COMP 401 - Software Development Based Project Proposal

Fall 2014 - Spring 2015

COMP401 culminates with the formal proposal of your senior capstone project. The proposal consists of three pieces

1. Written Project Proposal
2. Presentation and Defense
3. Poster

Your presentation is a summary of the written proposal given in a public forum while the poster serves as a condensed snap shot of the project you'll be implementing in COMP402. Neither should introduce information not found in the written proposal.

1 Written Proposal

Your written proposal should contain the sections listed below. Each section has an estimated page length next to it. Page length estimates assume *single spaced, 11 pt. font.* Please note that these are estimates and the most important thing is that you're detailed enough for your particular proposal.

1. Abstract [1-2]
2. Background/Introduction [5-10]
3. Use Cases [3-5]
4. Users, Stakeholders, and Ethical Questions [1-5]
5. Features [1-2]
6. Implementation Framework [1-2]
7. System Design [5-10]
8. Testing Plan [1-2]
9. Proposed Implementation Time Line [1-2]
10. Conclusion [1-2]
11. Bibliography & Works Cited [1-2]

1.1 Abstract

The abstract should be roughly one to two pages and should give a high level summary of the document. One should be able to read your abstract and determine what your project is and why it's interesting/important. In the case of software projects, abstracts act almost as if they were a sales pitch. Tell the reader what's compelling about your proposed software and what tools you plan to use to implement the final product. The goal is to get the reader interested in the software and believing that you have a firm plan to implement it.

1.2 Background/Introduction

This section introduces the project along with the problems and ideas that led to its conception. You should address:

- The problem (social, technical, or otherwise) addressed by your software.
- How, specifically, your software addresses/solves the problem.
- A brief discussion with background information on any hardware/software/computing technology integral to your project, but not specifically related to its implementation. The idea is to describe and discuss all the technology behind your software up to the point of specific implementation choices. For example, if your software is web-based, you should discuss the World Wide Web and the basics of web servers and browsers but not a specific client/server package. This information should be generalized and not specific to instances of the technology you'll be working with.
- The niche filled by your software. Meaning, if similar software is out there, then how is yours different/better, or if your software is totally unique, then to what do you attribute this uniqueness. If there is similar software available, then you should reference and and provide a comparison. If there is no comparable software, reference anything that my be close but falls short.

Once you have introduced the problem and how your software will address it, you should then highlight all of the technology which plays an integral part in the software system as a whole. The section can range from five to ten pages long, with the bulk of that coming from background information regarding the technology surrounding your software. You can assume your reader is computer literate and able to read and comprehend technological writing, but that he or she is not familiar with anything beyond basic computing and applications.

1.3 Use Cases

Here you should describe 2-5 typical use cases of your software. These are essentially stories detailing the experience of a typical user for your software. They should be interesting, compelling and clearly highlight how you see this software being used. Avoid being vague about the usage of your software but instead specifically highlight its key features. Here are a few things to consider.

- The sum total of your use cases should clearly highly all of the key features of your software.
- Rather than write one or two long use case "stories", favor short to medium length use cases which specifically target a small set of features.
- Do not focus on low-level technical details unless they are integral to the user-level usage of your software.

Use cases are about functionality and usage of the software and not the nuts and bolts of the underlying system.

1.4 Users, Stakeholders, and Ethical Questions

Here you should list and describe the people that you see using the software, *users*, as well as anyone else who may be impacted by the usage or availability of your software, *stakeholders.* Stakeholders should include the various development communities surrounding your software's problem domain and the tools used to implement the software. Once you have discussed all of the people and groups impacted by your software , discuss any ethical issues that may arise through the usage/availability of your software. If you are specifically addressing any of those issues in the design and implementation of your software, then you should discuss how. If you are not addressing an issue, then discuss why and propose a possible course of action taken by the users/stakeholders to address the issue at their end. It would be highly unethical of you to purposefully fail to mention a possible issue/problem that could arise from your software just so you do not have to discuss and write about it in your proposal.

1.5 Features

This section should contain a concise list of the features of your software along with a short description of each feature. The description should briefly touch on why the feature is available as well as any other software features that are dependent upon the feature being described. The set of features provided by your software should be easily extracted from within this section. Readers should not have to search for features amongst prose.

1.6 Implementation Framework

It is in this section where you should clearly pinpoint the technology you will use to build your software. You should give a concise list of the the tools (Programming Language(s), software, and hardware) as well as a rationale for choosing each item. Once again, readers should not have to parse this section to determine what tools you'll be using.

1.7 System Design and Architecture

This section provides the overall design of your software. The design should be detailed enough that a trained programmer could implement it but still have to make decisions regarding the functionality of some low-level functions/routines. All of the high-level and critical elements of the design should be detailed at the *what* level with the *how* level of detail left unsaid. The one exception to this would be the case where the choice of implementation (Data-Structure/Algorithm) is critical to the success of the software. Your design should clearly indicate:

- Modules/Subsystems of the Software
- Interactions and dependencies between each Module
- Languages/Software used by each Module (where appropriate)
- For each Feature, what Module(s) of the system are responsible for carrying it out.
- Key/Critical ADTs, Data-Structures, and Algorithms to be used along with where and why they'll be used.

You are highly encouraged to utilize any design tool you've come across in your studies. This includes, but is not limited to:

- Diagrams and Visual Models
- UML Specification/Diagrams
- Contract/Purpose Based Specifications

Once again, the design you're being asked to detail here should paint a clear picture of all the parts of the system and how they will work together to carry out *all* of the features of the software. It is acceptable to leave out implementation details out, but the overall structure of the system and the programs involved should be fairly obvious. *The use of diagrams and visual modeling tools is highly encouraged.*

1.8 Testing Plan

Clearly layout the plan for testing your software system. Discuss how you plan to test each module and how its testing will be timed with respect to the testing of other modules. You should not discuss how the testing fits into the complete project time line, but rather how testing itself will be ordered over the course of the semester. You should clearly describe the data/test cases you expect to use in testing. Essentially, this section should make explicit how you intend to verify that your system is in working order.

1.9 Proposed Implementation Time Line

Give a weekly time line for the implementation of your proposed software over the course of the spring semester. You should have clear, explicit tasks listed for each week. These must at least include:

- When modules/features are being implemented
- If a proposed task is a partial implementation of a module or feature, what exactly is being implemented and to what module/feature it relates.
- Tests/Testing To be Done

In the spring, you will be doing checkpoint presentations based on these time lines at which you describe and justify your progress. It will be your responsibility to either stick to your time line or be able to give reasonable justification for deviations from your time line. Being unprepared for weekly checkpoints will have a negative effect on your grade. It is therefore important that you think hard about how to budget and manage your time with respect to this project. Not all tasks, related to the implementation of your software, are created equal and your time line should reflect this.

1.10 Conclusion

The final section of your proposal should be both a summary of the proposal and a place for you to reflect on the work ahead of you. Be certain to write about:

- The importance/significance of your software
- The technical challenges that you foresee in implementing your software
- The personal challenges that you foresee in implementing your software

Make it clear to the reader that this software is worth writing and that you are fully aware of the technical and personal challenges that await you and ready to face them head on.

1.11 Bibliography & Works Citied

Any information that is not your own and is used in the writing of your proposal should be properly cited. If you refer to technical details about software/hardware that you yourself did not create, then you should cite a reliable source for this information. Beyond the citation of information your bibliography should include:

- At least one entry to a reliable Reference *for each* item listed in your Implementation Framework

Essentially, you need to show that you have at least one reference or guide for each piece of computing technology (language, software, hardware, etc.) that you will use to implement your software along with any other sources that you used in researching the various aspects of your project.

1.11.1 Regarding Wikipedia and Web Sources

Wikipedia is a great starting place, but rarely is it a good final source. If you directly cite a Wikipedia entry you should reference it, but you should really seek out a primary source for that information. *Obvious reliance on Wikipedia as your primary source of information is likely to reflect negatively on your grade. Given the prevalence of links to other sources at the end of most Wikipedia entries, failure to explore those sources in search of a primary source is just lazy.* The above comments are limited to Wikipedia and do not generalize to all web sources. It is highly likely that many of your references will be web-based. Most computing technologies, be they 'hard' or 'soft', have online documentation produced by the technology's creators. You are should seek these references out and *include them in your bibliography when available.* If there is a source of information that you plan to use to aid in the completion of your project, then it should be in your bibliography.

2 Presentation and Defense

You'll be giving a *twenty to thirty minute* proposal of your project the focus of which should be on the overall features of the software and design for carrying out those features. The defense aspect of the presentation comes from the fact that you should be prepared to discuss/defend your design. If it's not clear where/how a feature is managed by the software, expect a question about it. If elements of the design are overly vague, expect questions about that. If there's an obvious ethical consideration with your project and you fail to mention it in the proposal/presentation, expect questions about it. These questions are not intended to "trap" you, but rather to make sure you've thought out the design and your plan for carrying it out prior to actually beginning to carry out the project in the spring. The goal of the proposal is to be certain that you have a clear path laid out for success. The presentation is the time when you convince us that what you wrote is more than just words on paper.

Here's a rough outline of what your presentation should entail with time estimates, in minutes, for each item. The information for each section should mirror/summarize the relevant section from the written proposal.

- Background/Introduction [2-5]
- Users, Stakeholders, and Ethical Questions [2-3]
- Features [2-3]
- Implementation Framework [2-3]
- System Design [5-10]
- Testing Plan [3-7]
- Proposed Implementation Time Line [2-3]
- Conclusion [2-3]

You must have Power Point slides to accompany the presentation.

3 Poster

Your poster will act as a snapshot of the written proposal but without a majority of the technical details. It touches on information from the following sections of your proposal.

- Background/Introduction
- Users, Stakeholders, and Ethical Questions
- Features
- Implementation Framework

The posters will be hung in the hallway where the Math/CS offices can be found. They are intended to get people excited about your work and give them a feel for the types of technologies you're working with to create your final product.