**08 – FarmData2 and Issue/Bug Trackers (WiD)**

**Activities**

COMP190 – Tools and Techniques for Software Development

Dickinson College

**Name:**

In this set of activities, you will begin to get familiar with the FarmData2 application. You will do some basic tasks with FarmData2 to better understand what it does and how it works. You will then use the FarmData2 issue tracker and explore of the FarmData2 bugs/issues reported there. Finally, you will investigate potential bugs in FarmData2 and add a new ticket in the issue tracker to describe one of them. The writing of this ticket provides practice with technical writing for an audience of software developers and is the writing in the discipline (WiD) activity for this course.

**Information:**

For this assignment you will use a running instance of the FarmData2 application that is available at the URL:

 <http://farmdatadev2.dickinson.edu>

Note: This URL is **only accessible when you are connected to the Dickinson College network** by having your laptop connected to the Dickinson WiFi or by using one of the lab machines.

You will also be using a copy of the FarmData2 repository and Issue Tracker that has been created just for this assignment that is available at the URL:

 <https://github.com/dickinson-comp190/FarmData2-190WiD-S23>

Note that this assignment uses a copy of the FarmData2 repository and Issue Tracker so that you can practice working with bug/issue tickets in a realistic environment. Once you have this practice you are more than welcome to interact with the live FarmData2 project and its issue tracker as well.

**Using the FarmData2 Application:**

The activities in this section will give you a little hands-on experience using the FarmData2 application. This will give you just a small feel for how it works and what it does. It will give you enough experience to begin evaluating the validity and quality of tickets in the issue tracker.

1. Use the URL given above and Log into FarmData2 using the credentials:

* Username: worker1, worker2, worker3, or worker4
* Password: farmdata2

Nothing is required here. Just be sure you are able to log in.

The main features of FarmData2 are contained in the *FieldKit* and the *BarnKit* tabs of the user interface. The elements of the FieldKit are designed to allow workers in the field to quickly and reliably enter new data about activities on the farm. The elements of the BarnKit are designed to support farmers in planning and organic certification tasks by allowing them view and edit reports. The work on FarmData2 so far has implemented features on these tabs support tracking and reporting of Seedings (i.e. planting seeds in a field or a greenhouse). The FieldKit includes a *Seeding Input* form and the BarnKit includes a *Seeding Report*.

2. A *Seeding Report* allows the farmer to retrieve information about the crops that have been seeded (i.e. planted) on the farm. These records can be searched and filtered in a variety of ways. The instance of FarmData2 that you are using is populated with a sample databse that contains anonymized data from all of 2019 and the first half of 2020.

Use the *Seeding Report* sub-tab within the *BarnKit* to answer the following questions.

a. What crop(s) were seeded between April 13th and April 17th in 2020?

b. In which area(s) (i.e. fields) were turnips planted between March 1st and June 30th in 2020?

c. What crops were planted in the CHUAU-2 area between March 1st and June 30th in 2020?

3. When a new crop is planted on the farm a new seeding record is created using the *Seeding Input Form* on *FieldKit* tab.

Use the Seeing Input Form to:

* create a new *tray seeding* that occurred on your most recent birthday.
* Place your name in the comment field.
* You can plant whatever crop you like in whatever area you like.
* Make up values for the seeding and labor fields.

When you have successfully created your new seeding, generate a Seeing Report that shows it. Paste a screen capture of the report showing your new seeding here.

**Bug Reports:**

In class we talked about what makes a good bug report (a.k.a. issue ticket, etc.). We saw the elements that every report should include. We learned about the purpose of each and the qualities that make them good. This section asks you to identify those elements so that you can use them in evaluating some tickets that appear in the FarmData2 issue tracker.

It is not required reading, but you might find a quick read ***How to write effective bug reports*** by Malcom Young a useful resource and an enhancement to what we discussed in class:

* <https://capgemini.github.io/testing/effective-bug-reports>

4. From the class slides, what are the three characteristics of a good bug/issue ticket? Give a sentence elaborating on what each characteristic means.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Attribute** | **Description** |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

5. From the class slides, what are the 5 elements that every good bug report should include? For each element, give a sentence or two elaborating on the qualities and content that make each element good.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Element** | **Description** |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

6. There is an additional element that can be relevant in some situations. What is that element? Briefly describe a situation in which you would want to be sure to include this information.

**The Issue Tracker:**

Use the link given at the top of this assignment to find our copy of the FarmData2 repository and its issue tracker. The issue tracker is in a tab at the top of the GitHub repository page:



This issue tracker currently contains only 22 tickets (i.e. Issues). But projects often have 100’s or even 1000’s of tickets in their issue trackers. In these projects it is essential that the issue tracker provide the project maintainers and developers with ways to filter, label and search for tickets. In this section you will become familiar with some of those features by exploring the Issue Tracker used by FarmData2.

All issue trackers will have some way to search for tickets. The issue tracker provided by GitHub (i.e. the one you are looking at), calls this “Filtering” rather than searching and provides the user interface element shown below.



The text is:issue is:open in the Filters text field indicates that the search will only look for *issue* *tickets* that are *open* (i.e. active and needing work).

You can modify or add additional text to the Filters to make your search more specific. For example the following will search for open issue tickets related to the Seeding Report.



7. When a user or developer observes a behavior in a product that might be a bug the first thing they should do is search the issue tracker to see if that bug has already been reported. Searching first, helps to prevent the creation of *duplicate tickets* describing the same bug. The following questions describe some behaviors that have been observed in FarmData2 and ask you to use the search feature (i.e. Filters) in the issue tracker to find the ticket reporting that behavior.

a. At the bottom of a FarmData2 seeding report there are tables that summarize the Direct Seedings and Tray Seedings in the report. The Direct Seedings Summary looks like:



Most of the information in this table is useful. However, in conversations with Matt (the Farmer who is helping us with FarmData2) indicated that the “Total Row/Bed Planted” does not communicate useful information and should be removed.

i. Use the search bar in the FarmData2 issue tracker to find the ticket that already exists for this bug. Give a search string that matches only the ticket for this issue. Hint: Try a few search strings until you get a set of results that contains the relevant issue. Then modify your search string until the results contain only the relevant issue. You may not use the ticket number or exact title in the search.

Note: Usually you wouldn’t need to be this precise in your search. You would only need to be precise enough to find the issue. However, this is an exercise to give you practice with searching issues and will help you to learn about the types of search string that work well.

 ii. What is the ticket number and title for the ticket that describes this issue?

b. The FieldKit and BarnKit tabs each have sub-tabs for their different features. For example, the FieldKit has tabs for “Info” and the “Seeding Input.” If you look at both the FieldKit and the BarnKit tabs you’ll notice that the ordering of the sub-tabs is inconsistent. The “Info” tab is on the left in the FieldKit and on the right in the BarnKit.

i. Use the search bar in the FarmData2 issue tracker to find the ticket that already exists for this bug. Give a search string that matches only the ticket for this issue. You may not use the ticket number or exact title in the search.

 ii. What is the ticket number and title for the ticket that describes this issue?

8. When tickets in the issue tracker are *open*, it is supposed to indicate that the issue they describe still exists and needs work. Thus, when the issue described by a ticket has been fixed the ticket should be *closed*. Tickets may also be closed for a variety of other reasons including if the issue they describe is found to not actually exist, has become unimportant or is duplicated by another ticket.

Closed tickets are not deleted, but instead are kept as part of the permanent record of the project. Looking at closed tickets can often provide helpful information about the project history. For example, a closed ticket can help explain how and why an issue was fixed (or not fixed). So, when working on an open issue, you might be able to find a closed ticket for a similar issue that has already been fixed. That ticket may contain information that will be helpful to you.

Modify the filter criteria to show all of the closed tickets in the issue tracker.

 a. What text did you use in the search field?

 b. What are the ticket numbers for the closed issues in our issue tracker?

9. You’ll notice that tickets can have “Labels” next to them (e.g. “bug”, “enhancement”, etc.). These labels help to organize the issue tracker and make it easier for maintainers and developers to find issues of interest (e.g. “good first issue”, “documentation”, etc.). The “Label” drop down provides a convenient way to filter the issues by their “Labels”.

Use the Label drop down to list just the tickets labeled as “documentation” and “good first issue”.

a. What text appears in the search field when you filter in this way?

 b. What are the ticket numbers that are “good first issues” focused on “documentation”?

**Bug Gardening:**

Ken Fogel in his book Producing Open Source Software says that “Most ticket databases eventually suffer from the same problem: a crushing load of **duplicate** or **invalid tickets** filed by well-meaning but inexperienced or ill-informed users.” In addition, even the most skilled and dedicated project maintainers will fall behind in managing the tickets in the issue tracker. Some tickets will be of higher quality than others, some will need additional information and some bugs/issues will get fixed but the ticket will not be closed. This leads to clutter in the issue tracker and makes it more difficult for developers to determine what needs to be done.

A neighborhood garden serves as a useful metaphor here. When a garden does not receive regular attention, it can become overgrown with weeds that obscure its real contents and diminish its beauty. By regularly weeding the garden and tending to the flowers that it contains, its beauty is maintained for the community to enjoy. Similarly, when an issue tracker does not receive regular attention, it can become overrun with that “crushing load” of duplicate, invalid or unclear tickets - obscuring its beauty and affecting its usefulness. “Gardening” the issue tracker by identify and remove duplicate tickets, improving unclear tickets and closing invalid tickets or tickets for issues that have been fixed makes the issue tracker more useful to everyone in the project’s developer community.

10. Use the URL for the copy of the FarmData2 repository that was given at the top of this assignment. Do the following for each of the tickets in the table below:

* Find the ticket in the Issue Tracker.
* Use the running instance of FarmData2 to attempt to reproduce the bug that is reported.
* Fill in the “Exists” column to indicate if the issue that is reported in the ticket still exists (“Yes”) or has been fixed (“No”). Note: Two of these issues still exist and two of them have been fixed.
* Fill in the “Quality” column with your rating of the ticket’s quality as “poor”, “good”, or “excellent” based on the criteria that you identified earlier in questions #4 and #5.
* In the “Assessment” column, write a few sentences explaining your quality rating for the ticket. Be specific about what is well done, what could be improved and/or what is missing from the ticket and how that affected your quality rating.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | **Ticket** | **Exists** | **Quality** | **Assessment** |  |
|  | 15 |  |  |  |  |
|  | 17 |  |  |  |  |
|  | 21 |  |  |  |  |
|  | 24 |  |  |  |  |
|  |  |  |  |  |  |

11. When you find an invalid ticket in the issue tracker (i.e. one for a bug that no longer exists) you should add a comment to the ticket to let the project maintainers know that the issue may have been fixed and that if so the ticket can be closed.

Pick one of the bugs that did not exist in the prior question. Then add a comment to the ticket in the issue tracker explaining why you think the bug no longer exists and suggesting that the ticket be closed.

You can find a link to your comment using the “…” that appears at the right side of the header for your comment.



Give the link to your comment here.

12. As mentioned by Ken Fogel, duplicate tickets can often become a problem by cluttering up an issue tracker with repeated information. In question #10 you investigated a number of issues and found that some of them still exist. For one of the issues that still exists there is another ticket (i.e. a *duplicate ticket*) that reports this same issue.

a. Use the search feature in the issue tracker to find the ticket that is the duplicate. **Your search text must find the issue based on its title.** What search text did you use?

b. When you find a duplicate ticket in the issue tracker you should add a comment to the ticket to let the project maintainers know that the issue may be a duplicate and thus could be closed.

Compare the duplicate ticket that you found to the original ticket. Choose the ticket that is “less good” by our criteria and add a comment to that ticket indicating that it is a duplicate of #xx and suggest that it can be closed.

Note, when you replace the xx with the number of a ticket, the issue tracker will automatically create a link to that ticket. For example, saying: This is a duplicate of #172, will automatically create a link to ticket #172.

Give the link to your comment here.

**GitHub Markdown:**

You should recall that one of the things that makes a good bug report is to use nice formatting. Some of the tickets that you read in earlier questions contained images, bulleted and numbered lists that helped to describe the steps required to reproduce the bug. Other tickets might use italics, bold face and links to help in describing the issue.

In the GitHub issue tracker (and many others as well) you can use a language called *Markdown* to format the content of your tickets. In fact, GitHub allows you to use Markdown for formatting pretty much anywhere you are asked to write information.

Use the following GitHub Markdown reference to answer the questions below:

* <https://docs.github.com/en/github/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax>

You can test your work by creating a new issue in the issue tracker and switching between the Write and Preview tabs, or by using a number of different on-line markdown editors (e.g. <https://jbt.github.io/markdown-editor/>).

13. Give a line of Markdown that will accomplish each of the following tasks.

 a. Displays your name using the third largest heading.

b. Displays a sentence containing some **bold** text and then some *italic* text and then some test that is ***both bold and italic at the same time***.

 c. Displays a link with the text Dickinson College that links to the Dickinson home page.

 d. Displays a bulleted list with your three favorite foods.

e. Fixed-width fonts are often used to display commands, directories or short pieces of code so that they stand out. For example. The text below uses a fixed-width font for a filename:

 The README.md file is a good starting point learning about a project.

 Similarly, the line below uses a fixed width font to show a Unix command:

 The command head -n 5 <file> will display the first 5 lines of the <file>.

 Give a line of text that contains some text in a fixed-width font.

f. Blocks of code are also displayed using a fixed-width font because it makes it easy to neatly indent and align the code. For example, the following if statement and a nicely formatted block of code:

 if (x > 0) {

 print “Yep, I’m positive”;

 }

Give markdown that will display this (or some other code) as a block using a fixed-width font.

**Writing a new Ticket:**

As you know, FarmData2 is under active development. Thus, there will be bugs, issues, missing features, etc… that should be documented in the issue tracker when they are found. Listed below are a few *suspect behaviors* that have been observed, but do not yet have bug reports for them in the issue tracker (I know… I searched for them ☺).

Your job in this section will be to write a new bug report ticket for one of these suspect behaviors using the quality guidelines that you have learned.

The Suspect Behaviors:

* The main table in the Seeding Report is supposed to be sorted by the values in the date column. It seems to be most times, but sometimes it is not. The different behaviors seem to be dependent on the date range.
	+ Hint: Do a search for a fairly wide date range until you see the behavior. Then do some more searches narrowing down the date range until you ***find a minimal example*** that illustrates the bug.
* Sometimes in the Tray Seeding Summary table at the bottom of a Seeding Report, the “Total Number of Tray Seeds Planted” and “Average Seeds Planted per Hour” are reported as NaN (Not a Number) instead of the proper values. The different behaviors seem to be dependent on the date range.
	+ Hint: Do a search for a fairly wide date range until you see the behavior. Then do some more searches narrowing down the date range until you ***find a minimal example*** that illustrates the bug.
* Each row of the main table in the Seeding report has an edit button (a pencil icon). Clicking the edit button on some rows seems to cause that row to disappear. The different behaviors seem to be dependent on which row is edited.
	+ Hint: Do a search that has enough results to make the table have a scroll bar. Then try the edit button on some different rows. Then refine your example until you ***find a minimal example*** that illustrates the bug.

For the behaviors that depend on the date range it will also be useful to know that the sample data provided with FarmData2 includes:

* Jan 1, 2019 - July 15, 2020

If you are interested there is a complete description of the sample data available in the FarmData2 repository in the docker/sampleDB/README.md file. Or you can find it in GitHub at:

* <https://github.com/DickinsonCollege/FarmData2/tree/main/docker/sampleDB>

14. Pick one of the suspect behaviors described above and explore it using the running instance of FarmData2 until you find an example that demonstrates the bug.

**Then, reduce your example to a good *minimal example*. Then use your minimal example to create a new excellent quality ticket for the issue in the FarmData2 issue tracker.**

Give the title and URL of the ticket that you created as your answer for this question. Your instructor will review your ticket in the issue tracker.

**Upload to Your WiD Repository:**

COMP 190 is part of the Writing in the Discipline (WiD) thread that runs through the computer science major. In each course on the WiD thread you will complete a writing assignment and add it to your WiD Repository on GitHub. The bug report that you have written is the Writing in the Discipline (WiD) assignment for this course.

15. A WiD Repository has been created for you on GitHub. If this is your first WiD course in computer science you will have received an e-mail inviting you to join that repository. Please accept that invitation. Log into GitHub and then visit the Dickinson-COMP-WiD organization on GitHub to find your WiD repository:

* <https://github.com/Dickinson-COMP-WiD>

If you do not see your repository in that organization, check your e-mail (including your spam folder) for the invitation. If you still cannot find the invitation, get in touch with your instructor as soon as possible to resolve the situation.

Give the URL of your WiD repository on GitHub.

16. The following set of instructions will guide you through the process of uploading a pdf of your ticket to your WiD repository:

* Find your issue ticket in the repository used for this assignment.
* Print the issue ticket to a pdf file from your browser.
* Log into GitHub with your username and password.
* Navigate to your WiD repository in the Dickinson-COMP-WiD Organization using the link below:
	+ <https://github.com/Dickinson-COMP-WiD>
* Create a folder named COMP190 for this course in your repository. See the link below for detailed instructions:
	+ <https://github.com/KirstieJane/STEMMRoleModels/wiki/Creating-new-folders-in-GitHub-repository-via-the-browser>
* Drag and drop your pdf file into the COMP190 folder that you created. See the link below for detailed instructions:
	+ <https://help.github.com/en/github/managing-files-in-a-repository/adding-a-file-to-a-repository>
* If you have problems creating the COMP190 folder or uploading your PDF please see your instructor for assistance.

**Optional:** To help us improve and scope these activities for future semesters please consider providing the following feedback.

a. Approximately how much time did you spend on this activity outside of class time?

b. Please comment on any particular challenges you faced in completing this activity.