Software Documentation Assignment (Lab 5a and 5b)

Professor Redd
IT&C 210B

Objective

The purpose of this assignment is to give you practice writing software documentation. There are different types of software documentation. Some documentation is written as help files for the users of a program. Other software documentation is written for future developers who may inherit your code (including yourself after you've forgotten what you did last year ☺). This assignment focuses on the latter (i.e., documentation for future developers).

Procedures

Part of good writing is knowing your purpose and your audience. These are spelled out for you below for this assignment:

- **Purpose:** Document your code for lab 5 (both part a and b, but focus on the finished product in part b) in enough detail that you can (a) get an overview of the overall design, (b) understand why certain decisions were made (e.g., understand why we are using a different port than the standard port 80), and (c) have detailed descriptions of the specifics (e.g., descriptions of routes, variables, methods, and functions used – note this is often done in the code itself using commenting).
- **Audience:** You can assume a technical audience (i.e., future developers) who is familiar with technical terms. However, even a technically competent reader can't read your mind, so make sure you spell out why you've done what you've done and the implications, which they might not recognize at first. In short, imagine you're writing to another IT&C 210 student, though make sure to use a more formal business writing tone. For example, you should not use the first person (Instead of "I added this because", use "The meme's creation page includes a … because"). Don't talk about "lab 5" by name. Instead, describe it as if it were part of a project that isn't necessarily for a class, so you can get practice writing for a professional audience. Just describe what work was accomplished and how it fits into the larger picture of the project. For example, the title shouldn't be "Lab 5 write-up" – it should be something tied to what you actually did in lab 6 (e.g., "Creating ToDo App with NodeJS and Vue"). As always, avoid the passive voice whenever possible and be efficient and clear (i.e., unambiguous) with your language.

With that background, here are the specific steps needed to complete this assignment:

Please upload (via LearningSuite) a documentation file in a single .doc or .pdf format (to facilitate easier commenting on it). If it is necessary to quote material or graphics from other sources, you must make this clear in the appropriate way in the document and you must give appropriate credit for the material quoted or used. You may not work with others on this project. The report should include the following components:

Cover page: Include an appropriate title, your name, the class, and semester

Table of Contents: Include a table of contents with page numbers for the major parts of the file

Overview: Summarize, in 3-5 sentences, what was accomplished during the project (i.e., this lab). Be specific about the technologies used to do it. Mention why it was done the way it was (justify the design and the choice of technologies used, but *briefly*).

Design: Describe the overall design of your work. Specifically, discuss how the webpage/functionality works in terms of the technologies used and how they interact. This is *not* where you go into the details of your code. Rather, explain the architectural design of what you did. There are two parts to this section:

a) Describe how the user interacts with the new features that you've added. For example, "Users will type in a new to do item and click submit. This will call the … route, which will have NodeJS add … to the MongoDB database and … Users will then see …" Use screenshots to help illustrate and explain.
b) Present and explain the UML Sequence Diagram(s). From a technical perspective, explain what data is being passed between the various component parts (e.g., NodeJS server, Vue front-end, Mongo database). This section should fully convey the high level approach you took. It should also include justification for your design (e.g., "An API architecture was used in order to…") and any limitations therein (e.g., because MongoDB is stored in the cloud…).

Detailed Documentation: This is where you discuss your code. Describe in a succinct way the key files (and how they're arranged), objects, functions, methods, attributes, etc., that are pertinent to what was done in the lab. This is likely best done using bullets or tables. Make sure they are organized in a meaningful way and sufficient detail is provided.

For example, key routes, functions, and variables should be described in a sentence that outlines their purpose and any notes about their use. Where appropriate you can insert pieces of your code or reference their position / location in the Documented Code section.

Tips & Warnings: Include at least 3 tips and/or warnings that highlight common errors or misunderstandings. These are similar to tips from regular lab write-ups that you create. Remember, don't use first person.

References: Refer to references in the text (i.e., the sections above) with numbers (e.g., [1] or [2-3]) and then in this section list the references themselves using the following examples as a guide:

1. AuthorFirst, AuthorLast, (Other authors), "Title of the reference", Publication, publisher, date

2. Organization_or_committee (if no author listed), "Title of publication or web-page" web-site, publisher, date

3. Wireshark Foundation, "Wireshark: the world's foremost protocol analyzer" http://www.wireshark.org/, Wireshark Foundation, accessed Sept 2011.

4. Stack Overflow, "How do I convert a string into an integer in javascript?" http://stackoverflow.com/questions/1133770/how-do-i-convert-a-string-into-an-integer-in-javascript, Stack Overflow, accessed Sept 2011.

Documented Code: Include as an appendix your documented code. This should include comments that help explain variables, functions, etc. in context. They may include the same text as used in the Detailed Documentation section, but in this case they will be inside of your code using the language commenting features. Hopefully you have been doing this throughout the project, so it won't be a lot of extra work. It's a key habit to get into for any coding.

General Rules of Writing:

1. All figures and images need to be labeled and captioned. When referring to them in your writing, use the label (e.g., "Figure 1" or "UML 2"). The caption should briefly inform the reader of what the image is.
2. Do not refer to your work as "Lab 5," or as a lab in general. Give your work/project a name, and use that. Write this as though you were performing in a professional situation, not in a class.
3. Do not write in the first person. I.e., do not use the pronouns *I* or *we*, or any of their corresponding constituents: *me, my, our, us,* etc.
4. This is not a narrative document. Your purpose is not to tell your story of how you did the lab; your purpose is to describe and explain the work itself. For example, "The developer chose to use Java because…" is a narrative statement. Instead, say "Java possesses properties X, Y, Z. This allows it to interact with Technology W in such a such a way, and accomplish V." The more you avoid narration, the easier it will be to also avoid writing in the first person.
5. Use headings for your sections. It seems obvious, but make sure the headings are visually distinct – large font, bold text. Headings are like chapter titles in a book; they tell your reader where he/she is in the document.
6. Do your best to keep the document visually organized and tidy. When it comes to formal writing, presentation is just as important as the content of the document itself.

Grading Rubric

Your assignment will be graded based on the following:

- Clarity & accuracy of technical writing (15 points)
    - Avoid vague terms like "it" when stating what "it" is would be clearer; be specific and detailed.
    - Make sure to introduce terminology that may not be obvious to your audience.
    - Make sure your statements are not ambiguous (e.g., "clicking on the ASP.NET 'submit' calls the Node.js ImageFilter function, which applies the filter chosen by…" is less vague and potentially ambiguous than "clicking the button runs the filter").
    - Make sure your statements are accurate.
    - Use appropriate headers for subsections so people can quickly find where they want to go if they are only scanning the tutorial for certain information.
- Style, grammar, and spelling (15 points)
    - Avoid grammatical and spelling errors (e.g., changing tenses within a section; plural/singular agreement problems such as "the [singular] *user* did this, and then [the plural] *they* did that")
    - Code snippets should be in a different font than the explanatory text. Use a monospace font (e.g., Consolas, Courier New) that clearly differentiates between otherwise hard-to distinguish letters: l, 1, I, (l, 1, I).
    - All figures and tables should be labeled with appropriate titles (e.g., Figure 1: UML Activity Diagram Showing Login Process; Table 3: Variables used in Login Process). Not

all code fragments need to be labeled, unless you want to refer to them more than once.
    - This is formal documentation writing, so make sure your language is professional. Nevertheless, personal pronouns ("I", or "we") are acceptable.
    - Use appropriate formatting. You can use a Word Template if you'd like (which may even create the table of contents for you). It should look professional.
- Comprehensiveness and accuracy of content (20 points)
    - Cover Page and Table of Contents (1 points)
    - Overview (2 points)
    - Design (4 points)
    - Detailed Documentation (4)
    - Tips and Warnings (4)
    - References (1)
    - Documented Code (4)