CS 471 - Senior Capstone I, Fall 2013
Technical Writing Tips

General Technical Writing Tips

- Keep in mind that writing is more difficult than programming (the language is bigger with a much more expressive syntax)
- Should look like a professional document
- Keep technical documents simple and "dry"
- If you can express an idea in 7 words instead of 10, do it!
- Formatting/fonts/margins/acronyms/etc must be consistent
- You have to assume that these will be printed and used in hard-copy form. Therefore, you will need a quality table of contents with correct sections numbers.
- Use strong/professional language:

weak/informal	**strong/professional**
made	created/developed
give	provide
is going to	will
will help	improve
wants to have	requires
be able to let	allow
handle	support
talk over network	communicate
so as to	to
to reduce excess verbosity	

Software Requirements Tips

- Use and follow the template provided - it is similar to following a checklist
- Start with user stories. They should be narrative and describe how users will interact with the software system. *(Example: After accepting a contract, the user will create several tasks, providing a time estimate and priority for each new task. Next, the user displays all unfinished tasks by priority and makes any adjustments needed.)*
- Include sample interfaces - a picture is worth 1000 words - to help drive discussions with the client and reduce writing required
- Number all requirements for reference *(e.g. This code tests the non-functional requirement 4.8.1 in the Software Requirements Document)*
- This document describes WHAT is needed, not HOW it is done.
- Rare to use a bullet list - everything needs to be numbered for reference (in SWR, DPD, TESTING, code, etc)
- We usually do not specify priorities of tasks in SWR, since they are almost always "high" or "very high" and will change during development.
- Use a metric than can be measured instead of vague goals, e.g. needs to run quickly:

vague	**measureable**
must be fast	must respond to user input within 1 second
must handle big task lists	must support at least 1000 active tasks and 1,000,000 archived tasks
deal with heavy loads	must support 1,000 simultaneous users
easy to use	after 4 hours of training a user will make less than 1 mistake per hour