Deliverable 1

Our top 3 Candidates:

Cadasta: "Technology to help communities document their land rights around the world."

Main website: http://cadasta.org/

GitHub: https://github.com/Cadasta/cadasta-platform

Martus: "Martus, the Greek word for "witness," is a software tool that allows users to document incidents of abuse by creating bulletins, and storing them on redundant servers located around the world. Using Martus helps countries torn apart by civil conflicts come to a consensus and rational understanding of their histories, leading to reconciliation and reform processes."

Main website: https://www.martus.org/

GitHub: https://github.com/benetech/Martus-Project

Insight Segmentation and Registration Toolkit (ITK): "National Library of Medicine: ITK is an open-source, cross-platform system that provides developers with an extensive suite of software tools for image analysis. Developed through extreme programming methodologies, ITK employs leading-edge algorithms for registering and segmenting multidimensional data."

Main website: https://itk.org/

GitHub: https://github.com/InsightSoftwareConsortium/ITK

Reasoning:

Cadasta caught our attention, because it is an interactive map to document land rights around the world. This is particularly interesting because of the political implications of land rights, especially in regions of the world such as the Middle East. Cadasta is primarily implemented in Python, and its GitHub repo has just under 1,000 commits.

Martus caught our attention, because it is software aimed at aiding less-developed countries in documenting and understanding reports of abuse. Additionally, it was interesting that Martus emphasizes security and redundant

servers. Martus is implemented in Java, and is hosted on a much smaller repository than Cadasta or ITK.

ITK caught our attention, because it's aimed at aiding devlopers with image analysis. Likewise, it is not enduser software, but a library given to developers that are building tools for endusers in the medical field. It's a huge project with over 45,000 commits, so this would provide invaluable experience for our team with working with large projects. Plus, it's an excuse to learn C++.

For this deliverable, we were to clone our chosen project’s GitHub repository to our local machines and manually build the code. At first, it seemed like we would be able to do this with our original selection, Martus, but we struggled with compiling the code. Further research, we discovered enough evidence to suggest that the Martus project may be dead. Many of the dependencies that were required for compiling and running this code were out-of-date, and many of them were not hosted for download any longer. Given that we could not compile the code independently and that the project was dead, we decided to consider changing projects.
We revisited the other project candidates listed in the previous chapter. This time, Cadasta, the land-rights documentation program, caught out attention, but we were concerned with ensuring that this project was not dead. Our team noticed there were over one-thousand commits, and they were as recent as the previous week. Additionally, the dependencies were up-to-date and their GitHub Wiki had an easily understood installation instructions for development. Essentially, to install Cadasta for development, the user creates a Vagrant virtual machine to run as their Cadasta server, and then they can SSH into the server to run commands.

What we found inside was that Cadasta had a fairly comprehensive testing framework already implemented. A Python file(runtests.py) runs all of their integrated tests and reports the results. Additionally, they provide the option to export these test results to HTML files. After the tests are finished, it reports a quick overall result: "1998 passed, 501 warnings in 972.86 seconds."

When examining the test result's directory, what you find is a collection of many HTML files that correspond to each unit of the software. At the bottom, is an HTML file labeled "index" that provides a quick overview of all test results. If you wish to view a specific test, however, you can browse to that HTML file specifically. What this does is it allows a developer to get a quick, comprehensive overview of all the unit tests, but still allows them to delve deeper and get a granular view of the unit tests.

Altogether, Cadasta appears to be a much more approachable project that is aimed at OSS development. The easy installation instructions combined with a well-maintained code base give much more confidence than Martus. Additionally, the current testing framework inspires ideas for furthering the testing database.

However, with Cadasta, we would continue to run into unexpected complexity. Cadasta, being a web application, relied very heavily on Django, the Python-written web framework. It would turn out that handling the Django imports and executing the tests would prove too difficult to write an automated testing framework. After this unexpected complexity became too great of a roadblock, we, once again, reevaluated our project and decided to change.

At that point in development of our automated testing framework. We had coded a testing framework that was generic enough to theoretically handle any Python-written code base given that the user written the test cases in the proper syntax. Additionally, our testing framework’s code base was large enough to support the number of test cases needed for the project’s specifications. After consideration and consultation with Dr. Bowring, we changed our project to test itself.