**COMP491 Assignment HW3: Project reviews**

This assignment is worth 60 points and should take about six hours to complete. Record your activity in real time on the Slack live log. Save the results of your work in a document that will be submitted to Moodle as PDF; this is called your *responses document*. In addition, you must add certain information to our course wiki.

**Overview**

You will complete detailed reviews of two H/FOSS projects. (Exception: students working on individual research or honors projects will review only one project.) The reviews are placed into your responses document. Log your activity using the Slack live log. Each review should take about three hours, for a total of six hours spent on the assignment. You should review the projects that you believe are most promising for your senior seminar capstone projects. It is easier to review projects that you explored in the previous assignment, but it is also fine to review a project that you have not yet explored. You are encouraged to review the projects that you are most excited about, regardless of whether you have previously explored them. Feel free to browse the course wiki and discuss with other students to find out about potentially promising projects that were explored by others. You can also explore and review a completely new project that has not yet been explored by anyone in our class.

**Part A: Reviews**

For each project you review, the following information must be recorded in your responses document.

* **Project Exploration:** This is the information from the Project Exploration assignment. Paste it in at the start of your review if this is a project you have already explored. If you have not previously explored this project, complete the information from the Project Exploration assignment and inserted into your responses document.
  + **Developer Community Communication:** This section will explore how active and responsive the community is. It will give you an idea of what it would be like to try to ask questions or engage in discussion with the community.
    - **Channels:** Most projects will have several communication channels (e.g. IRC/Gitter/Slack/Discord/Mailing List/Forums/etc.), with each serving a different purpose for the community (team meetings, Q&A, knowledge base, live help, user forums, developer chat, community building, ect.). What are the main channels that developers use to communicate with each other in this project? Note that while the Issue Tracker is a communication channel, it is addressed separately below, so should not be included here. Answer the following questions for each of the channels that the project uses.
    - **Name/Link:** What is the name of this communication channel? What platform is used for this channel? Give a link to the main page for this communication channel.
    - **Purpose:** Describe how the community uses this particular communication channel?
    - **Currency:** When was the most recent information added to this channel?
    - **Activity:** How active has this channel been recently? You might estimate how many new messages are posted in the last day, week or month. Briefly describe how you made your estimate.
    - **Responsiveness:** How likely are requests for information/help on this channel to receive responses from the community? Categorize this as: very likely, probably, maybe, or never. Briefly describe why you chose the category that you did.
    - **Response Time:** When a response is received, on average how long does it take?
    - **Members:** Who are a few of the most communicative community members on this channel? (Nicknames or handles are fine.)
    - **Tone:** Describe the tone of the discussion on this channel. Is it welcoming, formal, informal, highly technical? Do you think you would be comfortable interacting with the community on this channel?
    - **Summary:** Summarize your overall thoughts on the communications of this project community. Are the communications channels active? Are community members responsive and helpful? Is it a welcoming and inclusive space? Are newcomers and beginner questions treated kindly?
* **Issue Tracking:** This section will explore how issues are reported/tracked/discussed by the community. It will give you an idea of what it will be like to work on bugs/features within the community.
  + **Name/Link:** What system is used to track bugs/feature requests? Give a link to the main page of the issue tracker.
  + **Volume:** How many open (non-closed) issues are there at this moment?
  + **Currency:** When were the most recent 5-10 issues opened? Note: Most issue trackers will allow you to sort by creation date. Estimate this as within a week, within a month, within 6 months, between 6 months and a year, or over a year ago.
  + **Activity:** When were the most recent 5-10 comments/edits/revisions of issues? Note: Most issue trackers will allow you to sort by the date of the last activity. Estimate this as within a week, within a month, within 6 months, between 6 months and a year, or over a year ago.
  + **Resolution:** When were the most recent 5-10 closed issues closed? Note: Most issue trackers will allow you to view closed issues and sort by the date they were closed. Estimate this as within a week, within a month, within 6 months, between 6 months and a year, or over a year ago.
  + **First Issues:** Most issue trackers allow issues to be tagged/labeled. What (if any) tags/labels does the project use to identify issues that would be good for new developers?
    - **Volume:** How many issues are tagged/labeled in this way?
    - **Currency:** How recently were they posted?
    - **Approachability:** Read a few of these issues. Do they provide enough information so that they seem approachable?
    - **Activity:** Have comments/questions on them received helpful responses from the community?
    - **Resolution:** Are pull/merge requests for issues tagged in this way being accepted?
  + **Summary:** Summarize your overall thoughts on this project’s issue tracker. Is the issue tracker active overall with issues being discussed and resolved in a timely manner? Does the community encourage, support and value first contributions?
* **Getting Started:** This section will explore what resources are available to new developers looking to join the community. It will give you an idea of what it might be like if you decide to work on this project. Be sure to look at the project’s web site, its repositories and any other sources they provide.
  + **Documents:** Identify and give links to any documents relevant to newcomers (e.g. installation, getting started, how to contribute, newcomer threads in communication channels, etc.). Skim each one fairly quickly and give an assessment of it as very clear, clear, confusing, or useless.
  + **Organization:** How well organized is the project overall? Some questions to consider here include: How easy was it to find information as a newcomer to the project? Is it easy to identify the information you might need? Is the amount of information helpful without being overwhelming? Is there out-of-date or redundant information that may be confusing?
  + **Code of Conduct:** Does this project publish a code of conduct for its community? Review the code if it exists and comment on your thoughts of it for governing conduct in the community.
  + **User Installation:** Describe how the product is distributed to users of the product. This may be distributed separately from the source code. Give a link to the download site if one exists. Some projects also provide live demo sites that you can visit for experimentation. Give links to any such demo sites.
  + **Developer Installation:** How do developers looking to contribute to the project get set up to contribute? Give a link to the repository or organization where the source code (e.g. on GitHub/GitLab/etc) can be obtained. Some projects will use multiple sites or migrate from site to site. Some projects will consist of a single repository, others will have multiple repositories within an organization. Be sure you are considering the most current repository or organization for your project. Give links to any documents that describe how to set up a developer environment.
  + **Summary:** Summarize your overall thoughts on the “getting started” materials for this project. Are the materials organized in a way that makes sense to you? Are they written in a way that is accessible to newcomers to the project? Will you be able to find what you need without getting lost or overwhelmed?
* **Code Base:** This section will explore the code base in detail to get a better idea of how active the project is and how open it is to contributions from newcomers. Use information from the organization and/or repositories to answer the following questions:
  + **Repositories:** How many repositories make up the project (1, 2-5, 5-10, >10)? Are they well organized and is it clear what the purpose of each repository is? Select a few (1-3) of the most active repositories and report the following:
    - **Origins:** When was this repository created?
    - **License:** Which open source license is applied to this repository?
    - **Activity:** Answer the following questions to get an idea of how actively this repository is being worked on.
      * **Commits:**
        + When was the most recent commit to this repository?
        + How many commits have been made to this repository in the last few months?
      * **Open Pull/Merge Requests:**
        + How many pull/merge requests are currently open?
        + How many pull/merge requests are older than a week?
        + When was the most recent pull/merge request created?
      * **Closed Pull/Merge Requests:**
        + When was the most recently accepted pull/merge request?
        + How many pull/merge requests have been accepted in the past few months?
        + Do accepted pull/merge requests seem to have timely and helpful discussion?
        + About how long does it take from the time a relatively small pull/merge request is created until it is merged?
    - **Contributors:** How many different people have contributed to this repository in the last year?
    - **New Contributors:** Of those contributors how many have made their first contribution within in the last year?
    - **Top Contributors:** Who are the top 3-5 contributors to this repository? How many commits have they each made in the last year?
    - **Summary:** Summarize your overall thoughts on the code base. Is it being actively worked on by contributors and managed by maintainers? Are things happing too fast or too slow? Are newcomers being successful in making contributions?
* **Tools/Languages/Libraries/Frameworks:** This section looks at the project-specific tools, programming languages, libraries and/or frameworks that are used by the project. Use information from the project’s web page, repositories, forums, etc. to answer the questions below. You may also need to do some external reading about tools, libraries and frameworks to determine their purposes.
  + **Tools:** What specialized tools are used to run, create, build the project (e.g. docker, linters, formatters, maven, npm, node, etc.)? For each tool, give a short explanation of what it does and how it is used in the project.
  + **Languages:** What are the main languages used in the project repositories and approximately what percentage of code is in each language?
  + **Application Libraries/Frameworks:** What libraries and/or frameworks are used in the main project code (e.g. React, Spring, Django, Flask, Express, Bootstrap, etc.)? For each library or framework, give a short explanation of its purpose and estimate how important knowing it will be to contributing to the project.
  + **Testing Libraries/Frameworks:** What libraries and/or frameworks are used for testing in the project (e.g. JUnit, PyUnit, Jest, Mocha, Cypress, etc.)? For each library or framework, give a short explanation of its purpose.
  + **Summary:** Summarize your overall thoughts on the tools, languages, libraries and frameworks used by the project. Are there things that you are already familiar with? Things that you are not familiar with and will need to learn? How hard do you think they will be to learn? Are you excited about and interested in learning them?
* **Assessment**: Give an *overall assessment* of the project based on your review. Be sure to comment on why you are interested (or are no longer interested) in the project, what you believe would facilitate or make it difficult to join this project, and what concerns you have about joining this project. When finished, paste just this *overall assessment* section into the class wiki on Moodle. To do this, first visit the “Project reviews” page. If there are no reviews posted for this project, create a new entry for this project in the bulleted list and a corresponding new wiki page. Now visit the review page for this project (which may have existed previously or you may have just created) and paste in your *overall assessment* of the project, giving your name at the top as the author of the assessment.

**Submission**

* Submit your responses document to Moodle.
* Create or edit the project review wiki page for each project you reviewed, as described in the final instructions above.

**Rubric**

A good grade can be achieved by: following all instructions accurately; producing material that reflects about three hours of effort for each project reviewed (with evidence of time spent on the Slack live log); employing correct grammar and clear presentation of concepts. For an excellent grade, some evidence of additional research, insight, or thoughtfulness may be required.

**Acknowledgements**

This assignment was written by Professor Grant Braught with edits by John MacCormick. It builds from and adapts ideas and content from the following activities created by others:

* https://github.com/ChrisMurphyOnline/open-source-software-development-course/blob/master/activities/foss-evaluation-activity.txt
* http://foss2serve.org/index.php/Intro\_to\_FOSS\_Project\_Anatomy\_(Activity)
* http://foss2serve.org/index.php/FOSS\_Field\_Trip\_(Activity)
* http://foss2serve.org/index.php/Project\_Evaluation\_Activity\_V2
* https://github.com/ChrisMurphyOnline/open-source-software-development-course/blob/master/activities/foss-get-involved.txt