Digital Display Garden

https://ddg.wcroc.umn.edu/bed/ALL

The Project:

Description:

A website application that was built for the West Central Research Outreach Center (WCROC) located in Morris, MN. The website was designed to enhance visitor experience in their horticulture garden and to retrieve feedback for WCROC. Key features of the project are split into two portions: The Visitor Interface and Administration Interface.

Visitor Interface: Visitors will be able to access the website through QR Codes located at a bed of flowers. Once at the website there are a plethora of actions a visitor can take such as rate, comment and view information about any of the garden plants (Figure 1).

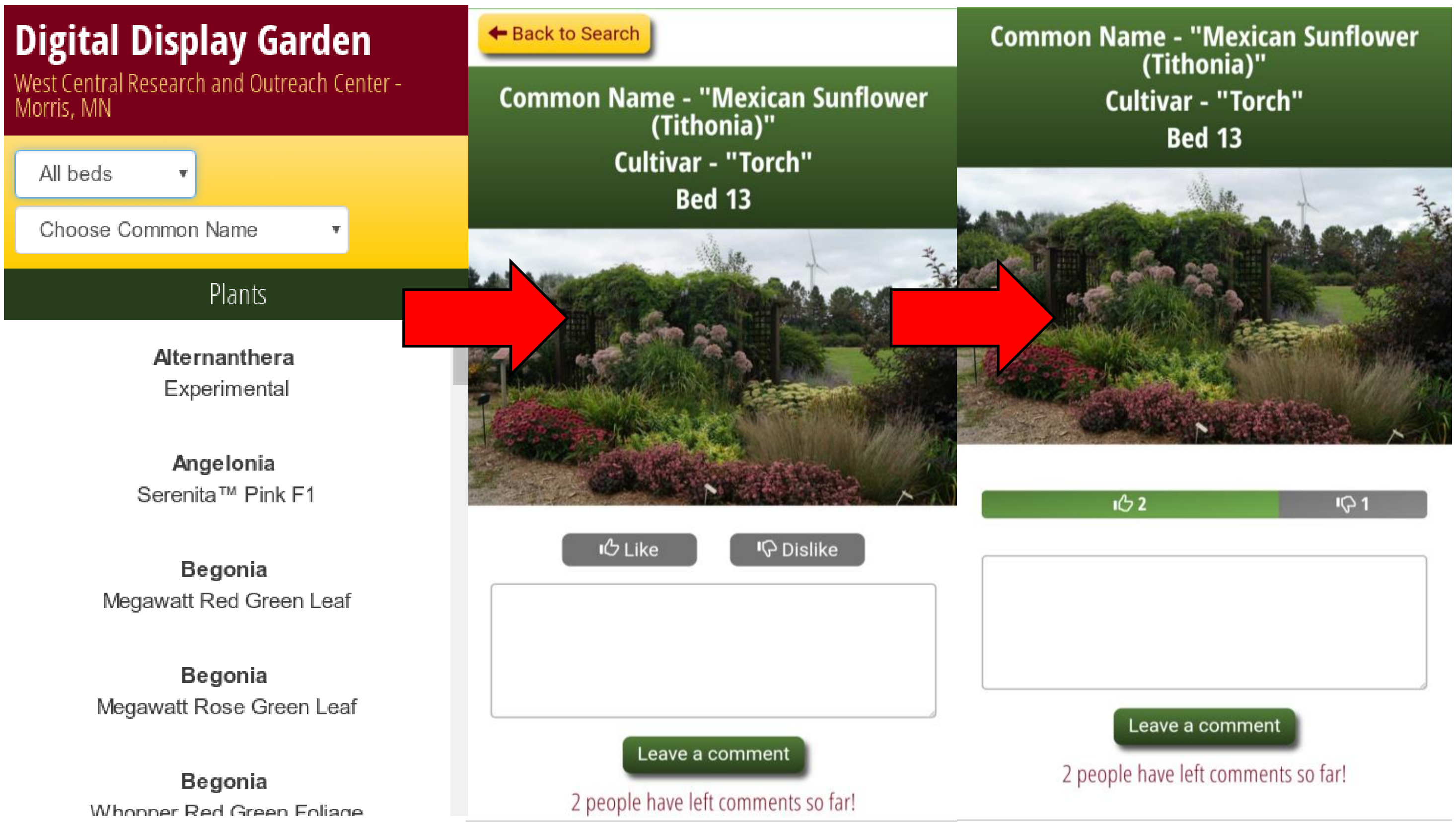


Figure 1: A glance from the starting the screen to selecting a flower to rating and commenting on a plant

Administration Interface: All the while administrators at WCROC can track when and

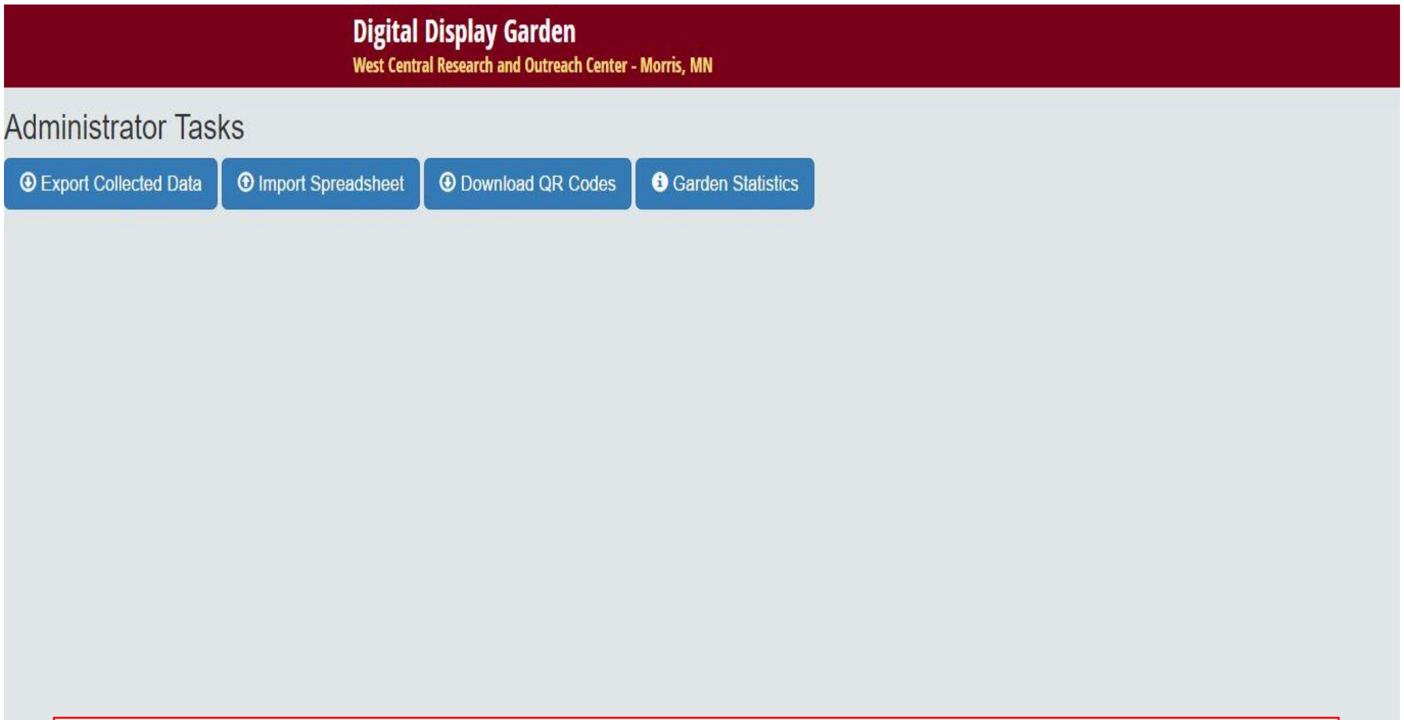


Figure 2: Administration Page with the functionality that it offers Exporting, Importing, Downloading QR codes and viewing Garden Statistics

where in the garden people are viewing, liking, and commenting on plants through interactive graphics, and exportable spreadsheets. Additionally, administrators can maintain the website through simple functionality of importing a spreadsheet, updating the website without losing data, and downloading QR codes for each bed (Figure 2).

For further details visit the Github repository:
https://github.com/UMM-CSci-3601-S17/digital-display-garden-iteration-4-revolverenguardia-1

Compilation:
*The Digital Display Garden was built at University of Minnesota Morris, by the CSCI 3601 Software Design and Development class in spring 2017 by multiple groups. The project that is represented was made by Team Revolver en Guardia++. The final team was made up of 8 people.

Approach: Agile Process

We used Agile Development to build the Digital Display Garden implementing multiple techniques including:

- **Inception Deck:** Full group meeting with the customers where we discussed what the project was, what is wasn’t, how much work everything would be, and what it was going to take to finish the product.
- **Story Planning:** Full group meeting after inception deck where we turned and brainstormed what the customer wanted into doable features.
- **Stand Up Meetings**: Iteration group meetings that lasted about 5 – 10 minutes where we discussed each feature as an overall progress report, as well as requested assistance.
- **Test Driven Development:** Technique we aimed to use, where we would write test before writing the functionality of the features.
- **Estimation:** Technique used to make the best educated guess about how much time and work a certain feature would ‘cost’ to implement fully including functionality as well as testing.
- **Refactoring:** Technique to make our code both cleaner to read and well commented about functionality.
- **Continuous Integration**: Technique used to keep track of failing test on any part of the project as well maintain code reliability.

Tools and Platforms Used:

Languages used
Java: Language used on the server side
TypeScript: Language primarily used on client side

Version Control
Git/GitHub: used to keep track and maintenance among files for the entire team.

Group Communication
Slack: used for communicating about progress, updates and team meetings.
Zenhub: used to keep track of progress certain stories(features) and where they stand in terms of being ‘done’.

IDE:
IntelliJ: Tool used for both client and server side development.

Contiguous Integration
Travis CI: service used to build and test our product during development.

Libraries used
Client Side
Angular 2: Client side framework for building single page apps.
Jasmine and Karma: Testing framework and runner used for testing
Gradle: used to tell Yarn to orchestrate the client side.
Yarn: Package manager.

Server Side
Java Spark Platform: used for server operations
MongoDB: noSQL database used to store user metadata and information.
Gradle: Build processes to build entire project.
JUnit: used for unit testing.
Apache: used for importing and exporting data in .xlsx format.
Zxing: used for generating QR codes (supports reading them if we want)
joda: used for making unique LiveUploadID(unique identifier of spreadsheets).

Personal Contribution:

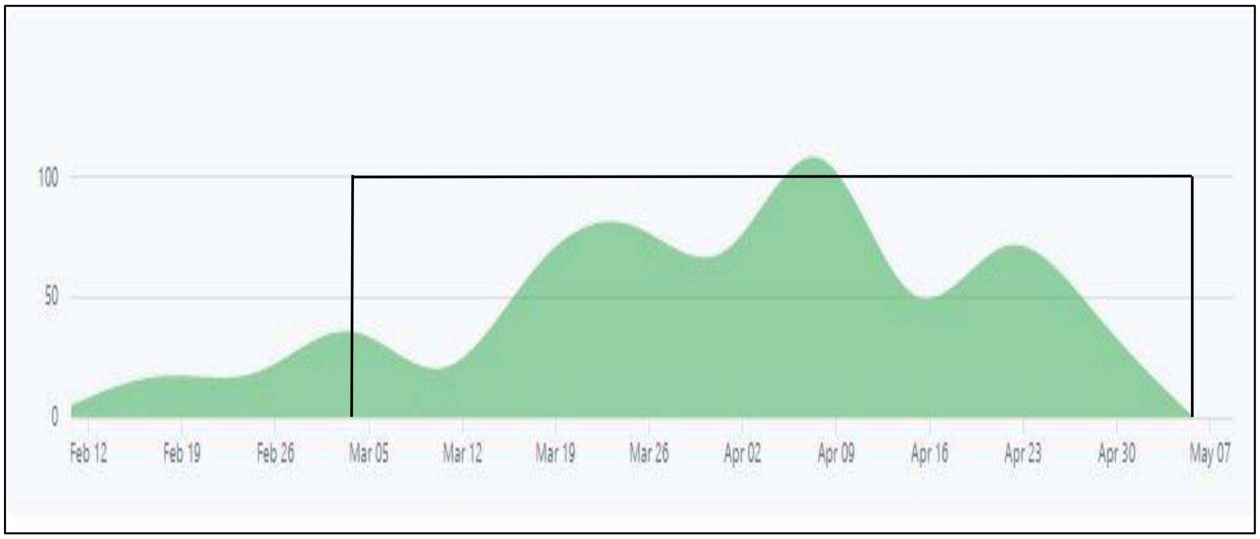
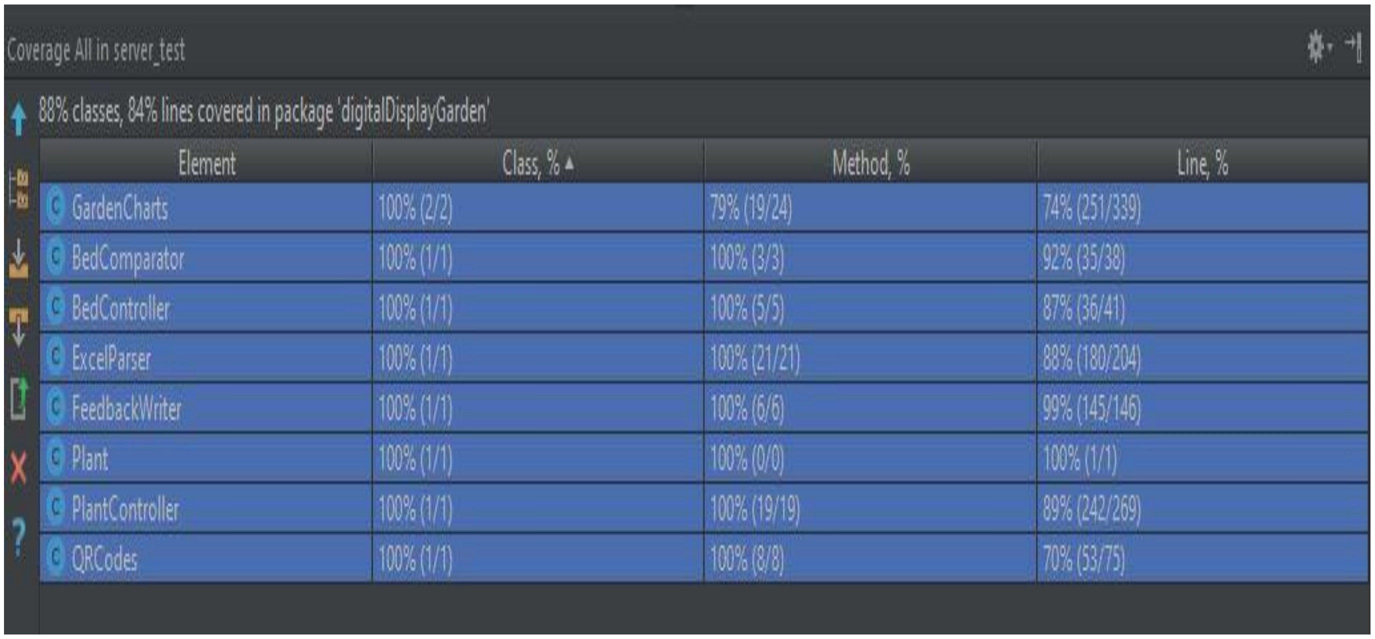


Figure 3: Total Commits of project marked from start to finish of (March 3rd – May 5th 2017)

*A note of worthiness: Our class and each Team within it was personally invited to present our progress of the Digital Display Garden to the Horticulture Advisory Committee, a group that provides WCROC with assistance in their education planning, visioning for the garden and various other task.

Test Coverage: Testing is very important for any software to prosper. The Digital Display Garden is no different. Following the agile approach of Test Driven Development, we tested our code heavily, both client and server. I worked heavily in writing server side JUnit test for methods that would exist as well as missed test for methods that were already written. The IDE we used let us see how much actual test coverage we had on our server side (Figure 4)



Coverage All in server_test

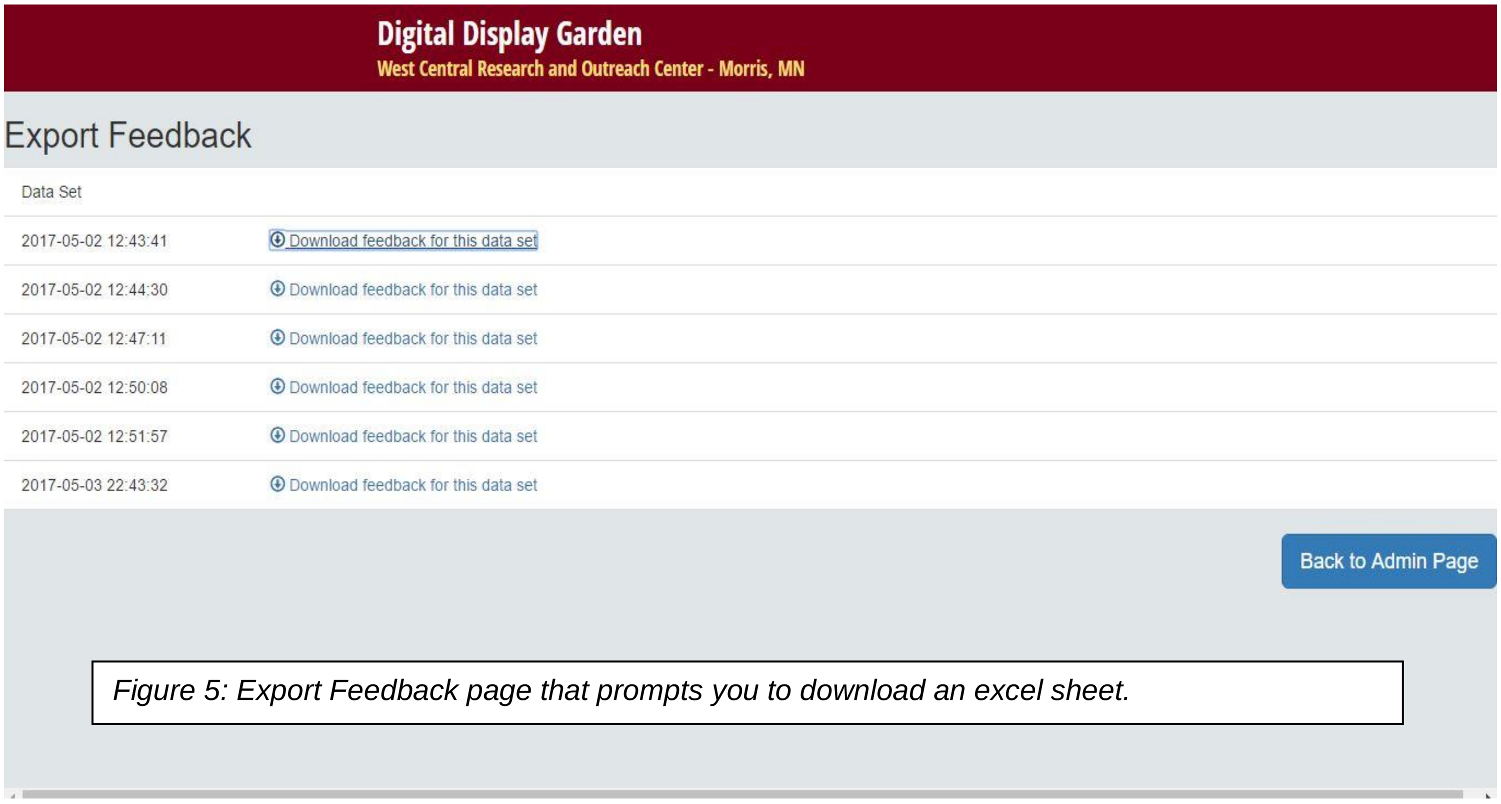
88% classes, 84% lines covered in package 'digitalDisplayGarden'

Element	Class, % ▲	Method, %	Line, %
GardenCharts	100% (2/2)	79% (19/24)	74% (251/339)
BedComparator	100% (1/1)	100% (3/3)	92% (35/38)
BedController	100% (1/1)	100% (5/5)	87% (36/41)
ExcelParser	100% (1/1)	100% (21/21)	88% (180/204)
FeedbackWriter	100% (1/1)	100% (6/6)	99% (145/146)
Plant	100% (1/1)	100% (0/0)	100% (1/1)
PlantController	100% (1/1)	100% (19/19)	89% (242/269)
QRCodes	100% (1/1)	100% (8/8)	70% (53/75)

Figure 4: IntelliJ's Test Coverage feature which gives not only Total Percent of how much your code is covered but in great detail.

Export Feedback: I worked heavily a pair programming group with another group member for this feature. This feature works by clicking on *Export Data* on the Admin page, then the user will be prompted to *"Download feedback for this data set"* with a certain date relating to the last import or update of the website as seen in *Figure 5*. Once the download starts feedback will be exported in an Excel Spreadsheet containing three sheets: Comments, Plant Metadata, and Bed Metadata (*Figure 6).*

The code that does the named processes is in FeedbackWriter as well as PlantController spread-out through various methods.



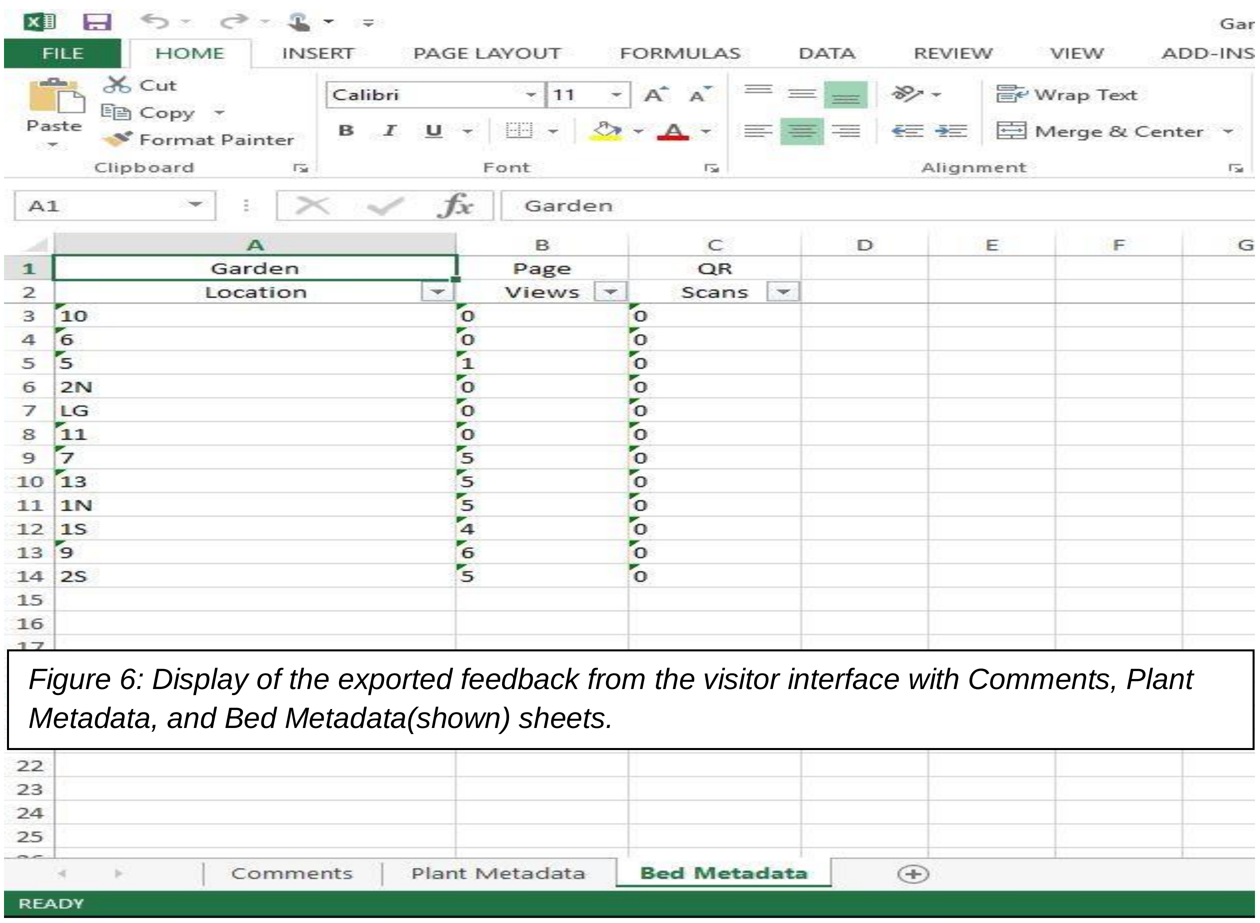
Digital Display Garden

West Central Research and Outreach Center - Morris, MN

Export Feedback

Data Set	
2017-05-02 12:43:41	Download feedback for this data set
2017-05-02 12:44:30	Download feedback for this data set
2017-05-02 12:47:11	Download feedback for this data set
2017-05-02 12:50:08	Download feedback for this data set
2017-05-02 12:51:57	Download feedback for this data set
2017-05-03 22:43:32	Download feedback for this data set

Figure 5: Export Feedback page that prompts you to download an excel sheet.



Garden Location	Page Views	QR Scans
10	0	0
6	0	0
5	1	0
2N	0	0
LG	0	0
11	0	0
7	5	0
13	5	0
1N	5	0
1S	4	0
9	6	0
2S	5	0

Figure 6: Display of the exported feedback from the visitor interface with Comments, Plant Metadata, and Bed Metadata(shown) sheets.