**Epic Project Overview**

In some agile software engineering methodologies, the word “epic” is used to refer to a single large block of a software project. For some optional background reading, you can examine the description of epics by the software company [Atlassian](https://www.atlassian.com/agile/project-management/epics). We will refer to your COMP290 FarmData2 project as your *epic project*.

**Introduction:**

At this point you have developed the fundamental skills and knowledge required to make technical contributions to the FarmData2 project. You have learned:

* how sub-tabs are added to the FarmData2 inside the farmOS user interface.
* some HTML, JavaScript and Vue.js.
* about web APIs, how to use the farmOS API and the FarmData2 convenience methods.
* how to create end-to-end tests using Cypress.
* how to add custom Vue Components to a page and how to access them in tests.

During the remainder of the semester, you will work on issues that are of value to the FarmData2 project. You will be working as part of a team but will also be responsible for demonstrating your individual efforts in several ways. The sections below provide additional details about these team and individual components.

**Project Activities and Deliverables:**

Your team and individual effort during this project work will be assessed based on the following activities and deliverables:

* **FarmData2 Contributions:** Teams will collaboratively produce, test, revise and contribute code that addresses FarmData2 issues.
* **Team Meetings:** Class time will be allocated for meetings between the team and the instructor.
* **Informal team presentations:** Teams may be asked to present their current progress in class. No preparation is required for these presentations.
* **WiD Assignment:** Each individual will complete a Writing in the Discipline (WiD) assignment for this course.
* **Final Team Presentation:** Each team will give a final presentation of their project work, in the exam slot for our course.

For grading purposes, the activities above are split into four graded assignments: EPR1, EPR2, EPR3, and FP. Each of these assignments is described in a separate document. The remainder of this documents gives an overview of information needed for completing all assignments.

**FarmData2 Contributions:**

Teams will collaboratively produce, test, revise and contribute code that addresses FarmData2 issues.

In doing so Teams will gain additional experience with software project *workflows*. Teams will use the *Forking Workflow* introduced in COMP190 when learning about Git and GitHub to make contributions to FarmData2. This workflow used by most open source projects and is also the process described in FarmData2’s CONTRIBUTING.md document. Internally, teams will use a modified *Feature Branch Workflow* to support small group collaboration and provide team members with experience resolving merge conflicts and making pull requests.

These workflows are outlined in the sub-sections below and will be explained in class. The following links are provided for reference if you would like to read more:

* Forking Workflow:
  + <https://www.atlassian.com/git/tutorials/comparing-workflows/forking-workflow>
  + FarmData2 CONTRIBUTING.md:
    - <https://github.com/DickinsonCollege/FarmData2/blob/main/CONTRIBUTING.md>
* Feature Branch Workflow:
  + <https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow>

*1. Setting up the Forking Workflow:*

The team should setup the Forking Workflow by doing the following:

1. Have one team member create a free GitHub organization for the team.

* <https://docs.github.com/en/organizations/collaborating-with-groups-in-organizations/creating-a-new-organization-from-scratch>

1. Invite all team members to be owners of the organization.

* <https://docs.github.com/en/organizations/managing-membership-in-your-organization/inviting-users-to-join-your-organization>

1. Fork the FD2School-FarmData2 repository into the organization. We refer to this new fork as the *origin* repo.

* <https://docs.github.com/en/get-started/quickstart/fork-a-repo#forking-a-repository>

1. Have all team members clone the origin repository into their FarmData2 development environment.
   1. Copy the URL of the repo in your team’s organization using the green “Code” button.
   2. In a terminal in the FarmData2 Development Environment use the commands.

cd ~  
git clone <repo url> COMP290-FarmData2  
cd COMP290-FarmData2  
git set remote upstream https://github.com/DickinsonCollege/FD2School-FarmData2.git

To summarize:

* your origin remote should be at a URL like https://github.com/YourOrganization/FD2School-FarmData2.git
* your upstream remote should be at https://github.com/DickinsonCollege/FD2School-FarmData2.git

Verify this by executing git remote -v.

*2. Find and Understand Your Team’s Issue:*

In class your team will have been assigned a “Good First Issue” to begin working on. The team should:

1. Visit the Issue tracker in the FD2School-FarmData2 upstream repository.
2. Find the “Good First Issue” ticket that has been assigned to your team.
3. All team members should comment on the issue ticket indicating that they will be working on it. This will allow the instructor to assign the issue to your team members.
4. Read your issue ticket and understand what it is asking your team to do.
5. Discuss the issue among your team and develop a strategy for addressing it.
6. If you have questions or want clarification, post a comment on the issue ticket. Your instructor or another FarmData2 community member will respond with additional information.

*3. Work Using the Modified Feature Branch Workflow:*

When your team understands its issue and is ready to begin working, you will be using the modified feature branch workflow. The following steps describe what your team should be doing to follow this workflow:

1. Synchronize your main branch with the upstream.
2. Create a new feature branch based on main with a short but descriptive name for your team’s work on the issue in your origin repo—the FD2School-FarmData2 repository in your team’s organization.
3. Have all team members pull the new feature branch into their local repositories.

git pull origin branchname

1. All team members should work on the issue using the modified feature branch workflow described in class.
   * Be sure to follow the suggestions from class for minimizing merge conflicts.
   * Be sure that commits include a descriptive commit message. You can use multiple lines in your commit messages if a longer description is needed:
     + <https://levelup.gitconnected.com/how-to-commit-multiline-messages-in-git-commit-bcd76f81919c>
   * Be sure to use a co-authored commit if you have collaborated with a teammate on the code in that commit.
     + <https://docs.github.com/en/pull-requests/committing-changes-to-your-project/creating-and-editing-commits/creating-a-commit-with-multiple-authors#creating-co-authored-commits-on-the-command-line>
2. When one team member has pushed a commit to the feature branch on the origin, the team should make a Draft Pull Request for the feature branch to the upstream. The pull request body should describe the work that is being done and contain, on its own line, the statement:  
     
    Closes #xx   
     
   where xx is the number of the issue ticket on which the team is working.
3. The team members should continue working on the issue, being sure to pull, merge changes from their teammates, resolve conflicts and push their own commits to the origin on a regular basis.
4. If clarifications are needed or questions arise members of the team should comment on:
   * the issue ticket, if the question is about the issue to be addressed.
   * the pull request, if the question is about the specific code that is in the draft pull request.

*4. Pull Request to Upstream:*

When a team believes that it has fully addressed its assigned issue it should:

1. Conduct a code review as a team to ensure that:

* All code is neat and consistently indented and aligned.
* Variable and function names are concise, consistent, and meaningful.
* Any code or output added for debugging or incremental development steps has been removed.
* Any unused variables, code or functions have been removed.
* All commented out code has been removed.

1. Convert the pull request from a draft pull request to a regular pull request. One team member should add a comment to the PR saying that it is ready for review and tag the instructor.
2. Move to step “5. Repeat” (below), but also be responsive to any comments received on the PR. For example, if the instructor requests changes to the code, the team should stop work on other issues and return to make the requested changes and push them to update the PR.

Note that if the team’s PR is of high quality, it will then also be considered for merging into the official FarmData2 project—which looks good on a resumé!

*5. Repeat:*

While you are waiting for feedback on your PR or if it has been merged, the team should begin work on another issue. The team should:

1. Review the Issue Tracker looking at tickets that are labeled as “Good Second Issue.” (If your team has already completed a “Good Second Issue” then look for another ticket that is neither labeled as “Good First Issue” nor “Good Second Issue.”)
2. Go Back to step “3. Work Using the Modified Feature Branch Workflow.”

**Team Meetings:**

When a class period includes Team Meetings, each team for 10-15 minutes during the designated class time. During this time the team is expected to discuss:

* what each team member has accomplished since the prior meeting.
* demonstrate the progress that has been made since the prior meeting.
* describe any challenges that are currently being faced by the team.

The instructor may meet with your team, or you may be asked to meet on your own. During team meetings (whether it is with the instructor or not), we will be modeling a software engineering technique known as *standup meetings*. You can get a quick overview from the [Wikipedia page about standup meetings](https://en.wikipedia.org/wiki/Stand-up_meeting). This is a technique used in most Agile software engineering methodologies, especially the Scrum methodology. These concepts will be explained in class.

**Writing in the Discipline (WiD):**

The ability to pose well-formed technical questions using online tools (e.g. issue trackers and forums) is a valuable skill for software developers and computer scientists. You have certainly come across questions on sites like StackOverflow that receive lots of answers and other that receive none. What makes the difference between a question that gets an answer and one that does not is often how the question is written. Writing concise questions that contain the information others need in order to quickly understand the question and give an accurate answer highly increases the likelihood of getting an answer and of getting the answer you are looking for. Experience also suggests that asking the first question is the hardest!

Thus, to give you some practice and to break that first question anxiety, the Writing in the Discipline (WiD) assignment for this course is to:

1. Post at least one technical question on:

* an issue that your team is working on to the issue tracker in the FD2School\_FarmData2 issue tracker; and/or
* a Pull Request that your team has submitted as part of its work.

1. Follow up on any comments, questions, or requests for additional information that are posted in response to your question.
2. Post a PDF of the Issue or PR showing your question, the responses and your follow up to your WiD repository.

Before posting your question, review the following short article that gives good advice on how to pose technical questions and what information to include to maximize the chances of getting a helpful response:

* [https://www.freecodecamp.org/news/how-to-ask-good-technical-questions](https://www.freecodecamp.org/news/how-to-ask-good-technical-questions/)

Here are some optional additional readings:

* <https://opensource.com/life/16/10/how-ask-technical-questions>
* <https://www.10stripe.com/articles/kb/how-to-ask-a-question.php>

**Final Presentation:**

Each team will give a 15-20 minute presentation on their project work. The presentation need not be comprehensive of everything that the team did. However, it must:

* Involve approximately equal participation by all team members.
* Demonstrate the code included in at least one pull request.
* Discuss some of the challenges faced.
* Reflect on what was learned during the project work.

**Acknowledgement:** this assignment is based closely on content authored by Professor Grant Braught. It has been edited by John MacCormick with the permission of Professor Braught.