Project Selection Guide

Krzysztof Czarnecki
January 22, 2021

The selection of a suitable project topic can be challenging; therefore, I provide you with some guidance and tips.

1. Key considerations

The key questions to consider when selecting a project topic are the following:

1. What are the interests of the team members (e.g., application domains, technologies)?
2. What opportunities exist? For example, some team members may be working in a research lab or research project that has some software needs; some might have developed some software as a hobby project that can be expanded on; some may have experience with an open source software that could be improved as part of this project; there may also be ideas for software to address some personal needs, e.g., door camera/display to screen visitors at a house entrance.
3. What skills exist in the team (e.g., knowledge of certain technologies and application domains)?

2. Project idea brainstorming

As part of the selection process, organize a brainstorming session with your team. You could start the session by answering the three questions above, and then continue by going over the different project categories listed in the next section and generating multiple project ideas for each of them. When generating ideas, avoid too much critique and being overly negative – you want to generate as many and diverse ideas as possible, without killing ideas too early. In the second part of the brainstorming session, you want to go over the generated ideas and evaluate them based on the following criteria:

1. **Motivation/excitement**: You want the project to have some aspect (e.g., application domain, problem being addressed, or technology) that will excite and motivate the team members.
2. **Learning value**: The main objective of the project is to learn software development practices hands on, but additional learning opportunities, such as learning about an application domain or technology are also valuable.
3. **Feasibility**: Assess the existing experience and expertise to work on a particular project idea; this can be within your team or by being able to access some online resources, such as tutorials, sample applications using certain technology, etc. The scope of a project is usually not a problem, since you can always reduce the scope to fit the timeline, and you don't need a finished product at the end, but only a working prototype of some core or interesting functionality.

Alternatively, one or two of you may already have an project idea that you are sure about; in that case, you want to see if you can recruit additional team members for the specific idea on Piazza.

3. Project Categories

Project ideas can be generally classified into the following categories:

1. **New product**: innovative software solution that addresses an important problem and gap on the market.

This category is most challenging, as it requires recognizing a real business opportunity. The development would involve doing some market study (like a user survey and/or interviews) along with prototyping a minimal viable product (MVP).

2. **Open source**: contribution to an existing open-source software of your choice.

The objective is to contribute to an existing open-source project of your choice. This could be some software that you are already using, or perhaps even one with a codebase a team member has already some experience with. The project would typically involve selecting some bug reports or feature requests from the issue tracker of the software and developing pull requests to address these issues and having them accepted by the maintainers of the open source software. A first iteration should normally pick a bug that is very simple to fix, to get familiar with the codebase and the overall contribution process. The second iteration may then pick a more challenging bug or a new feature request.

3. **Application software in an established area**: a “clone” replicating functionality form one or more existing applications.

Since the purpose of the project is to practice software development, the novelty of the solution is irrelevant. One way to get experience in a certain application domain (e.g., enterprise software, consumer apps, embedded systems) and the technology stack used in that domain is to replicate some functionality from already existing products.

4. **Advanced technology**: project that explores advanced technology, such as machine learning, distributed systems, blockchain, brain-computer interfaces, etc.

Team members may be interested in learning some advanced technology, and possible project would involve building prototypes that use the technology for some purpose. The novelty of the application is irrelevant, since the focus is on getting hands-on experience with the technology.

5. **Research**: software supporting research, typically being done by one or more of the team members.

The context for this project category is research that is aimed at producing publishable scientific results, but the software to be developed is needed for this research and may represent prototypes of new methods or algorithms that are subject of the research, or may be needed to support experiments.

6. **Customer**: develop software for a specific customer.

The category relies on an opportunity where a customer who has some software needs is available. This could be a research lab, a company, a non-for-profit, etc., that one of the team members works or worked for. Such project can be most rewarding, as having a customer focuses the effort. On the other hand, the customer needs to be available for consultation with the team in a timely manner in order not to delay the progress. Also, customers that are companies may create intellectual property issues.

4. Examples of Past Projects

There is a wide variety of projects that students have worked on in the past. Some of them are listed below.

Business systems (mostly in the "Application software in an established area" category):
1. Seat reservation and real-time ordering system for restaurants
2. Event management system (e.g., event space, catering, flowers, etc. for weddings, birthdays, etc.)
3. Web application for online auctions
4. Patient record and visit booking software for family doctor offices
5. Online grocery ordering and delivery application
6. Investment tracking application

Consumer apps (mostly in the "Application software in an established area" category; some attempting "new product" category):
1. Recipe recommendation app
2. Habit tracker app – develop good habits with challenges and reminders
3. Personal finance planner

E-learning (some attempting "new product" category):
1. Sharing comments and important video locations for lecture videos
2. E-learning flashcard software
3. Algorithm visualization tool – website with interactive visualization of how algorithms (e.g., sorting, searching, shortest path computation, network optimization) operate

Advanced technology/ Research
1. Human sentiment analysis and visualization – using ML to visual sentiment on social media (e.g., Twitter)

2. Music recognition and note transcription software
3. Automatic generation of image sprites for game development using machine learning

Customer

1. Video labeling software for a research lab (where one of the team member works)
2. Refactoring of autonomous drone software for a research lab (where one of the team member works)

5. Writing the project abstract

As stated in the course description, the project abstract should state the problem that the software will address, sketch the expected solution (e.g., listing the main functionalities to be implemented), and state the expected learning from developing the system (e.g., development practices and technologies that team hopes to learn). The abstract should be roughly half-a-page long in PDF. As you start your development, you do not have to stick exactly to what you have proposed originally—you have the flexibility to adjust the topic and scope, or even pivot, as you gain experience. If you pivot, just submit an new abstract.

Think about how you want to break up your project into iterations, e.g., each two week. Two of your releases get evaluated by the instructor: one around the midterm, and another one at the end. Assign an overall objective to each iteration. Think what tasks the project involves and capture them in your backlog. Break down the most important tasks into smaller ones, each doable by a team member in a day or two, and assign the important tasks to your iterations. You can and should revise your plan as you go, keeping track of it in your task board. Thinking about the breakdown early will allow you to pick realistic project scope.

When writing the abstract, include the following items. See the lecture slides for a sample abstract.

1. Opportunity and problem
   a. What is the opportunity that your software will exploit?
   b. Who will use it?
   c. What problem will it solve?
2. Objective
   a. What does this system/project will accomplish within the given context?
3. Plausible design approach
   a. What is the high-level solution approach that will be pursued?
4. Expected benefits over existing alternatives
   a. (Note: this is important for an actual product or service, but much less for your course project. Here you can also focus on your expected learning outcomes)

Make sure that you abstract submission includes the project title, project category, names and student ids of the members.