consciousness brain dumps without any structure. For some guidance on criteria to consider in your presentation, see the peer review form categories.

There are no fixed length requirements for reports, formatting or font requirements. The presentation must be formatted so that it's easy to extract and understand the necessary information, but there are no requirements beyond that. If you have any questions, or would like recommendations, I'm happy to work through ideas with you.

Here are some things to keep in mind throughout all stages of the project: First, one purpose of the senior project is to show that you can balance choices and make good decisions. This means that you should be documenting alternatives that you consider, and why you made the choices you made. It's tempting just to simply report on what you put into your project, but that's not enough here — you also need to report on what you *didn't* put in your project (within reason, of course)! You should also keep a meeting log or journal of team meetings and decisions made. This doesn't need to be as formal as keeping meeting "minutes," but should document when you met, what items you discussed, and what decisions you made.

Project Topic Proposal (the "Pitch")

Concise description. Give your project a *name* and be able to describe the basic idea in just one or two sentences.

Background. Briefly describe anything that your project depends on that isn't common knowledge. For example, if your project is related to an existing game, describe the basics of that game.

More details. Next, flesh out your description a little. Give a high level description of your proposed solution — providing a basic screen layout describing how a user might work with the system would be very beneficial.

Feasibility and challenges. Describe what skills you think will be necessary to complete the project, such as knowledge of networking, databases, web development, etc. What skills does your team have and what will you need to learn? Do you foresee any particular challenges?

Incremental development. It's good to be ambitious, but you need to make sure you get something finished. What is the "baseline" project that you absolutely know you can get working? How will you prioritize features to add beyond this baseline? Does your project allow for a "plugin" style development where features can be added somewhat independently of the core project?

Team management. Describe how your team will interact. Will you have regular meetings? How will decisions be made? You don't have to have a lot of details here, but should have thought a little about how you will work together.

Report 1: Requirements/Specification

This report focuses on the user experience — you should describe *what* your system should do, but not really worry about *how* it's going to do it. Of course, you need to keep in the back of your mind what is feasible and what isn't, but you don't need to worry about internal data representations, algorithms, or anything like that yet.

System profile and requirements. Describe the computer requirements for your project. Will it run on a server? A desktop machine? A mobile device? You should consider these components: processing, memory and secondary storage requirements, and network use. Are there any special requirements for any of these? For example, some projects may require a lot of communication, so require a fast network. Some projects might need specialized hardware — if that's the case, you need to describe fairly fully what is required.

Use cases. Use cases describe how a user performs major tasks using your system. Each use case should come from a situation in which you say "The user wants to do X" — this is the *Goal* of the use case. In addition to the goal, each use case should include a clearly marked definition of *Actors* (who is involved, if multiple users) and a *Sequence of Events*. The sequence can be described using a list, a flowchart, UML, or even something like pseudocode — it's *not* describing an algorithm your system uses, but rather it's describing what the user is doing to accomplish this task. Other things that are suitable in certain use cases are *Preconditions*, *Postconditions*, and *Triggers* (what causes a use case to begin). You should have a use case defined for every major type of task the user is able to do in your system.

Use cases describe how the user interacts with the system, and so screen designs and other user interface components should be described here. You should also consider whether there are different *classes* of users — for example, in some systems there might be authenticated users (i.e., logged in users), guest users, superusers, etc. If use cases vary considerably for different classes of users, be sure to include this information.

Questions to be answered. As you sketch out what you want your system to do, you'll probably think of questions that you'll need to answer as you develop the design of your system. For example, will you keep all your data in memory, or work out of a database? These questions don't need answers now, but you should document these questions here.

Report 2: Planning and Analysis

The last stage was user-centric, describing how the user would interact with your system. This stage is system-centric: describe how information is modeled and represented in your system, how it is processed, and what tools you will use in building your system.

Preliminary object/process model. This report should contain definitions of the data types that you will use in your system, using standard representations. For example, if you use an object-oriented design for your system, use standard UML diagrams to describe classes and relations between classes. If you use a relational database, produce an entity-relationship diagram that describes tables and relations. Information about these representations and some tools to help you are available from the class web page.

For a process model, describe how information flows through your system and is processed for at least some of the use cases you defined in the previous report. You should think about places where you have to identify a specific data structure or algorithm, or make other design decisions.

System and algorithm analysis. As you develop the process model, you will identify places where significant computations are performed. By "significant," I mean anything that's not just doing some assignments and copying data around. This could be as simple as sorting some data or as complex as optimizing parameters of a complex system. In most cases there are multiple solutions for these computational problems — describe some alternatives, give the pros and cons of each as well as you can (use proper computer science reasoning that you've learned in your courses, such as asymptotic time analysis). If you can make a decision at this point, describe how you made that decision based on the pros and cons (and potentially other issues like the amount of data involved). If you cannot make a decision at this point, describe how you will make this decision —

will you code multiple algorithms and see which on works best? Do a more in-depth analysis? Just like in the initial design, you are encouraged to prioritize here: you can select an algorithm that you know will get the job done, while giving a lower priority to later testing and replacement with a more efficient algorithm. Your design must be modular enough to accomplish this though!

Tool selection. At this point you should have a fairly well-defined project, and you need to select what tools you will use. This means deciding things like a language, development environment, database server or package, and third-party libraries. It is important not to re-invent the wheel! If you are using standard data structures, use the ones provided with the language you're working with. If you need some standard math operations then look to see what's available (linear algebra packages, etc.). There are some excellent graphics and user interface libraries out there — do a little research and find one that is a good match for your system and your other tool choices. It is likely that in some cases there are multiple options to choose between, so pick one or two places where you had to make a decision and describe the process you used to make that decision. List the pros and cons for each option, which could address things like stability/quality of the tool, performance (you could run tests), usability, reviews by other people, etc. Finally, describe what considerations you used in making your choice.

There is one tool that everyone is required to use: You will use the Subversion (SVN) revision control system, with the repository on a department server. We will discuss how to work with SVN at an appropriate time, but all of your code that you turn in will be submitted by checking it in to SVN. You are encouraged to use it regularly in order to coordinate code development with your teammate(s).

Report 3: System/Research Design

In this stage, you move beyond planning and design, and into constructing parts of your system. At this point you will be primarily defining your data types and constructing initial databases. You should take your object/data models from the previous stage and turn those into actual programming language definitions (e.g., a Java class) or SQL definitions (for example, a MySQL script that does "CREATE TABLE" commands).

Amended models. Once you start realizing your object/data model, it is often the case that you discover things that need to be added or changed. Any updates to the models defined earlier should be reported here. Note that changes to your model reflect decisions that you've made — explain clearly why you decided to make this change (what problem is it correcting, and what would have happened if you didn't make the change).

Detailed designs and controls. The code that defines your objects, or the SQL statements that create your databases, contain all the precise information about your data types and representations. You should make sure to check in all code to this point, including the code that defines/creates your data types. You don't need to turn in a complete printout of this, but might include parts in your report to help illustrate particular changes you had to make (in describing "Amended models").

Test plan. The test plan is one of the most important parts of this stage. This should very carefully describe tests that you will perform on your system, including details such as where the testing data comes from: is it a constructed data set, or a randomly generated data set, or what? You should have both unit test cases (testing a particular class, for example) and integration testing (testing how the units work together). Keep in mind the use cases you defined earlier, and make sure your tests can support the claim that your system is achieving the goals defined in each use case. Your test cases should also include extremes and even invalid data to check border cases. You are encouraged to use a code coverage tool in performing tests, but if you can justify that your test cases are thorough and complete without such a tool you can do that as well.

Report 4: Implementation and Testing

The presentation for this phase is a little different — rather than a summary of the report you'll turn in, this presentation should be a "code walkthrough." You won't have time to go through all the code you've developed, so first give a high-level overview of how your code is structured, and then pick a part that you think is particularly interesting (because it solves a challenging problem, for example), and go through that code more carefully. If there are places where you're stuck or would like help, this is a good place to seek help from your classmates.

Source code. You don't need to turn in a printout of your source code, but you need to make sure that everything is checked into the SVN repository — this includes all the code you've developed, including tools or scripts to set up databases or perform tests.

Test results and discussion. The only hardcopy report that you'll turn in gives the results of the test cases. Remember the test cases you defined in Report 3? You should execute those plans now! For each test, did you get the answer you expected? Is there anything that might need extra work?

Final Poster

Your final poster should describe your project in a visually pleasing way at a high level. These will be printed to be approximately 3 feet by 4 feet, and a template is supplied for you to use. Posters should use lots of figures, diagrams, and pictures, and be relatively light on text — a lot like a good power point presentation, but in a different format.

At the top of the poster you must put a title describing your project and the names of the students on the project team. The first thing people should see after the title is a concise and clear description of the problem you are trying to solve. Then a separate block should describe your solution — just focus on the big ideas at this point. Including screenshots to show how the user interacts with your system is very beneficial here!

The rest of the poster varies depending on project and what's interesting in the project. Is there a design component that was particularly interesting? If your project has a computationally intensive part where selection of an algorithm was important, explain that. If you deal with a lot of data, explain how it is structured, stored, and used. If there is an emphasis on user interaction, describe the choices you made for the user interface. In each of these cases, it would be good to briefly describe why you made the decisions that you made: Did you test multiple algorithms? Did you run a usability test? At the end of this part, you should include some statistics or metrics: How many lines of code did you write? How large is your database? Is anyone actually using your system yet? Place yourself in the position of a technical person learning about your project for the first time — does your poster answer the questions you'd be most likely to ask?

As a final part of the poster, you could include thoughts on future directions — in other words, if this project were to continue, what would you want to do next? Add additional features? Tune performance? Improve the interface?

Final Presentation and Report

The final presentation should be a coordinated presentation in which each team member speaks for roughly the same amount of time in which they focus on some aspect of the project, highlighting their individual contributions toward the project's success. Other students and faculty will be attending your presentation, so impress us on what you've done and how great your project is!

The final report is due at the scheduled final exam time for this class. The final report is a *technical report* that includes a self-contained one page summary, suitable for putting on a web page. Your technical report should summarize what you've done in your project, without being redundant with reports written for the earlier stages. You should give a final summary of any design decisions you made during the course of the project, and why you made the decision you made. For projects that aim to develop a usable software system, the technical report must include a "user's manual" that describes the system to an end-user. In addition to the team's project report, each member should individually turn in a short summary of their own contributions and their own thoughts on teamwork aspects of the project (specific guidelines will be provided).

Final reports should be emailed to the instructor in PDF format. I will grade these and add comments using the PDF commenting and markup features, and then will mail this back to you.